



# UNIVERSITY OF PELOPONNESE

Faculty of Economics and Technology,  
Department of Informatics and Telecommunications

PhD Thesis

“Model-driven Software Architectural Design based on  
Software Evolution Modeling and Simulation and Design  
Pattern Analysis for Design Space Exploration Towards  
Maintainability”

by  
Chris Karanikolas

Supervisor: Konstantinos Masselos

A dissertation submitted in partial fulfillment of the requirements for the  
degree of Doctor of Philosophy in Computer Science

November 2022

Theses / Dissertations remain the intellectual property of students (“authors/creators”), but in the context of open access policy they grant to the UOP a non-exclusive license to use the right of reproduction, customization, public lending, presentation to an audience and digital dissemination thereof internationally, in electronic form and by any means for teaching and research purposes, for no fee and throughout the duration of intellectual property rights. Free access to the full text for studying and reading does not in any way mean that the author/creator shall allocate his/her intellectual property rights, nor shall he/she allow the reproduction, republication, copy, storage, sale, commercial use, transmission, distribution, publication, execution, downloading, uploading, translating, modifying in any way, of any part or summary of the dissertation, without the explicit prior written consent of the author/creator. Creators retain all their moral and property rights.

---

## *Acknowledgments*

---

I would like to thank all people who have contributed to the accomplishment of my doctoral studies.

Initially, I would like to thank my supervisor, professor at University of Peloponnese, Konstantinos Masselos about his scientific guidance and support at my Ph.D. studies as well as for his useful suggestions all these years.

I would like also to thank and express my appreciation to Dr. Grigoris Dimitroulakos who provided me with his knowledge on architectural design of compilers and his assistance not only during the period of my Ph.D. studies, but also during the elaboration of my postgraduate thesis.

Many thanks to my classmate at University of Peloponnese, Dr. Chris Lezos for helping and inspiring me during my first steps in diligent research.

Finally, I would like to thank my wife and my parents for their invaluable support during the time of my postgraduate studies as well as my children for their patience during this persistent effort.

## Abstract

In software architectural design, critical decisions among design alternatives with regards to maintainability arise early in the software design cycle. Existing exploration approaches are neither design-pattern-oriented nor formal. Such approaches are not reusable, have narrow scoping, and usually lead to suboptimal results. Furthermore, the effectiveness of existing techniques and models on predicting maintenance effort is usually verified on a limited number of case studies under heterogeneous metrics and settings. Conventionally, developers use their intuition, experience, or instant judgment to get such decisions, which leads to accrued technical debt, high risk, and suboptimal results regarding code quality. Owing to the confirmed lack of architectural awareness, developers underestimate the negative impact of such early and critical design decisions. Selecting between different design options is a crucial decision for object-oriented software developers affecting important code quality characteristics such as maintainability.

In this thesis, a systematic modeling method for deriving formal comparison models for the efficient evaluation of object-oriented design alternatives in terms of maintainability early in the software design cycle is introduced. The method is suitable for modeling significant, general, and frequently tackled design problems which have dominant impact on the overall maintainability of the system, where different design alternatives are competing to address the same requirements. The derived formal models provide early estimates of required effort per design alternative in terms of proportionally equivalent effort assessments mainly for comparison purposes. The proposed approach considers the software expansion trend through the structural evolution of the engaged design patterns. This is achieved by formalizing change rates of individual design attributes for basic maintenance scenarios and their probabilities in the form of continuous differential equations to predict the required maintenance effort. Alternatively, the required effort is assessed by measuring the change impact of repeatedly applied scenarios in connection with the evolving design attributes under the view of a gradual (event-based) quantitative analysis. The proposed method has been evaluated on the significant and general design problem of recursive hierarchies of part-whole aggregations. The generated formal comparison models address the selection of Visitor over Composite design patterns against the direct inheritance-based approach. The derived models capture maintainability as a metric of software quality and provide reliable assessments for each implementation alternative. Furthermore, the proposed method suggests the structural maintenance cost metric based on which the progressive analysis of maintenance process is introduced. The proposed measurement approach has been applied to several test cases for different relevant quality metrics. The results prove that the proposed modeling method derives formal models which deliver reliable effort assessments mainly for comparison purposes. Thus, the proposed method can be used for comparing different implementation alternatives against various measures and quality factors, before code development leading to reduced effort and cost for software maintenance.

Furthermore, the introduced modeling method has been applied to three different extensions of the general selection problem, thus assessing its applicability to even more realistic settings. The Decorator, Mediator, Observer, Abstract Factory, and Prototype design patterns have been modeled. The generated formal models have been tested on a sample of several specific instances representing the entire design-space of each general problem. The results prove that the derived formal models are reliable and can efficiently support decision-making among design alternatives early in the design cycle, leading to significant benefits in terms of maintenance time and effort. The results also suggest that the method can model general problems and support decision-making even in the (high-level) architectural design stage of systems.

In addition, a multi-variable simulation model for validating the decision-making reliability of the modeling theory and derived formal comparison models for the significant designing problem of recursive hierarchies of part-whole aggregations is introduced. The proposed simulation model has been implemented in the forms of functional and modular representations. In the absence of a strictly validation against real-world observations, the simulation model has been thoroughly calibrated concerning its decision-making precision based on empirical evidence from time series analysis, approximating the highly uncertain nature of actual maintenance process. The decision reliability of the formal models has been statistically validated on a sample of one thousand possible instances of design attributes representing the entire design-space of the problem under analysis. Despite the limited accuracy of measurements, the results show that the models demonstrate an increasing selection reliability in a long-term perspective even under assumptions of high variability. Thus, the proposed modeling theory delivers reliable formal comparison models that significantly reduce decision-risk, maintenance effort, and relevant cost. Methods that yield such formal, general, and reusable models can bring software engineers closer to informed design decisions, and thus develop more maintainable software of higher quality.

### **Keywords**

Architectural design, Model-driven software engineering, Software design engineering, Design optimization, Object-oriented architectures, Software evolution, Maintainability, Design pattern analysis, Software design tradeoffs, Equational models, Modeling and simulation, Statistical validation, Classes and objects, Visitor, Composition.

## Περίληψη

Στην αρχιτεκτονική σχεδίαση λογισμικού και κατά την διάρκεια του πρόωρου σταδίου σχεδίασης, προκύπτουν κρίσιμες αποφάσεις μεταξύ εναλλακτικών σχεδίασης αναφορικά με την συντηρησιμότητά τους. Οι υπάρχουσες διερευνητικές προσεγγίσεις δεν είναι προσανατολισμένες στα σχεδιαστικά πρότυπα ούτε επαρκώς τυπικές. Τέτοιου είδους προσεγγίσεις δεν είναι επαναχρησιμοποιήσιμες, έχουν περιορισμένο πεδίο εφαρμογής και συνήθως καταλήγουν σε μη βέλτιστα αποτελέσματα. Περαιτέρω, η αποτελεσματικότητα των υπάρχοντων τεχνικών και μοντέλων αναφορικά με την πρόβλεψη της προσπάθειας συντήρησης, συνήθως επιβεβαιώνεται από ένα περιορισμένο αριθμό περιπτώσιολογικών μελετών υπό το πρίσμα ετερογενών μετρικών και συνθηκών. Συμβατικά, οι προγραμματιστές χρησιμοποιούν την διαίσθησή, την εμπειρία ή την στιγμιαία κρίση τους προκειμένου να λάβουν τέτοιου είδους αποφάσεις, γεγονός που οδηγεί σε συσσώρευση τεχνικού χρέους (αστοχιών), υψηλό ρίσκο και μη βέλτιστα αποτελέσματα αναφορικά με την ποιότητα του κώδικα. Λόγω της επιβεβαιωμένης έλλειψης αρχιτεκτονικής συνείδησης, οι προγραμματιστές υποτιμούν τον αρνητικό αντίκτυπο τέτοιου είδους πρόωρων και κρίσιμων σχεδιαστικών αποφάσεων. Η επιλογή ανάμεσα σε διαφορετικές σχεδιαστικές εκδοχές είναι μια κρίσιμη απόφαση για τους προγραμματιστές του αντικειμενοστραφούς λογισμικού που επηρεάζει σημαντικά ποιοτικά χαρακτηριστικά του κώδικα όπως η συντηρησιμότητά του.

Σε αυτή την διατριβή εισάγεται μια συστηματική μέθοδος μοντελοποίησης για την άντληση τυπικών μοντέλων σύγκρισης για την αποτελεσματική αξιολόγηση αντικειμενοστραφών εναλλακτικών σχεδίασης, υπό όρους συντηρησιμότητας, κατά το πρόωρο στάδιο σχεδίασης του λογισμικού. Η μέθοδος είναι κατάλληλη για την μοντελοποίηση σημαντικών, γενικών και συχνά εμφανιζόμενων σχεδιαστικών προβλημάτων τα οποία έχουν κυρίαρχο αντίκτυπο στη συνολική συντηρησιμότητα του συστήματος, όπου διαφορετικές σχεδιαστικές προσεγγίσεις ανταγωνίζονται για την αντιμετώπιση κοινών απαιτήσεων. Τα προκύπτοντα επίσημα μοντέλα παρέχουν πρόωρες εκτιμήσεις της απαιτούμενης προσπάθειας ανά εναλλακτική σχεδίαση υπό όρους αναλογικά ισοδύναμων εκτιμήσεων προσπάθειας, κυρίως για σκοπούς σύγκρισης. Η προτεινόμενη προσέγγιση λαμβάνει υπόψη την τάση επέκτασης του λογισμικού μέσω της δομικής εξέλιξης των εμπλεκόμενων σχεδιαστικών προτύπων. Αυτό επιτυγχάνεται διαμέσου των ρυθμών αλλαγής μεμονωμένων χαρακτηριστικών σχεδιασμού για βασικά σενάρια συντήρησης και για τις πιθανότητες τους υπό τη μορφή συνεχών διαφορικών εξισώσεων προκειμένου να προβλέψουν την απαιτούμενη προσπάθεια συντήρησης. Εναλλακτικά, η απαιτούμενη προσπάθεια εκτιμάται δια της μέτρησης του αντικτύπου αλλαγής των επαναλαμβανόμενα εφαρμοζόμενων σεναρίων σε σχέση με τις μεταβαλλόμενες ιδιότητες σχεδίασης υπό το πρίσμα μιας βαθμιαίας (με βάση τα γεγονότα) ποσοτικής ανάλυσης. Η προτεινόμενη μέθοδος αξιολογήθηκε για το σημαντικό και γενικό πρόβλημα σχεδίασης των αναδρομικών ιεραρχιών συναθροίσεων (μέρους-όλου). Τα παραγόμενα τυπικά μοντέλα σύγκρισης αντιμετωπίζουν την επιλογή ανάμεσα στο συνδυασμό των σχεδιαστικών προτύπων «Visitor» και «Composite» και στην άμεση προσέγγιση που βασίζεται στην κληρονομικότητα του «Composite» σχεδιαστικού προτύπου. Τα παραγόμενα μοντέλα καταγράφουν την συντηρησιμότητα ως μια μετρική της ποιότητας λογισμικού και παρέχουν αξιόπιστες εκτιμήσεις για κάθε εναλλακτική υλοποίηση. Περαιτέρω, η προτεινόμενη μέθοδος εισαγάγει την μετρική του δομικού κόστους συντήρησης με βάση την οποία παρουσιάζεται η προοδευτική ανάλυση της διαδικασίας συντήρησης. Η προτεινόμενη μέθοδος μέτρησης εφαρμόστηκε σε πολλές δοκιμαστικές περιπτώσεις για διαφορετικές μετρικές ποιότητας. Τα αποτελέσματα αποδεικνύουν ότι η προτεινόμενη μέθοδος μοντελοποίησης εξάγει τυπικά μοντέλα τα οποία παρέχουν αξιόπιστες εκτιμήσεις προσπάθειας κυρίως για σκοπούς σύγκρισης. Έτσι, η προτεινόμενη μέθοδος μπορεί να χρησιμοποιηθεί για την σύγκριση διαφορετικών εναλλακτικών υλοποιήσεων έναντι διαφόρων μετρικών και παραγόντων ποιότητας, πριν

το στάδιο ανάπτυξης του κώδικα, οδηγώντας στη μείωση της προσπάθειας και του κόστους συντήρησης του λογισμικού.

Περαιτέρω, η εισαγόμενη μέθοδος μοντελοποίησης εφαρμόστηκε σε τρεις διαφορετικές επεκτάσεις του γενικού προβλήματος επιλογής, αξιολογώντας έτσι την ευκολία εφαρμογής της σε ακόμη πιο ρεαλιστικές συνθήκες. Τα σχεδιαστικά πρότυπα “Decorator”, “Mediator”, “Observer”, “Abstract Factory”, και “Prototype” μοντελοποιήθηκαν για το σκοπό αυτό. Τα παραγόμενα τυπικά μοντέλα δοκιμάστηκαν σε ένα δείγμα πολλαπλών στιγμιότυπων που αναπαριστούν το σύνολο του σχεδιαστικού χώρου του εκάστοτε γενικού προβλήματος. Τα αποτελέσματα αποδεικνύουν ότι τα εξαγόμενα τυπικά μοντέλα είναι αξιόπιστα και μπορούν να υποστηρίξουν αποδοτικά τη λήψη σχεδιαστικών αποφάσεων μεταξύ εναλλακτικών σχεδίασης κατά το πρόωρο στάδιο σχεδίασης, προσφέροντας σημαντικά οφέλη υπό όρους χρόνου και προσπάθειας συντήρησης. Επίσης, τα αποτελέσματα υποδεικνύουν ότι η μέθοδος μπορεί να μοντελοποιήσει γενικά προβλήματα καθώς και να υποστηρίξει τη λήψη αποφάσεων ακόμη και στο στάδιο της (υψηλού επιπέδου) αρχιτεκτονικής σχεδίασης συστημάτων.

Επιπρόσθετα, παρουσιάζεται ένα πολύ παραμετρικό μοντέλο εξομοίωσης για τον έλεγχο της αξιοπιστίας λήψης αποφάσεων της προτεινόμενης θεωρίας μοντελοποίησης και των παραγόμενων τυπικών μοντέλων σύγκρισης, αναφερόμενο στο σημαντικό σχεδιαστικό πρόβλημα των αναδρομικών ιεραρχιών των συναθροίσεων μέρους-όλου. Το προτεινόμενο μοντέλο εξομοίωσης υλοποιήθηκε στις μορφές της συναρτησιακής και αρθρωτής αναπαράστασης. Εν’ απουσία ενός αυστηρού ελέγχου έναντι πραγματικών παρατηρήσεων, το μοντέλο εξομοίωσης βαθμονομήθηκε διεξοδικά αναφορικά με την ακρίβεια του στη λήψη αποφάσεων με βάση εμπειρικά στοιχεία από την ανάλυση χρονοσειρών, προσεγγίζοντας έτσι την εξαιρετικά αβέβαιη φύση της πραγματικής διαδικασίας συντήρησης. Η αξιοπιστία των αποφάσεων των τυπικών μοντέλων ελέγχθηκαν στατιστικά σε ένα δείγμα χιλίων πιθανών περιπτώσεων των σχεδιαστικών ιδιοτήτων, αναπαριστώντας ολόκληρο το σχεδιαστικό χώρο του υπό ανάλυση προβλήματος. Παρά την περιορισμένη ακρίβεια των μετρήσεων, τα αποτελέσματα έδειξαν ότι τα τυπικά μοντέλα επιδεικνύουν μια αυξανόμενη αξιοπιστία λήψης αποφάσεων σε μια μακροπρόθεσμη προοπτική ακόμη και υπό υποθέσεις αυξημένης μεταβλητότητας. Έτσι, η προτεινόμενη θεωρία μοντελοποίησης προσφέρει αξιόπιστα τυπικά μοντέλα σύγκρισης τα οποία μειώνουν σημαντικά τον κίνδυνο λήψης λανθασμένων αποφάσεων, την προσπάθεια συντήρησης, και το σχετικό κόστος. Μέθοδοι που αποδίδουν τέτοια τυπικά, γενικά και επαναχρησιμοποιήσιμα μοντέλα μπορούν να φέρουν τους μηχανικούς λογισμικού πιο κοντά σε τεκμηριωμένες αποφάσεις σχεδιασμού και έτσι να αναπτύξουν πιο συντηρήσιμο λογισμικό υψηλότερης ποιότητας.

### Λέξεις – Κλειδιά

Αρχιτεκτονικός σχεδιασμός, Υποβοηθούμενη από μοντέλα μηχανική λογισμικού, Μηχανική σχεδίασης λογισμικού, Βελτιστοποίηση σχεδιασμού, Αντικειμενοστραφής αρχιτεκτονικές, Εξέλιξη λογισμικού, Συντηρησιμότητα, Ανάλυση σχεδιαστικών προτύπων, Αντισταθμίσεις σχεδίασης λογισμικού, Μοντέλα εξισώσεων, Μοντελοποίηση και εξομοίωση, Στατιστική επικύρωση, Κλάσεις και αντικείμενα, Επισκέπτης, Σύνθεση.

# Table of Contents

1	Introduction.....	1
1.1	Field of Application .....	1
1.2	Necessity of the Introduced Theory .....	2
1.2.1	Need for Early Maintainability Assessment of Design Alternatives .....	2
1.2.2	Need for Formal Models to Support Decision-Making of General Problems .....	2
1.2.3	Motivation Example of General Design Problem .....	2
1.2.4	Need for Decision-Making Reliability and Validation of Formal Models .....	4
1.3	Main Research Goals and Contribution .....	5
1.4	Brief Review of Related Work (Existing Approaches) .....	6
1.5	Brief Presentation of Proposed Theory.....	7
1.5.1	Modeling Theory and Method.....	7
1.5.2	Structural Maintenance Cost (SMC) Metric.....	7
1.5.3	Deriving Formal Comparison Models.....	8
1.5.4	Advantages of Modeling Method & Formal Comparison Models.....	9
1.5.5	Simulating Software Evolution Towards Statistical Validation.....	11
1.5.6	Advantages of Simulation Model .....	12
1.6	Contribution and Innovations of the Study .....	13
1.6.1	Applicability of Formal Comparison Models .....	13
1.6.2	Adaptability of Modeling Theory .....	13
1.6.3	Computer-Aided Derivation of Formal Models .....	13
1.6.4	Simulation Model & Statistical Validation .....	13
1.6.5	Alternate Computer-Aided Model Implementations .....	14
1.6.6	Design Decisions Under Uncertainty & Horizon Analysis.....	14
1.6.7	Designers and Developers (Practitioners) Perspective .....	14
1.6.8	Researchers and Accademia Perspective .....	15
1.6.9	Project and Quality Managers Perspective.....	15
1.6.10	Novelty and Critical Evidence .....	15
1.7	Thesis Organization .....	16
2	Related Work.....	18
2.1	Analysis of Design Patterns .....	18
2.2	Visitor Design Pattern vs Inheritance-Based Implementation .....	18
2.3	Effort Estimation During Software Evolution and Maintenance Process.....	20
2.4	Assessing Maintainability During Early Design Stage.....	21
3	Quantitative Analysis of Design Problems .....	22
3.1	Chapter Overview.....	22
3.2	General Principles.....	23
3.2.1	General Architectural Design Principles .....	23
3.2.2	Object-Oriented Design Principles .....	24
3.3	General Decision – Design Problem.....	25
3.3.1	Recursive Hierarchies of Part-Whole Representations .....	25
3.3.2	Engaged Design Patterns .....	26
3.4	Software Quality Measures .....	30
3.4.1	Quality Measures.....	30
3.4.2	Expressing Software Maintainability .....	31
3.4.3	Deriving Maintainability Effort.....	33
3.4.4	Considerations upon Other Quality Characteristics and Properties .....	35
3.4.5	Characteristics and Criteria of Major Maintenance Scenarios.....	36
3.5	Analysis of Method.....	38
3.5.1	Deriving Problem’s Characteristics and Attributes .....	38
3.5.2	Asymptotic Evaluation of Structural Maintenance Cost .....	39



3.5.3	Merging Structural Maintenance Cost .....	41
3.5.4	Combining Maintenance Cost .....	42
3.5.5	Summarizing Structural Maintenance Cost .....	43
3.5.6	Maintenance Process .....	43
3.5.7	Combined Analysis .....	44
3.6	Quantitative Analysis .....	45
3.6.1	Basic Analysis .....	45
3.6.2	Combined Analysis .....	46
3.7	Progressive Analysis .....	48
3.7.1	Deriving Progressive Maintenance Cost .....	48
3.7.2	Progressive Maintenance Cost Computation .....	49
3.7.3	Reverse Analysis (Verification) .....	51
3.7.4	Graph of Progressive Maintenance Cost .....	52
3.7.5	Integrated Maintenance Cost .....	53
3.8	Application of the Proposed Model .....	54
3.8.1	Summarizing Maintenance Cost of the Model .....	54
3.8.2	Classification and Application Flowchart of Proposed Model .....	54
3.8.3	Application Examples .....	56
3.8.4	Alternate Maintenance Measures .....	58
3.8.5	Comparison to Relevant Metrics .....	59
3.8.6	Discussion .....	60
3.9	Methodology Determination .....	60
3.9.1	Methodology Description .....	60
3.9.2	Example of Weighted Effort Measurement .....	62
3.9.3	Further Discussion .....	63
3.10	Conclusions .....	64
3.10.1	General Requirements and Limitations .....	64
3.10.2	Extensions and Further Research .....	65
3.10.3	Overall Assessment .....	66
4	Modeling Software Evolution .....	67
4.1	Chapter Overview .....	67
4.2	Background of General Decision Problem .....	68
4.3	Modeling Approach .....	71
4.3.1	Designs for Change Principle .....	71
4.3.2	Corresponding Architectural Design Principles .....	71
4.3.3	Characteristics of SMC Effort Metric .....	73
4.3.4	Fundamental SMC Effort Metric Derivation .....	73
4.3.5	Software Expansion Concept .....	75
4.3.6	Analysis of System’s Size Change Rate .....	76
4.3.7	Structural Evolution through Change Rates .....	77
4.3.8	Differential Analysis and Model Derivation .....	78
4.4	Generalizing and Formalizing Modeling Method .....	81
4.4.1	Modeling Framework .....	81
4.4.2	Framework Implementation on General Problems using MATLAB® .....	83
4.4.3	Graph Generation and Decision-Making for Specific Practical Systems using MATLAB® .....	84
4.4.4	Formal Model Application in Examples of Practical Specific Problems .....	84
4.4.5	Justification of Formal Model’s Derivation Cost .....	85
4.5	Validation Evidence .....	87
4.6	Conclusions .....	87
4.6.1	General Requirements and Limitations .....	87
4.6.2	Extensions and Further Research .....	88
4.6.3	Overall Assessment .....	88

5	Extended General Design Problems .....	90
5.1	Chapter Overview .....	90
5.2	Attaching Decorator Design Pattern.....	90
5.2.1	Problem Description .....	90
5.2.2	Derivation of Effort Measurements and Formal Models.....	91
5.2.3	Formal Model Application in Examples of Practical Specific Problems....	92
5.2.4	Average Rate of Gained or Avoided Wasted Effort .....	93
5.3	Attaching Mediator and Observer Design Patterns.....	94
5.3.1	Problem Description .....	94
5.3.2	Derivation of Effort Measurement and Formal Models .....	95
5.3.3	Formal Model Application in Examples of Practical Specific Problems....	97
5.3.4	Average Rate of Gained or Avoided Wasted Effort .....	97
5.4	Attaching Abstract Factory and Prototype Design Patterns .....	98
5.4.1	Problem Description .....	98
5.4.2	Derivation of Effort Measurement and Formal Models .....	100
5.4.3	Formal Model Application in Examples of Practical Specific Problems...	101
5.4.4	Average Rate of Gained or Avoided Wasted Effort .....	102
5.5	Average Rate of Gained or Avoided Wasted Effort of CVP vs CIBI .....	102
5.6	Summarizing the Contribution of Modeling Method.....	103
5.7	Conclusions .....	104
5.7.1	General Requirements and Limitations .....	104
5.7.2	Extensions and Further Research.....	104
5.7.3	Overall Assessment.....	104
6	Simulation of Software Evolution.....	106
6.1	Chapter Overview.....	106
6.2	Background .....	108
6.2.1	Example of Practical General Problem.....	108
6.2.2	Characteristics of SMC Effort Metric.....	110
6.2.3	Software Expansion Concept and Formal Models Derivation .....	114
6.2.4	Formal Models Application in Specific Instances of the General Problem 115	
6.3	Method Evaluation and Validity Concerns.....	116
6.3.1	Effort Measurement Validity Concerns.....	116
6.3.2	Modeling Method Validity Concerns.....	116
6.4	Method Validation Through Simulation .....	117
6.4.1	Scoping and Planning.....	117
6.4.2	Hypothesis Formulation .....	117
6.4.3	Variables and Treatments Selection .....	117
6.4.4	Selection of Sample (Subjects and Objects).....	118
6.4.5	Conceptual Analysis of Validation Process.....	119
6.4.6	Experiment Design .....	123
6.4.7	Instrumentation .....	125
6.4.8	Conducting and Data Validation.....	135
6.4.9	Calibration of Model’s Stochastic Behavior .....	137
6.5	Results & Inferences.....	150
6.5.1	Analysis and Interpretation .....	150
6.5.2	Inference Extraction.....	152
6.5.3	Pattern Exploration of Decision Errors.....	161
6.5.4	Uncertainty Considerations.....	164
6.5.5	Statistical Evaluation per Sample Instance .....	167
6.5.6	Summarizing Results and Inferences.....	170
6.6	Conclusions .....	170
6.6.1	General Requirements and Limitations.....	170

6.6.2	Extensions and Further Research.....	171
6.6.3	Overall Assessment.....	172
7	Alternate Use of Formal Comparison Models .....	173
7.1	Chapter Overview.....	173
7.2	Decisions Under Uncertainty .....	174
7.2.1	Transforming Formal Models to Support Decision-Making Under Uncertainty .....	174
7.2.2	Example of Decision-Making Under Uncertainty.....	175
7.2.3	Additional Decision-Criteria to Decision-Making Process.....	177
7.3	Horizon Analysis .....	178
7.3.1	Separating Maintenance Process to Sub-Periods .....	178
7.3.2	Example of Decision-Making Supported by Horizon Analysis .....	179
7.4	Alternate Computer-Aided Implementation of Formal Models .....	180
7.4.1	Discrete Implementation of Models .....	181
7.4.2	Continuous Implementation of Formal Models .....	182
7.4.3	Discrete Implementation of Simulation Model.....	184
7.4.4	Comparison of Implementations’ Outcomes.....	186
7.5	Seeing Software Design as Investment .....	190
7.5.1	Cost of Missed Opportunities .....	190
7.5.2	Accounting Perspective .....	190
7.6	Conclusions .....	191
8	Conclusions and Future Work .....	192
8.1	Contribution of Dissertation .....	192
8.2	Future Work.....	193
	Appendix A: Mat Lab Modeling Framework .....	195
	Appendix B: Sample Data of General Problem.....	199
	Appendix C: Variability of Sample Instances.....	204
	Appendix D: Graphs of Statistical Evaluation.....	231
	Publications.....	275
	Bibliography.....	276

# List of Figures

Figure 1-1: The decision-making between design alternatives in a typical soft-ware lifecycle, and the negative consequences in case of wrong selection.....	1
Figure 1-2: Typical timeline for design pattern selection process during software design stage referring to the CVP vs CIBI design problem .....	3
Figure 1-3: Decision-making between design alternatives supported by the introduced modeling method and derived formal models.....	5
Figure 1-4: Overall presentation of the introduced modeling method, derived formal/simulation models, and validation process.....	8
Figure 1-5: Conceptual class-diagram and relevant effort metrics of CIBI and CVP design alternatives.....	10
Figure 1-6: Results of the application of the Formal comparison Model on the practical example of Interpreter as a specific instance of the CVP vs. CIBI general design problem. ....	11
Figure 1-7: Dependence graph of chapters. ....	17
Figure 3-1: Typical flow diagram of architectural design activities. ....	24
Figure 3-2: Example of class and object diagrams for typical hierarchies of objects. ....	27
Figure 3-3: Example of class diagram for a typical structure based on Composite design pattern (CP) and inheritance-based implementation (IBI).....	28
Figure 3-4: Typical code example based on Composite design pattern (CP) and inheritance-based implementation (IBI).....	28
Figure 3-5: Example of class diagram for a typical structure based on Composite and Visitor design patterns (CP, VP).....	29
Figure 3-6: Typical code example based on Composite and Visitor design patterns (CP, VP). ....	30
Figure 3-7: Conceptual diagram of interrelation of code properties regarding software maintainability assessment. ....	32
Figure 3-8: Tradeoffs between various types of arriving events during maintenance. ....	37
Figure 3-9: Design attributes of typical CIBI and CVP implementations.....	39
Figure 3-10: Typical code example after implementation of one new element and one new operation scenarios for CIBI and CVP implementation alternatives. ....	40
Figure 3-11: Typical conceptual flowchart of software maintenance process.....	44
Figure 3-12: Maintenance process flowchart for CVP and CIBI implementations. ....	44
Figure 3-13: Graph of maintenance cost of modifications on a Composition for a single future addition referred to the inheritance-based implementation (IBI) and Visitor design pattern (VP).....	46
Figure 3-14: Graphs of maintenance cost of modifications on a Composition for a single future addition, related to $\mu$ and $p_{nE}$ factors, referred to the Visitor design pattern (VP) and Inheritance based implementation (IBI). ....	46
Figure 3-15: Graph of asymptotic cost differentiation $C_{diff}(3)$ for modifications on a Composition for a single future addition, related to $\mu$ and $p_{nE}$ factors, referred to the Visitor design pattern (VP) and inheritance-based implementation (IBI). ....	47
Figure 3-16: Graph of balance cases (equal maintenance cost) for Visitor design pattern (VP) vs Inheritance based implementation (IBI).....	47
Figure 3-17: Computation of progressive maintenance cost for $\lambda$ future additions/modifications on a Composite using Visitor design pattern (CVP). ....	49
Figure 3-18: Graph of progressive maintenance cost differentiation $pc_{em(dist)}(p_{nE})$ for modifications on a Composition for $\lambda$ future additions, related to the $\mu$ and $p_{nE}$ factors, referred to the Visitor design pattern (VP) and Inheritance based implementation (IBI). ....	53

Figure 3-19: Computational pattern of Structural Maintenance Cost.....	54
Figure 3-20: Classification diagram of the proposed model. ....	55
Figure 3-21: Application flowchart on the use of Composite, Visitor, Iterator design patterns and inheritance-based implementation. ....	56
Figure 3-22: Graph of balance cases (equal maintenance cost) for CVP vs CIBI. ....	57
Figure 3-23: Example: computation of progressive asymptotic cost for inheritance-based implementation (IBI) and Visitor design pattern (VP). ....	58
Figure 3-24: Diagram of methods comparison through progressive analysis of multiple measures. ....	59
Figure 3-25: Results of metrics comparison. ....	59
Figure 3-26: Graph of balance cases (equal maintenance cost) for CVP vs CIBI based on w factor. ....	63
Figure 4-1: Conceptual UML class-diagram of CIBI (Inheritance-Based Implementation inside a Composition) and CVP (Visitor design pattern over Composition's pattern) design combinations. ....	70
Figure 4-2: Conceptual representation of the architectural design principles connected to the proposed theory and Modeling Method ....	72
Figure 4-3: Consequence flow (logical model) during impact analysis for changes on the Visitor over Composite design combination (CVP). ....	74
Figure 4-4: Consequence flow (logical model) during impact analysis for changes on the Inheritance-Based Implementation into Composite design combination (CIBI). ....	75
Figure 4-5: Abstract representation of software dynamic expansion during the maintenance process. ....	77
Figure 4-6: Results of the application of the Formal comparison Model on the practical examples of Interpreter, and Graphic User Inter-face (GUI), specific problems as instances of CVP vs. CIBI general problem. ....	85
Figure 5-1: Conceptual UML class-diagram of CIBI-DP vs CVP-DP design combinations. ....	91
Figure 5-2: Results of the application of the Formal comparison Model on the practical examples of Graphic User Interface (GUI) specific problems as instances of CVP-DP vs. CIBI-DP general problem. ....	93
Figure 5-3: Box plots of frequency distributions of sample's instances, concerning all the parameters of CIBI-DP vs. CVP-DP general problem. ....	93
Figure 5-4: Conceptual UML class-diagram of CIBI-MP vs CVP-MP design combinations. ....	94
Figure 5-5: Conceptual UML class-diagram of CIBI-OP vs CVP-OP design combinations. ....	95
Figure 5-6: Results of the application of the Formal comparison Model on the practical examples of Graphic User Interface (GUI) specific problems as instances of CVP-MP vs. CIBI-MP vs. CVP-OP vs. CIBI-OP general problem. ....	97
Figure 5-7: Box plots of frequency distributions of sample's instances, concerning all the parameters of CIBI-DP vs. CVP-DP general problem. ....	98
Figure 5-8: Conceptual UML class-diagram of CIBI-AF vs CVP-AF design combinations. ....	99
Figure 5-9: Conceptual UML class-diagram of CIBI-PT vs CVP-PT design combinations, including basic design attributes (N,M,F) and analysis of the affected code units per major maintenance scenario. ....	100
Figure 5-10: Results of the application of Formal comparison Model on the practical examples of Graphic User Interface (GUI) specific problems as instances of CVP-AF vs. CIBI-AF vs. CVP-PT vs. CIBI-PT general problem. ....	101

Figure 5-11: Box plots of frequency distributions of sample’s instances, concerning all the parameters of CVP-AF vs. CIBI-AF vs. CVP-PT vs. CIBI-PT general problem .....	102
Figure 5-12: Box plots of frequency distributions of sample’s instances, concerning all the parameters of CIBI vs. CVP general problem .....	103
Figure 6-1: Representation of study’s goals, contribution, and limitations .....	107
Figure 6-2: Conceptual UML classes diagram of CIBI and CVP design combinations, including basic design attributes (N,M) and analysis of the affected code units per major maintenance scenario. ....	109
Figure 6-3: Consequence flow (logical model) during impact analysis for changes on CVP design combination, including alternate scenarios.....	112
Figure 6-4: Consequence flow (logical model) during impact analysis for changes on CIBI design combination, including alternate scenarios.....	112
Figure 6-5: Typical code example after the application of one new element and one new operation scenarios for CIBI and CVP design alternatives .....	114
Figure 6-6: Diagram of formal & simulated comparison models applied on the practical example of Interpreter (N:40, M:10, $p_{ne}:0.5$ , $e_{xp}:1$ ) as an instance of CVP vs. CIBI general problem .....	116
Figure 6-7: Experimental setup visualization .....	119
Figure 6-8: Visualization of research strategy, focusing on reality and theoretical aspects. ....	120
Figure 6-9: Visualization of the experiment context, focusing on contradistinction among theoretical and observational aspects .....	122
Figure 6-10: Software entropy concept in relation to code’s aging.....	129
Figure 6-11: Abstract (indicative) representation of the Simulation Model implementation as functional model.....	131
Figure 6-12: Class diagram of an indicative modular representation of the simulation model.....	132
Figure 6-13: Object diagram of an indicative run-time representation of the modular simulation model .....	133
Figure 6-14: Example of intermediate results/outcomes of DSL implementation of the Simulation Model for CVP design combination .....	134
Figure 6-15: Example of GUI of the DSL Implementation of Simulation Model .....	135
Figure 6-16: Frequency distributions of Simulation Model’s CVP, CIBI total effort assessments, and their differences, for all (1000) object-subject instances of the selected sample, where $\lambda=200$ , relevant to the 7 <sup>th</sup> simulation state in Table 6-4.....	136
Figure 6-17: Frequency distributions of Formal Model’s CVP, CIBI total effort assessments, and their differences, for all (1000) object instances of the selected sample, where $\lambda=200$ , relevant to the 7 <sup>th</sup> simulation state in Table 6-4. ....	136
Figure 6-18: Indicative frequency distributions and scatter diagrams of Simulation Model’s outcomes (CVP, CIBI, CVP-CIBI) for 100 repeated simulations in a single object-subject instance (N:40, M:10, $p_{ne}:0.5$ , $e_{xp}:1.0$ ), where $\lambda=200$ relevant to the 7 <sup>th</sup> simulation state in Table 6-4. ....	137
Figure 6-19: Multi-resolution modeling approach towards calibration of Simulation Model .....	138
Figure 6-20: Frequency distributions and time series of Simulation Model’s overall stochastic factor and intermediate outcomes (CVP, CIBI) of an indicative single object-subject instance (N:40, M:10, $p_{ne}:0.5$ , $e_{xp}:1.0$ ), where $\lambda=[1, \dots, 200]$ , relevant to the 7 <sup>th</sup> simulation state in Table 6-4. ....	139
Figure 6-21: Consistency criterion of simulation model’s calibration.....	140

Figure 6-22: Overall assessment of the statistical parameters of the $\Sigma Y_i^{\text{repeated}}$ and $Y_i$ variables concerning all the sample instances. ....	143
Figure 6-23: Time series analysis (ARIMA) of total CVP effort assessments .....	146
Figure 6-24: Time series analysis (ARIMA) of (single differentiated) intermediate CVP effort assessments.....	147
Figure 6-25: Time series analysis (ARIMA) of (double differentiated) intermediate CVP effort assessments (simulation of sample instance N. 002) .....	148
Figure 6-26: ACF and PACF diagrams of (double differentiated) effort time series .....	149
Figure 6-27: Time series analysis ARIMA(0,2,1) of (double differentiated) intermediate CVP effort assessments (simulation of sample instance N. 002) .....	150
Figure 6-28: Scatter diagrams of external correlation among formal and simulated values of total effort (CVP, CIBI, CVP-CIBI), concerning all (1000) sample instances, of the 7 <sup>th</sup> fully stochastic simulation state in Table 6-4. ....	151
Figure 6-29: Overall control diagrams (left):coefficient of correlation, (right):hypothesis T-test, (mid):average error rate & critical error rate, concerning 1000 sample instances, of the 7 <sup>th</sup> fully stochastic simulation state in Table 6-4.....	152
Figure 6-30: Scatter diagrams of external correlation among formal and simulated values of total effort (CVP, CIBI, CVP-CIBI), concerning all (1000) sample instances, of the 1 <sup>st</sup> simulation state in Table 6-4. ....	153
Figure 6-31: Overall control diagrams (left):coefficient of correlation, (right):hypothesis T-test, (mid):average error rate & critical error rate, concerning 1000 sample instances, of the 1 <sup>st</sup> simulation state in Table 6-4. ....	154
Figure 6-32: Scatter diagrams of external correlation among formal and simulated values of total effort (CVP, CIBI, CVP-CIBI), concerning all (1000) sample instances, of the 2 <sup>nd</sup> simulation state in Table 6-4. ....	154
Figure 6-33: Overall control diagrams (left):coefficient of correlation, (right):hypothesis T-test, (mid):average error rate & critical error rate, concerning 1000 sample instances, of the 2 <sup>nd</sup> simulation state in Table 6-4. ....	155
Figure 6-34: Scatter diagrams of external correlation among formal and simulated values of total effort (CVP, CIBI, CVP-CIBI), concerning all (1000) sample instances, of the 3 <sup>rd</sup> simulation state in Table 6-4. ....	155
Figure 6-35: Overall control diagrams (left):coefficient of correlation, (right):hypothesis T-test, (mid):average error rate & critical error rate, concerning 1000 sample instances, of the 3 <sup>rd</sup> simulation state in Table 6-4.....	156
Figure 6-36: Scatter diagrams of external correlation among formal and simulated values of total effort (CVP, CIBI, CVP-CIBI), concerning all (1000) sample instances, of the 4 <sup>th</sup> simulation state in Table 6-4. ....	157
Figure 6-37: Overall control diagrams (left):coefficient of correlation, (right):hypothesis T-test, (mid):average error rate & critical error rate, concerning 1000 sample instances, of the 4 <sup>th</sup> simulation state in Table 6-4.....	157
Figure 6-38: Scatter diagrams of external correlation among formal and simulated values of total effort (CVP, CIBI, CVP-CIBI), concerning all (1000) sample instances, of the 5 <sup>th</sup> simulation state in Table 6-4. ....	158
Figure 6-39: Overall control diagrams (left):coefficient of correlation, (right):hypothesis T-test, (mid):average error rate & critical error rate, concerning 1000 sample instances, of the 5 <sup>th</sup> simulation state in Table 6-4.....	159
Figure 6-40: Scatter diagrams of external correlation among formal and simulated values of total effort (CVP, CIBI, CVP-CIBI), concerning all (1000) sample instances, of the 6 <sup>th</sup> simulation state in Table 6-4. ....	159
Figure 6-41: Overall control diagrams (left):coefficient of correlation, (right):hypothesis T-test, (mid):average error rate & critical error rate, concerning 1000 sample instances, of the 6 <sup>th</sup> simulation state in Table 6-4.....	160
Figure 6-42: Experimental error rate assessment. ....	162

Figure 6-43: Design space representation through 1000 sample instances of CVP vs. CIBI general problem, including long-term error and critical error rate assessment ( $\lambda=200$ , 7 <sup>th</sup> simulation state in Table 6-4). .....	164
Figure 6-44: Frequency distributions (Box Plots) of Simulation Model’s outcomes (CVP, CIBI, CVP-CIBI) for 100 repeated simulations in a single object-subject instance (N:40, M:10, $p_{ne}:0.5$ , $e_{xp}:1.0$ ), of the 7 <sup>th</sup> fully stochastic simulation state in Table 6-4. ....	165
Figure 6-45: Levels of uncertainty distributions of Simulation Model’s outcomes (CVP, CIBI) for 100 repeated simulations in a single object-subject instance (N:40, M:10, $p_{ne}:0.5$ , $e_{xp}:1.0$ ), of the 7 <sup>th</sup> fully stochastic simulation state in Table 6-4. ....	166
Figure 6-46: Statistical evaluation of GUI implementation based on simulation outcomes .....	167
Figure 7-1: Graph of total effort per design alternative (CVP vs. CIBI) for different horizons (sub-periods) of the software lifecycle, including preliminary development stage, referred to the example of Interpreter requirements with initial design attributes during maintenance N=40 and M=10.....	180
Figure 7-2: Computer-Aided implementation of the discrete Formal Model for the general decision problem CIBI vs CVP, using VENSIM® tool.....	181
Figure 7-3: Computer-Aided implementation of the continuous Formal Model for the general decision problem CIBI vs CVP, using VENSIM® tool.....	183
Figure 7-4: Computer-Aided implementation of the event-driven Simulation Model for the general decision problem CIBI vs CVP, using VENSIM® tool.....	184
Figure 7-5: Difference of total effort among CVP and CIBI design alternatives for DFM, FM, and SM models, for the GUI implementation (N=15, M=14, $p_{ne}=0.7$ ) .....	187
Figure 7-6: Total effort of CVP (left) and CIBI (right) design alternatives for DFM, FM, and SM models, for the GUI implementation (N=15, M=14, $p_{ne}=0.7$ ).....	187
Figure 7-7: Analysis of Stochastic Factor components of SM model, for the GUI implementation (N=15, M=14, $p_{ne}=0.7$ ).....	188
Figure 7-8: Effort change rate of CVP (left) and CIBI (right) design alternatives for DFM, FM, and SM models, for the GUI implementation (N=15, M=14, $p_{ne}=0.7$ ) .....	188
Figure 7-9: Design attributes of Elements N (left) and Operations M (right) for DFM, FM, and SM models, for the GUI implementation (N=15, M=14, $p_{ne}=0.7$ ) .....	189
Figure 7-10: New element scenario probability ( $p_{ne}$ ) and shifted scenario probability of SM model, for the GUI implementation (N=15, M=14, $p_{ne}=0.7$ ).....	190
Figure 0-1: Sample instance N. 001 (N=161, M=60, $p_{ne}=0.67$ , $p_{np}=0.33$ ) .....	231
Figure 0-2: Sample instance N. 004 (N=56, M=123, $p_{ne}=0.09$ , $p_{np}=0.91$ ) .....	232
Figure 0-3: Sample instance N. 006 (N=101, M=12, $p_{ne}=0.90$ , $p_{np}=0.10$ ) .....	233
Figure 0-4: Sample instance N. 007 (N=31, M=7, $p_{ne}=0.15$ , $p_{np}=0.85$ ).....	234
Figure 0-5: Sample instance N. 008 (N=134, M=112, $p_{ne}=0.15$ , $p_{np}=0.85$ ).....	235
Figure 0-6: Sample instance N. 009 (N=104, M=38, $p_{ne}=0.08$ , $p_{np}=0.92$ ) .....	236
Figure 0-7: Sample instance N. 010 (N=81, M=6, $p_{ne}=0.70$ , $p_{np}=0.30$ ).....	237
Figure 0-8: Sample instance N. 011 (N=90, M=58, $p_{ne}=0.72$ , $p_{np}=0.28$ ).....	238
Figure 0-9: Sample instance N. 012 (N=29, M=102, $p_{ne}=0.34$ , $p_{np}=0.66$ ) .....	239
Figure 0-10: Sample instance N. 014 (N=188, M=133, $p_{ne}=0.14$ , $p_{np}=0.86$ ).....	240
Figure 0-11: Sample instance N. 016 (N=155, M=91, $p_{ne}=0.22$ , $p_{np}=0.78$ ).....	241
Figure 0-12: Sample instance N. 017 (N=21, M=5, $p_{ne}=0.26$ , $p_{np}=0.74$ ).....	242
Figure 0-13: Sample instance N. 018 (N=139, M=132, $p_{ne}=0.93$ , $p_{np}=0.07$ ).....	243
Figure 0-14: Sample instance N. 019 (N=89, M=126, $p_{ne}=0.30$ , $p_{np}=0.70$ ).....	244
Figure 0-15: Sample instance N. 020 (N=156, M=138, $p_{ne}=0.68$ , $p_{np}=0.32$ ).....	245
Figure 0-16: Sample instance N. 021 (N=70, M=99, $p_{ne}=0.57$ , $p_{np}=0.43$ ) .....	246



Figure 0-17: Sample instance N. 022 (N=160, M=13, $p_{nE}=0.90$ , $p_{nP}=0.10$ ).....	247
Figure 0-18: Sample instance N. 024 (N=46, M=89, $p_{nE}=0.63$ , $p_{nP}=0.37$ ) .....	248
Figure 0-19: Sample instance N. 025 (N=156, M=55, $p_{nE}=0.30$ , $p_{nP}=0.70$ ).....	249
Figure 0-20: Sample instance N. 026 (N=137, M=78, $p_{nE}=0.74$ , $p_{nP}=0.26$ ).....	250
Figure 0-21: Sample instance N. 027 (N=106, M=82, $p_{nE}=0.18$ , $p_{nP}=0.82$ ).....	251
Figure 0-22: Sample instance N. 029 (N=141, M=24, $p_{nE}=0.94$ , $p_{nP}=0.06$ ).....	252
Figure 0-23: Sample instance N. 031 (N=26, M=33, $p_{nE}=0.85$ , $p_{nP}=0.15$ ) .....	253
Figure 0-24: Sample instance N. 032 (N=154, M=99, $p_{nE}=0.46$ , $p_{nP}=0.54$ ).....	254
Figure 0-25: Sample instance N. 033 (N=22, M=68, $p_{nE}=0.85$ , $p_{nP}=0.15$ ) .....	255
Figure 0-26: Sample instance N. 034 (N=130, M=58, $p_{nE}=0.50$ , $p_{nP}=0.50$ ).....	256
Figure 0-27: Sample instance N. 035 (N=155, M=9, $p_{nE}=0.35$ , $p_{nP}=0.65$ ) .....	257
Figure 0-28: Sample instance N. 036 (N=182, M=111, $p_{nE}=0.48$ , $p_{nP}=0.52$ ).....	258
Figure 0-29: Sample instance N. 037 (N=196, M=41, $p_{nE}=0.87$ , $p_{nP}=0.13$ ).....	259
Figure 0-30: Sample instance N. 038 (N=166, M=103, $p_{nE}=0.46$ , $p_{nP}=0.54$ ).....	260
Figure 0-31: Sample instance N. 039 (N=66, M=139, $p_{nE}=0.40$ , $p_{nP}=0.60$ ).....	261
Figure 0-32: Sample instance N. 040 (N=139, M=134, $p_{nE}=0.61$ , $p_{nP}=0.39$ ).....	262
Figure 0-33: Sample instance N. 041 (N=48, M=89, $p_{nE}=0.20$ , $p_{nP}=0.80$ ) .....	263
Figure 0-34: Sample instance N. 042 (N=129, M=114, $p_{nE}=0.37$ , $p_{nP}=0.63$ ).....	264
Figure 0-35: Sample instance N. 043 (N=182, M=119, $p_{nE}=0.62$ , $p_{nP}=0.38$ ).....	265
Figure 0-36: Sample instance N. 045 (N=166, M=51, $p_{nE}=0.10$ , $p_{nP}=0.90$ ).....	266
Figure 0-37: Sample instance N. 046 (N=121, M=133, $p_{nE}=0.89$ , $p_{nP}=0.11$ ).....	267
Figure 0-38: Sample instance N. 047 (N=154, M=108, $p_{nE}=0.90$ , $p_{nP}=0.10$ ).....	268
Figure 0-39: Sample instance N. 048 (N=35, M=133, $p_{nE}=0.61$ , $p_{nP}=0.39$ ).....	269
Figure 0-40: Sample instance N. 049 (N=66, M=90, $p_{nE}=0.64$ , $p_{nP}=0.36$ ) .....	270
Figure 0-41: Sample instance N. 050 (N=43, M=76, $p_{nE}=0.28$ , $p_{nP}=0.72$ ) .....	271
Figure 0-42: Sample instance N. 051 (N=39, M=138, $p_{nE}=0.35$ , $p_{nP}=0.65$ ).....	272
Figure 0-43: Sample instance N. 052 (N=157, M=118, $p_{nE}=0.50$ , $p_{nP}=0.50$ ).....	273
Figure 0-44: Sample instance N. 053 (N=160, M=139, $p_{nE}=0.92$ , $p_{nP}=0.08$ ).....	274

*figures 0.\* refer to appendix content*

## List of Tables

Table 1-1: Examples of Software specifications for part-whole representations .....	4
Table 3-1: Maintenance Cost Terminology and Notation.....	35
Table 3-2: Maintenance Cost Terminology and Notation.....	39
Table 3-3: Asymptotic Evaluation of Structural Maintenance Cost on Inheritance-Based Implementation and Visitor Design Pattern .....	43
Table 3-4: Characteristics and attributes of individual problem descriptions.....	57
Table 3-5: Correlation of Model’s Measures with Related Models .....	58
Table 3-6: Methodology for deriving comparison models .....	60
Table 4-1: Example of Interpreter Software Specifications for the General Problem of Part-Whole Representations.....	68
Table 4-2: Correspondence of Concepts, Components, and Terms to the General Decision Problem of Part-Whole Representations. ....	69
Table 4-3: Equations of Fundamental Effort Metrics of CIBI vs. CVP General Decision Problem.....	75
Table 4-4: Differential Analysis and Comparison Model Derivation for CVP vs. CIBI General Problem.....	80
Table 4-5: Terminology and Notation of Modeling Framework.....	81
Table 4-6: Modeling Framework of Differential Analysis and Comparison Model Generation .....	83
Table 4-7: Formula for Finding the Relation Between Initial Model’s Derivation Cost and Long-Term Benefits (Return).....	85
Table 5-1: Overall results for 1000 instances of design attributes and scenario’s probabilities per General Problem .....	103
Table 6-1: Design Characteristics and Model’s Notation of the General Problem of Part-Whole Representations.....	109
Table 6-2: Equations of Fundamental Effort Metrics of CIBI vs. CVP General Decision Problem.....	111
Table 6-3: Experiment Variables per Treatment for Formal and Simulation Models on CIBI vs. CVP General Problem .....	118
Table 6-4: Combinations (States) of Simulation Model’s Independent Variables and Switches .....	124
Table 6-5: Statistical parameters of real-world against simulated effort-based observations .....	142
Table 6-6: Statistical Parameters of ARIMA Analysis of Simulated Effort Assessments	149
Table 6-7: Overall Control Evidence for all Experiment Scenarios and Simulation Model States .....	152
Table 7-1: Horizon Analysis Data referred to the Interpreter Specific Instance of CVP vs. CIBI General Problem.....	179

## List of Listings

Listing 4-1: Loading CIBI vs CVP general problem to MatLab Modeling Framework ....	83
Listing 4-2: Loading Interpreter characteristics to MatLab Modeling Framework .....	84
Listing 5-1: Loading CIBI-DP vs. CVP-DP general problem to MatLab Modeling Framework .....	92
Listing 5-2: Loading CIBI-MP vs CVP-MP vs CIBI-OP vs CVP-OP general problem to MatLab Modeling Framework.....	96
Listing 6-1: Pseudocode of simulation model calibration based on multi-resolution modeling .....	144
Listing 7-1: Code documentation of CVP vs CIBI general problem in VENSIM implementation.....	182
Listing 7-2: Code documentation of CVP vs CIBI Formal Models in VENSIM implementation.....	183
Listing 7-3: Code documentation of Simulation Model for CVP vs CIBI general problem in VENSIM implementation.....	185

## List of Abbreviations & Acronyms

ACF	Auto Correlation Function
ADF	Augmented Dickey–Fuller (test)
AIC	Akaike’s Information Criterion
AOP	Aspect – Oriented Programming
AR	Auto Regressive
ARIMA	Auto Regressive Integrated Moving Average
ARMA	Auto Regressive Moving Average
CAD	Computer Aided Design
CIBI	Inheritance-Based Implementation over Composite design pattern
CK	Chidamber & Kemerer (object-oriented metrics)
CL	Confidence Level (of statistical test - parameter)
CP	Composite Design pattern
CVP	Visitor over Composite (design) Pattern
DSL	Domain Specific Language
GUI	Graphic User Interface
IBI	Inheritance-Based Implementation
IR	Intermediate Representation
KPSS	Kwiatkowski–Phillips–Schmidt–Shin (test)
LOC	Lines of Code
LR	Literature Review
MA	Moving Average
MOOD	Metrics for Object Oriented Design
PACF	Partial Auto Correlation Function
QAR	Quality Attribute Requirement
SLR	Systematic Literature Review
UML	Unified Modeling Language
VP	Visitor (design) Pattern

# 1 Introduction

## 1.1 Field of Application

Software architecture, as a process, is a set of design decisions (Bass, Clements, & Kazman, 2012). It is about (re)arranging structural elements and entities based on proper architectural tactics for creating design alternatives that satisfy the pursued quality attribute requirement such as maintainability or modifiability (Bass et al., 2012). Maintainability is one of the most important quality properties, mostly connected to the ease of future maintenance of software code (ISO/IEC 25010, 2011; ISO/IEC 25023, 2016; ISO/IEC/IEEE 24765, 2010), corresponding up to 75% of the overall cost of software projects (Bass et al., 2012; Glass, 2002; Sommerville, 2010). The decisions among design alternatives are often complex, crucial, and affect major quality properties of the software such as maintainability (Bosch & Bengtsson, 2001; R. Pressman, 2010; Sommerville, 2010; Srivastava, 2004).

A critical decision arises when at least two design alternatives with high impact and conflicting pro and cons regarding their maintainability perspective are competing to address the same requirements in different design ways. Furthermore, critical design decisions arise in different levels of analysis from the high-level architectural design of the entire system to the low-level of object-oriented design. Such decisions are usually made during the software architecture design stage before code development as illustrated in Figure 1-1. A possible incorrect selection of a less maintainable design alternative has serious negative impact concerning either: a) the considerable wasted effort and cost during maintenance, indicated by (a) arrow in Figure 1-1, or b) the costly setback in the design stage of software lifecycle which requires redesign and refactoring of the existing code, indicated by (b) arrow in Figure 1-1. Such design decisions are present regardless of the followed software development model and their impact usually echoes in different or repeated cycles of activities (e.g., waterfall, v-model, incremental, iterative, spiral, Agile iterations like Scrum, etc.). Early and critical design decisions have disproportionate weight simply because they influence and constrain so much of what follows, especially for significant, general, and frequently tackled design problems (Bass et al., 2012).

In software architecture and object-oriented software development, designers and developers face complex design problems (Sommerville, 2010). To handle these issues, designers use combinations of established design patterns, such as those introduced by

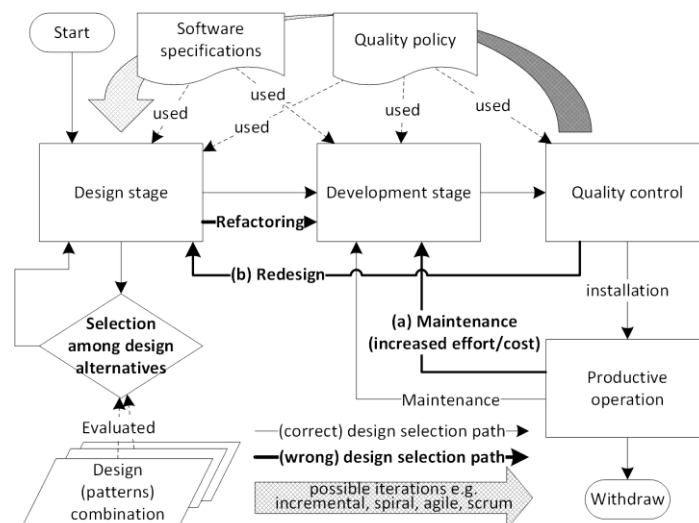


Figure 1-1: The decision-making between design alternatives in a typical software lifecycle, and the negative consequences in case of wrong selection.

Gamma et al. (Gamma, Helm, Johnson, & Vlissides, 1994). These design patterns are based on object-oriented models, and usually aim at solving common (frequently tackled) and significant design problems. Furthermore, many design (pattern) combinations for typical significant problems, such as recursive part-whole aggregations, affect significantly the majority of software quality properties (Bosch & Bengtsson, 2001; Srivastava, 2004). Efficient exploration, evaluation, and selection among design alternatives (i.e., patterns combinations) is crucial and has a direct impact on software quality (R. Pressman, 2010).

## 1.2 Necessity of the Introduced Theory

### 1.2.1 Need for Early Maintainability Assessment of Design Alternatives

Designers and developers often have to select between several object-oriented design alternatives which compete to solve significant, general, and frequently tackled design problems. These design decisions are often complex, crucial, and affect major quality attributes of the software such as modifiability and maintainability (Bass et al., 2012; R. Pressman, 2010). In general, software maintainability can be expressed through the required maintenance effort, and it is closely related to the code's complexity and size (Riaz, Mendes, & Tempero, 2009a). Therefore, early selection among design alternatives with regards to their maintainability profoundly affects future maintenance effort and cost as stressed in previous subsection. In practice, many designers and developers underestimate such critical decisions by using the most acknowledged design combination based on their intuition, experience, or instant judgment, thus usually leading to poor design decisions, accrued technical debt (Kruchten, Nord, & Ozkaya, 2012), high risk and cost (Williams & Carver, 2010), and increased maintenance effort (Xiao, Cai, Kazman, Mo, & Feng, 2016). The acknowledgment of designers about the negative consequences of a possible incorrect selection is the first step to confront the observed lack of architectural awareness (Paixao, Krinke, Han, Ragkhitwetsagul, & Harman, 2017). Consequently, early selection among competing design alternatives for significant design problems is a necessary step that must be made before code development, thus during the early design stage.

### 1.2.2 Need for Formal Models to Support Decision-Making of General Problems

Despite the observed lack of architectural awareness (Paixao et al., 2017), many software quality managers have a clear understanding of possible negative consequences of critical (early) design decisions, but they do not have a comprehensible theory or models to predict the impact and risk taken for each alternative selection. Whether a specific implementation (design alternative) is efficient with regards to maintainability is difficult to be assessed without a basis for comparison among design alternatives. The quality assessment of a design implementation is only meaningful when compared to other relevant design alternatives. The early evaluation and comparison with regards to maintainability of design alternatives are generally supported by predictions of the required maintenance effort. The design alternative with the minimum required maintenance effort is preferable. Effort predictions are usually provided by formal models applied during the design stage before code development. In this early design stage, effort prediction models that capture the structural evolution of the engaged design patterns, based on as less as possible distinct design attributes and parameters, are thus desired. Furthermore, software maintenance is a stochastic process heavily affected by many random and ambiguous factors. Hence, there is need for systematic approaches that derive (formal) deterministic models for maintenance effort estimation mainly for comparison purposes among design alternatives towards maintainability.

### 1.2.3 Motivation Example of General Design Problem

One example of significant, general, frequently tackled, and complex design problem is the part-whole representations and aggregations based on composite structures of objects (Sommerville, 2010). To handle such issues, designers and developers use combinations

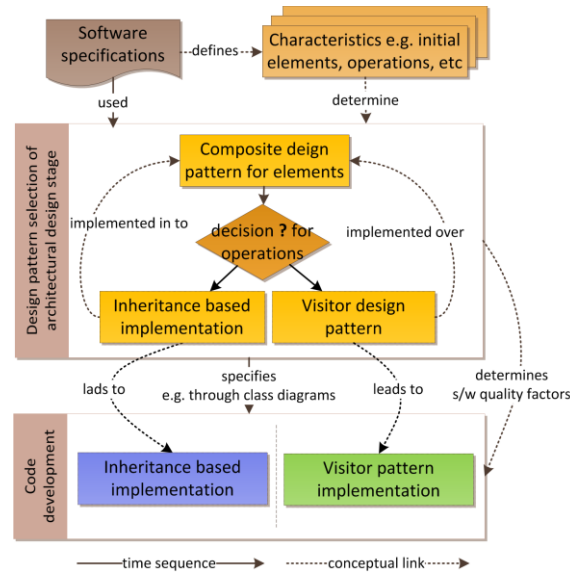


Figure 1-2: Typical timeline for design pattern selection process during software design stage referring to the CVP vs CIBI design problem

of established design patterns, which are based on object-oriented models, like the Composite and Visitor patterns (introduced in (Gamma et al., 1994)) or inheritance-based approaches. The proper choice among different implementation options is crucial and has a direct impact on the quality of software under development (R. Pressman, 2010) as stressed in previous subsections. More specifically, this general design problem is referred to the critical decision (selection) between Visitor design pattern (VP) and inheritance-based implementation (IBI) over a Composition design pattern (CP). IBI and VP are the two dominant implementation options (design alternatives) for a set of operations be applied on different types of elements of a CP object structure or composition (Gamma et al., 1994; Hills, Klint, Van Der Storm, & Vinju, 2011). More specifically, IBI is based on the standard inheritance property of CP class tree (hierarchy) where each type of operations is distributed as distinct methods inside each CP class/element. In VP all the methods for each type of operations are defined and placed in a single (visitor) class without changing the classes of the CP elements on which it operates. These patterns have opposing characteristics regarding their maintainability perspective. A wrong decision on selecting the proper pattern during software design stage can lead to substantial higher effort during maintenance. The choice between VP over CP (CVP) and IBI over CP (CIBI) is rather clear for a problem with a steady Composition structure or a steady set of operations (Gamma et al., 1994). However, when a problem with an extensible set of operations and an extensible Composition structure is addressed, the decision making becomes very complex. This decision is crucial and affects major quality factors of the software such as maintainability (R. Pressman, 2010). Such design decisions are usually made during software architecture design stage before code development as shown in Figure 1-2.

The usefulness and the necessity of the proposed work is highlighted through three simple examples of software specifications for the general design problem of part-whole representations (CVP vs CIBI) described above. Table 1-1 presents the descriptions of three practical systems (Compiler, Interpreter, GUI) as indicative examples or instances of the general and significant design problem of part-whole representations. Each instance has its own design characteristics (i.e., number of initial elements and operations) and individual specifications (i.e., likelihood of extensions). As concluded for the cases of Interpreter and GUI, when a problem with extensible sets of operations and Composition structure is addressed, the decision-making process becomes very complex.

Table 1-1: Examples of Software specifications for part-whole representations

Problem description	Number of initial distinct Elements of Composition structure	Number of initial Operations over composition's elements	Individual specifications
Compiler implementation for the standard C89 <sup>(1)</sup> high-level language	structure of 155 initial distinct types of the parse-tree nodes derived from C89 BNF grammar (proximally 85 tokens and 70 non-terminal symbols)	set of 20 initial distinct operations such as scope checking, type checking, dependency analysis, instruction selection-scheduling, code generation, etc.	- because C89 is a standard language, the structure set is rather unlikely to change during maintenance - operations could be extended during maintenance
	- an incorrect choice of inheritance-based implementation would have cause 155 new methods in 155 different classes for a single operation addition during maintenance - instead, the Visitor design pattern demands all 155 new methods to be placed in a single class, which needs far less maintenance effort - this is an easy choice due to structure stability based on the Visitor known advantages		
Interpreter implementation for a new custom (extendable) DSL language	structure of 40 initial distinct types of the parse-tree nodes derived from a custom BNF grammar such as terminal – nonterminal symbols, identifiers, etc.	set of 10 initial distinct operations type checking, code generation, executing, etc.	- because DSL is a custom and extendable language, both structure and operations could be extended during maintenance by equal probabilities
	- an incorrect choice of inheritance-based implementation would have cause 40 new methods in 40 different classes for a single operation addition and a new class with 10 new methods for a single element addition, during maintenance - instead, the Visitor design pattern demands all 40 new methods to be placed on a single class but 10 new methods to be placed in 10 different classes, which overall needs less maintenance effort - this is a difficult choice due to structure and operations expandability and there is no clear advantage		
GUI implementation for a simple graph designing tool	structure of 15 initial distinct types such as shapes, blocks, containers, layers, etc.	set of 14 initial distinct operations such as drawing, filling, resizing, moving, etc.	- both structure and operations could be extended during maintenance - structure is much likely to be extended instead of operations by 70%-30% probabilities
	- an incorrect choice of Visitor design pattern would have cause 15 new methods to be placed on a single class for a single operation addition but 14 new methods to be placed in 14 different classes for a single element addition, during maintenance - instead, the inheritance-based implementation demands 15 new methods in 15 different classes and a new class with 14 new methods, which overall needs more maintenance effort at the beginning but much less effort after some future addition due the individual maintenance probabilities - this is a very difficult choice due to structure and operations expandability, and there is no clear advantage for arbitrary maintenance probabilities		

Note: initial source code for Compiler and Interpreter will be generated by a parser tool such as Bison or ANTLR (Parr, 2013). Interpreter and GUI implementations are real cases descriptions. GUI implementation is direct without using standard frameworks such as .NET and WPF

<sup>(1)</sup> ANSI C Standard ANSI X3.159-1989 "Programming Language C"

To better understand the impact of future additions or modifications on their codes, software developers need a formal and mathematical approach for early exploring and evaluating relevant issues and comparing different design pattern combinations. Hence, the proposed method evaluates the effectiveness and maintainability degree of a design alternative by taking into consideration possible scenarios like future additions or modifications. A model that can deliver quantitative results based on specific design attributes of each general problem (e.g., initial structure size, number of initial operations and possibility of future extensions), is necessary. Using such models, software designers and developers can choose between VP and IBI combinations over Composite structures in an early stage, during software development phase. The introduced modeling method and the derived models address this necessity by providing deterministic results through equations based on specific design attributes and parameters of each general problem able to support early decision-making among design alternatives towards maintainability.

#### 1.2.4 Need for Decision-Making Reliability and Validation of Formal Models

In general, parametric formal models are mainly focused on maximizing the potential for being general over different instances of a given general problem. However, formal methods usually suffer from lack of realism of context and precision of measurements, as discussed in (Stol & Fitzgerald, 2018). Ideally, actual measurements and observations from case studies that maximize the potential for realism of context would be preferable for validation purposes. Nevertheless, in real life, finding identical actual systems with uniform



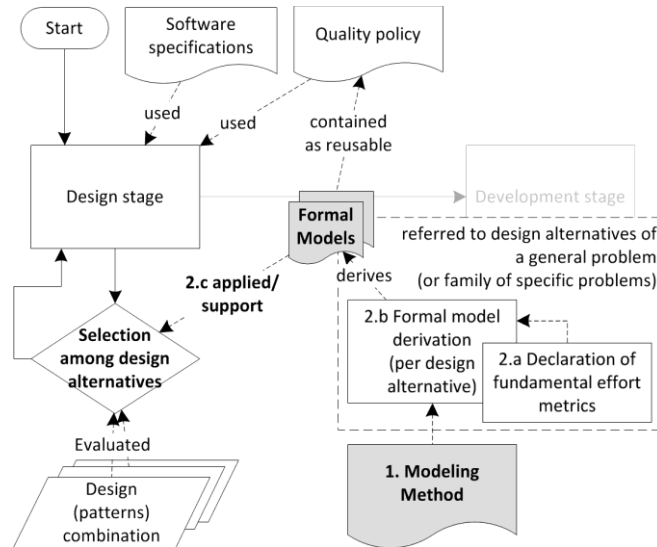


Figure 1-3: Decision-making between design alternatives supported by the introduced modeling method and derived formal models

design attributes, developed in different design variations is almost impossible. Additionally, the number of recorded observations is very limited per case study, using heterogeneous metrics, and unevenly conducted through literature. Thus, they are not statistically meaningful, heavily limiting the generalization of inferences, as pointed in (Langdon, Dolado, Sarro, & Harman, 2016; Shepperd & MacDonell, 2012). Moreover, developer-related aspects, such as experience level and learning rate are also ignored by these methods since they are heavily biased, as human-related, factors hard to be assessed and measured. Furthermore, there is a lack of evidence regarding the effectiveness of the prediction techniques and models of software maintainability (Riaz et al., 2009a; Shepperd & MacDonell, 2012). In addition, there is a confirmed need for further validation of maintainability prediction models (Riaz et al., 2009a), primarily through statistical techniques. Because of all these reasons, there is no easy way to determine the reliability of the models referred to possible incorrect design decisions in terms of maintainability. This is a standard concern with regards to validity since the attempt to validate the formal models based on a limited number and dissimilar case studies may increase realism of context while sacrificing generalizability which should be the models' primary goal.

Since the validation of formal models is critical to be made against real-world observations, there is a need for generating massive and homogenous observations properly classified in respect to all parameters (i.e., design attributes) of the general problem under study. Toward this direction, the simulation of software evolution that imitates the actual maintenance process and its highly uncertain and stochastic nature is a suitable option.

### 1.3 Main Research Goals and Contribution

Challenges arising from the motivation example in Table 1-1 imply that models that can deliver quantitative results based on attributes of each specific design problem (e.g., initial structure size, number of initial operations, and the possibility of future extensions) are needed. Using such models, software developers can choose between design alternatives in an early stage, during the software lifecycle, avoiding the possible negative consequences of incorrect selections and hence designing better software of higher quality.

Focusing on software design stage, the main contribution and goals of this thesis are reported, as roughly visualized in Figure 1-3. More specifically, the goals of the study are briefly described as follows:

1) *Introduction of a modeling theory and method*, as a solution-seeking study (Stol, Goedicke, & Jacobson, 2016), through a formal and rigorous mathematical framework supported by a concrete theory about software evolution and expansion during maintenance. Through this framework, reusable formal prediction models are derived which are used in the early stage of software architectural design domain. The purpose of these formal models is to compare maintainability degree and select the most maintainable option among competing design alternatives with respect to specific design attributes of a general problem.

2) *Illustration of the introduced modeling method* through the motivation example of the recursive part-whole aggregations in Table 1-1. The modeling method derives: a) the fundamental effort metrics for each design alternative and scenario type for a single scenario application as depicted in (2.a) stage in Figure 1-3, b) the formal prediction models, returning total effort predictions for each design alternative and for any number of applied scenarios, as depicted in (2.b) stage in Figure 1-3, and c) application of the derived formal models on the practical examples in Table 1-1, as indicated in (2.c) arrow in Figure 1-3.

3) *Demonstration of the applicability of the introduced modeling method* by applying it to three varieties (extensions) of the general problem presented in Table 1-1, incorporating the relevant design patterns of Decorator, Abstract Factory, Prototype, Observer, and Mediator, all introduced in (Gamma et al., 1994).

4) *Simulation of Software evolution* by introducing a simulation model that imitates the actual maintenance process and its highly uncertain and stochastic nature.

5) *Evaluation of the decision-reliability of the introduced, theory, modeling method, and derived formal models* by statistically validating their outcomes against simulated observations.

6) *Exploration of possible alternate uses of the introduced theory* through horizon analysis and decision-making under conditions of partial or full uncertainty.

## 1.4 Brief Review of Related Work (Existing Approaches)

Different approaches for maintainability assessment have been discussed in the literature taking into consideration software evolution, effort/cost estimation, and code complexity. Most approaches manage source code analysis through typical code metrics not appropriate to support early evaluation of design (pattern) alternatives. Moreover, existing approaches do not use formal models for evaluating design pattern combinations before code development. Such approaches lead to suboptimal results regarding code quality with regards to maintainability. Even the most relevant approaches (Bengtsson & Bosch, 1999; Bosch & Bengtsson, 2001) are strongly linked to minor specifications and functionality, with no links to design patterns. Thus, such approaches cannot support early evaluation of design patterns combination alternatives in an efficient way. Furthermore, the effectiveness of software maintainability prediction techniques and models has not been adequately proved (Riaz et al., 2009a; Shepperd & MacDonell, 2012). Methods, which are not related to well-known design pattern combinations for typical problems, are not easily reusable or adaptable. Without the insight provided by the structural behavior and the evolution pattern of the used design patterns in the event of major maintenance scenarios, existing approaches miss an important aspect of maintainability perspective, thus providing suboptimal estimations. This gap is even more obvious during the critical object-oriented software design stage before code development. Hence, there is great need for a well-defined systematic modeling method that generates formal comparison models well-fitted to specific design attributes allowing early maintainability assessment of design alternatives.

## 1.5 Brief Presentation of Proposed Theory

### 1.5.1 Modeling Theory and Method

The ultimate purpose of the introduced modeling theory is to derive formal models that provide effort estimations per design alternative for general design problems mainly for comparison purposes. During this early decision stage there is no source code for evaluation, thus the proposed theory examines (or predicts) the evolution of the engaged design patterns during maintenance. As a first step, the modeling method proposes a number of design alternatives (as combinations of established design patterns, i.e., CVP, CIBI) that address the general design problem under study. To analyze the evolution of each design alternative of the general problem, the method derives a number of major maintenance scenarios (as classes of resembling activities with common characteristics) that have high impact and likelihood to occur during maintenance. The method concentrates on those scenarios that add further functionality (i.e., new element, new operation) and expand the size of the software during maintenance since the expansion of software size is an innate trend of software evolution (Meir M. Lehman, Ramil, Wernick, Perry, & Turski, 1997) as confirmed by empirical evidence (Bakota et al., 2012; Barry, Kemerer, & Slaughter, 2007; C. R. Cook & Roesch, 1994; H. Gall, Jazayeri, Klosch, & Trausmuth, 1997; Jazayeri, 2002; M. M. Lehman, Perry, & Ramil, 1998; Yuen, 1988). Furthermore, the method derives a number of relevant design attributes (i.e., number of elements and operations) that affected by the major maintenance scenarios. In principle, the design attributes are quantitatively expressed and reflect the logical entities of the design problem (i.e., elements and operations) as represented by the code entities (i.e., methods, classes, modules) of the used design patterns. Each maintenance scenario affects each design alternative and relevant design attributes in different and conflicting ways depending on the pro and cons of the engaged design patterns. As the maintenance scenarios are applied in each design alternative, the required maintenance effort is quantitatively assessed based on an innovative measurement approach.

### 1.5.2 Structural Maintenance Cost (SMC) Metric

Measuring the future maintenance effort of a design alternative (combination of design patterns) is a challenge since during early design stage there is no source code for evaluation. To overcome this barrier, the proposed measurement approach analyzes and counts the structural changes for each design alternative per applied maintenance scenario. More specifically, the measure counts the number of required interventions (structural changes) that take place as a specific scenario applied in a design alternative. Taking advantage from the architectural behavior of the used design patterns in the event of a major maintenance scenario, the introduced measure expresses the required maintenance effort in terms of number of effected code entities (interventions, i.e., methods). Furthermore, the measure simultaneously counts different types of affected code entities (i.e., classes) to capture not only the number of interventions but the scattering degree or the locality of those interventions as well. The principal idea is that the number and the locality of required interventions are strongly related to code properties like coupling and cohesion degree, and thus these sizes are proportional to the required effort as also supported by empirical evidence (Araújo, Monteiro, & Travassos, 2012; L. C. Briand, Melo, & Wust, 2002; Jabangwe, Börstler, Šmite, & Wohlin, 2015). Since these code entities (i.e., methods, classes, modules) represent the problem's logical entities or design attributes (i.e., number of elements and operations), the required effort depends on the values of the problem's design attributes. Even if the number of interventions is not necessarily linked to actual (real-world) effort, the main intent of the proposed metric is to provide proportional equivalent effort estimations mainly for comparison purposes among design alternatives. That because, the actual effort of distinct (minor) interventions would be common for all design alternatives under evaluation, thus neutral concerning the decision-making process. Due to its properties, the introduced effort metric has been named

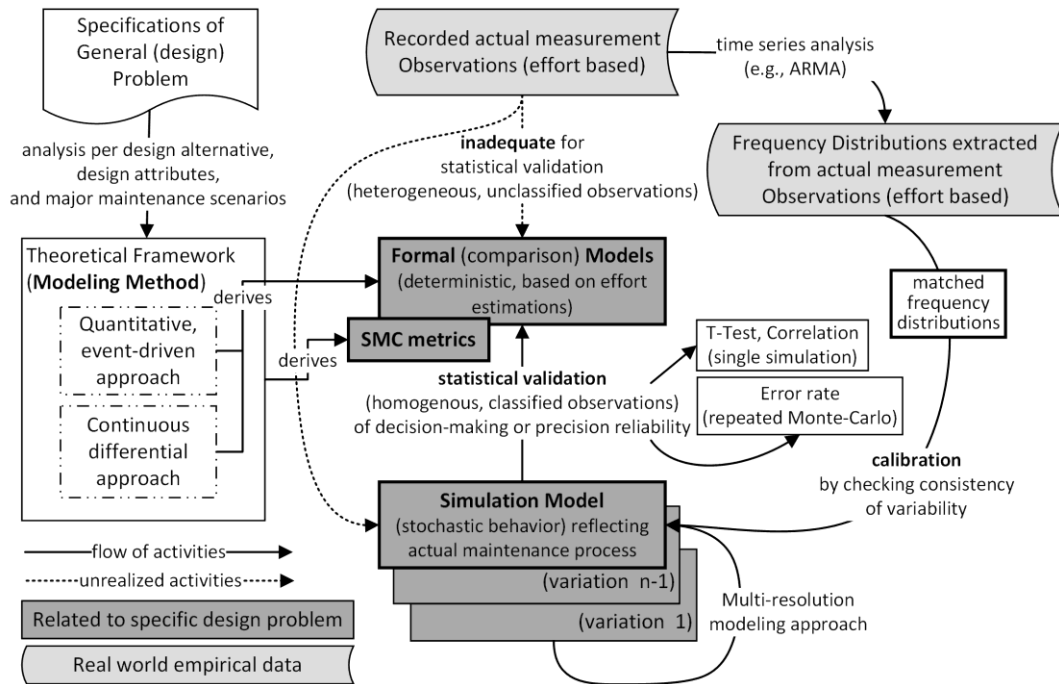


Figure 1-4: Overall presentation of the introduced modeling method, derived formal/simulation models, and validation process

Structural Maintenance Cost (SMC). Summarizing, SMC metric returns effort assessments per design alternative and maintenance scenario expressed in terms of number of affected methods (interventions) and classes (locality) as represented by the problem’s design attributes.

### 1.5.3 Deriving Formal Comparison Models

Given that SMC metric returns effort assessments for a single applied maintenance scenario, the proposed modeling method mathematically derives formal models per design alternative that estimate the overall required effort for any number of applied scenarios as depicted in Figure 1-4. Furthermore, since there several types of maintenance scenarios (i.e., new element, new operation), the engagement and assessment of scenario’s probabilities is required. Thus, scenario’s probabilities and design attributes distinguish a specific design problem (e.g., Interpreter, Compiler, GUI, etc.) as an instance of the general design problem (CVP vs CIBI) under study. However, the overall quality assessment of a design alternative requires the measurement of the relevant effort for different types of scenarios, each affecting the used design patterns in different and conflicting ways. Thus, a combined analysis of the effects of all scenarios based on their probabilities for each design alternative is required. By using the SMC metric, the proposed modeling method gradually calculates the required effort of several applied scenarios where each type of scenarios is engaged based on its individual probability.

The critical point is that the repeatedly applied maintenance scenarios gradually affect and shift the design attributes (i.e., increases the number of elements / operations) based on which the SMC metric of the next applied scenario is estimated. Mathematically, the modeling method gradually integrates the change rates of the problem’s design attributes and required effort for several applied maintenance scenarios based on their individual probabilities. This achieved by two ways: a) through a progressive quantitative and distinct (event-driven) analysis in chapters 3, and b) under the sight of continuous differential equations that examines the software expansion trend through the change rates of the problem’s design attributes in chapter 4. As a result of this probabilistic approach, the outcome of the derived formal models depends on the design attributes, the scenario’s

probabilities, and the number of applied scenarios. Thus, the derived formal models estimate the overall required effort per design alternative offering a suitable comparison base.

#### 1.5.4 Advantages of Modeling Method & Formal Comparison Models

The most important advantages of the introduced modeling method in chapters 3 and 4 are briefly listed below:

- Relies on the structural analysis of well-known and established design patterns, thus the derived formal models address difficult, general, and frequently tackled design problems in the field of software architecture.
- Resolves conflicting issues and tradeoffs among design alternatives concerning their pro and cons (advantages and disadvantages) in the event of different types of maintenance scenarios.
- Is capable to provide formal comparison models for different design alternatives, metrics, quality attribute requirements, and alternate design problems.
- Is capable to address general design problems for different levels of analysis from high level architectural design of systems to low level of object-oriented design.

The most important advantages of the introduced SMC metric in chapter 3 are briefly listed below:

- Is an adaptable metric that captures the expansion trend for each design combination under comparison by examining the structural evolution or the number of performed changes (number of interventions) on the affected code entities of the used design patterns.
- Returns effort assessments which are independent of accurate effort measurements, real-world observations, and realized costs by providing proportional equivalent effort assessments ideal for comparison purposes among design alternatives.
- Provides insight of the future required effort even in the absence of source code (during the early design stage) by taking advantage from the architectural behavior and structural evolution of the engaged design patterns.
- Relies on the explicit analysis of expansion scenarios which cover all the essence of the actual maintenance process among design alternatives mainly for comparison purposes.
- Effort/size assessments in terms of number of (classes and method) interventions are reliable measurement (proxy) units for comparison purposes in a mid-to-long term perspective.

The most important advantages of the derived formal comparison models are briefly listed below:

- They are independent of specific code implementations and run-time behavior and, thus it is ideal to support early decision-making among design alternatives during early design stage.
- They are sensitive to several design attributes and scenario probabilities, supporting decision-making for the entire design space (specific instances) of the general problem under study. Thus, they could be easily and repeatedly applied to any instance of the general problem under study.
- They are easily implemented in software for further analysis purposes (e.g., data manipulation, graphs generation, etc.) as parametric equations or dynamic functions.

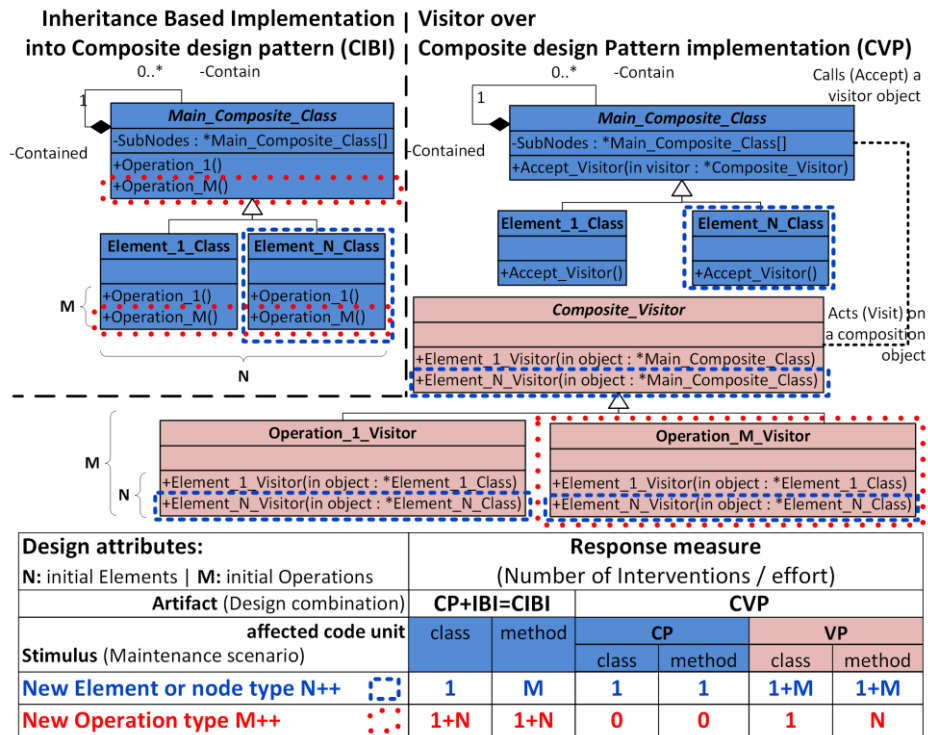


Figure 1-5: Conceptual class-diagram and relevant effort metrics of CIBI and CVP design alternatives.

- Their deterministic nature combined with their computational efficiency through software allows the exploration of the entire design space for a given design problem.
- Overcome the highly stochastic and uncertain nature of actual maintenance process by providing deterministic results (effort assessments) through parametric equations based on specific design attributes and parameters of each general design problem.
- The cumulative gain from the repeated use of the formal models (in terms of avoided maintenance effort) significantly overcomes their initial (one-time) derivation cost.

#### 1.5.4.1 Methods That Do Not Exploit Structural Evolution of Software

In general, methods that do not exploit the architectural behavior or the structural evolution of software during maintenance are unable to support early decision-making among design alternatives in an efficient way. During early design stage there is no code for evaluation since the actual code should be allocated in the preselected design structures (e.g., combinations of design patterns). Thus, in this early stage, the only available structures to evaluate are the selection among design alternatives that address the software requirements. These design decisions are in the core of architectural design theory (Bass et al., 2012) even if their conceptualization is relatively abstract. The proposed modeling method facilitates the comparison process and decision-making by analyzing the structural evolution of the used design patterns for major maintenance events. This is an active and more informative evaluation of maintenance perspective since evaluates the effect of various types of possible maintenance scenarios compared to the static evaluation of source code properties.

#### 1.5.4.2 An Example of Formal Model Application

Referring to the general design problem (CVP vs CIBI) in Table 1-1, the two events of new element and new operation are the major types of maintenance scenarios (stimulus) while the initial number of elements and operations are the problems' design attributes. The overall representation of the general design problem in the form of UML class diagrams

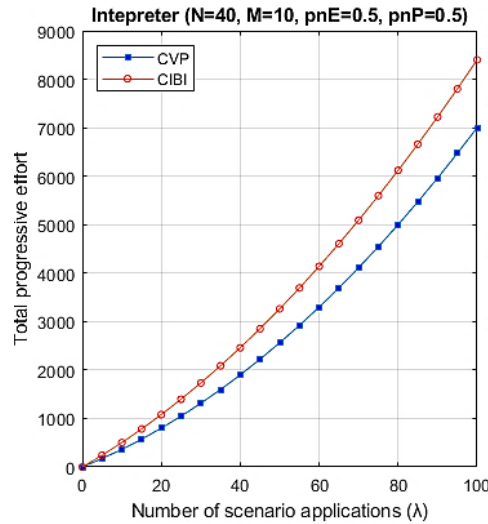


Figure 1-6: Results of the application of the Formal comparison Model on the practical example of Interpreter as a specific instance of the CVP vs. CIBI general design problem.

per design alternative (artifact) is visualized in Figure 1-5. In this representation, the change impact of each scenario per design alternative is quantitatively expressed by the number of required method and class interventions in connection with the problem’s design attributes. This change impact is captured by the introduced SMC metric. As an example, the SMC metric for adding a new element in CVP design alternative is equal to  $(M+2)$  method interventions into  $(M+2)$  different classes where factor  $M$  represents the number of operations as analyzed in chapters 3 and 4. Thus,  $SMC[\text{New element on CVP}] = 2(M+2)$ . Similar SMC metrics are derived for each maintenance scenario and design alternative of the general design problem under study.

By applying the introduced modeling theory, the formal models for each design alternative in the form of parametric equations are derived. For example, the  $\text{Total\_Effort}[\text{CVP}] = \frac{3}{2}\lambda^2 p_{nE} p_{nP} + \lambda p_{nP} N + 2\lambda p_{nE} M + 4\lambda p_{nE} + \lambda p_{nP}$  where  $N$  and  $M$  represent the initial design attributes (initial number of elements and operations),  $p_{nE}$  and  $p_{nP}$  represent the probabilities of the scenario types (new Element and new Operations respectively), and  $\lambda$  represents the number of applied scenarios. Thus, given the formal models in chapter 4, the total required effort per design alternative and any number of applied scenarios ( $\lambda$ ) can be directly computed and visualized.

As an example, referring to the specific problem’s instance of Interpreter in Table 1-1, the initial number of elements and operations are equal to 40 and 10 respectively. Furthermore, the probabilities of the two maintenance scenarios are equal to 0.5 and 0.5 respectively. The application of the derived formal models in the specific instance of the Interpreter ( $N=40, M=10, p_{nE}=P_{nP}=0.5$ ) is presented in Figure 1-6. In this case, CVP design combination is preferable since requires the lesser effort during maintenance. Notice that for different values of design attributes or/and scenarios probabilities the decision outcome could be different.

### 1.5.5 Simulating Software Evolution Towards Statistical Validation

Despite the argumentation about the validity of the introduced modeling method there still several concerns regarding the reliability of the derived formal models to conclude on correct design decision. The proposed modeling method introduces several assumptions concerning the stochastic nature of actual maintenance process in order to derive straightforward and deterministic equations of formal models. More specifically, the proposed modeling theory assumes a cyclical pattern of applied scenario types,

engagement of only expansion scenarios, constant scenarios probabilities, neutral size of code interventions, no aging issues (code decay), and neutral developers' experience level. To evaluate the decision-making reliability of multivariable formal models, their deterministic outcomes for several values of their parameters (reflecting specific instances of the general problem) should be statistically compared to actual effort observation recorded during the maintenance of real-world systems. Given the absence of such homogenous effort observations properly classified in respect to all problem parameters, there is no easy way for a statistical validation base on real-world observations.

To overcome this barrier, the simulation of software evolution that imitates the actual maintenance process is attempted as depicted in Figure 1-4. More specifically, a simulation model that replicates the variability and the highly uncertain and stochastic nature of actual (real-world) maintenance is introduced. The model replicates several underlying activities (overlooked by modeling method as assumptions) by engaging several random and stochastic factors while it follows the same structural evolution, design alternatives, parameters, and measurement approach as in formal models. Thus, it is properly adapted to the general design problem under study while offers a suitable validation base. Furthermore, due to its stochastic nature, the returned effort assessments demonstrate a variability for repeated simulations under the same parameters. Thus, the model returns massive effort assessments classified in respect to all problem's parameters ideal for statistical validation purposes.

However, the evaluation of the consistency of the simulation model against real-world observations is subject to the same constraints and limitation as in the case of formal models (i.e., heterogenous, and unclassified observations). Taking advantage from the fact that design decisions are based on the difference of proportionally equivalent effort estimations, decision-making reliability relays on the variability (precision) of effort estimations instead on their accuracy in terms of absolute values. The model's consistency has been tested through intensive calibration efforts (through multi-resolution modeling approach) by matching its variability with frequency distributions of real-world effort-based observations from the field of time series analysis as depicted in Figure 1-4. Thus, the consistency of the simulation model in terms of variability is well connected to real-world empirical evidence.

### 1.5.6 Advantages of Simulation Model

The most important advantages of the introduced simulation model are briefly listed below:

- Is adapted to the general design problem under study by replicating the same evolution pattern, design alternatives, design patterns, and measurement process, providing a suitable validation base.
- Is sensitive to the same problem's parameters as in the formal models providing massive amounts of homogenous and classified (effort-based) observations ideal for statistical validation purposes.
- Is well calibrated concerning its variability (precision) against frequency distributions of real-world observations from the field of time series analysis.
- Provides several parametric switches that control the simulation and allows the gradual engagement of stochastic behavior.
- Provides several favorable conditions by ensuring common comparison terms since its artificial nature offers sufficient control over several stochastic factors toward a better understanding of possible causal relationships.



## 1.6 Contribution and Innovations of the Study

### 1.6.1 Applicability of Formal Comparison Models

The applicability of the introduced modeling theory and SMC metric has been demonstrated on the significant and general design problem of part-whole aggregations in chapters 3 and 4. The derived formal models per design alternative (CVP, CIBI) offer a sufficient comparison base regarding their maintainability perspective. A series of graphs which represent a full-scale visual illustration of almost all solution space of CVP vs CIBI comparison is provided. In addition, the derived formal models have been applied on three indicative instances (Compiler, Interpreter, GUI, reported in Table 1-1) of the general problem to support the selection of the most maintainable design alternative. Furthermore, the computational pattern of the model has been compared with two similar metrics derived from the evidence of related works (Hills et al., 2011; Tom Mens & Eden, 2005). The results of the comparison showed that the assessments of the proposed formal models converge to a significant degree to the evidence of related works.

### 1.6.2 Adaptability of Modeling Theory

This thesis introduces a methodology on how the proposed modeling model can be used for comparing the impacts on maintainability for similar or different design pattern combinations or for alternate general design problems. For this purpose, an analytical step-by-step description of the suggested methodology, including requirements and limitations, is presented in chapters 3. Moreover, in chapter 5, the proposed modeling method is applied to three different extensions of the CVP vs CIBI general problem, assessing its applicability to even more realistic settings. In particular, the established design patterns of Decorator, Mediator, Observer, Abstract Factory, and Prototype have been engaged and modeled. The generated formal models have been tested on several specific instances of each general problem.

### 1.6.3 Computer-Aided Derivation of Formal Models

A computer-aided modeling framework developed in MatLab scripts is provided in chapter 4. This framework represents all the factors of each design problem (i.e., design alternatives and attributes, scenarios probabilities, SMC metrics, etc.) in the form of data sets and matrixes. It automatically generates the formal models (as dynamic functions - equations) per design alternative by performing the underline differential analysis and integration. This modeling framework is offered as a suitable template for any further adaptations such as different measures and alternate design problems. The general design problem CVP vs CIBI as well as its extensions in chapter 5 have been deployed on the modeling framework and provided in Appendix A for further research purposes.

### 1.6.4 Simulation Model & Statistical Validation

The proposed modeling method and derived formal models for the CVP vs. CIBI general design problem, analyzed in chapter 3 and 4, are statistically evaluated concerning their decision-making reliability. The statistical comparison is based on massive and homogenous measurement observations which have been generated by a well calibrated and highly stochastic simulation model that imitates the variability of actual maintenance process, introduced in chapter 6. The derived formal comparison models have been validated under several statistical techniques to evaluate the decision-making reliability of the proposed modeling method. A sample of one thousand possible system's instances with specific design attributes and scenarios' probabilities has been randomly selected. A simulation model that replicates the underlying activities of actual (real-world) maintenance process, providing sufficient, unbiased, classified, and homogenous validation data is introduced. The simulation model has been designed and developed in the forms of MATLAB© functional model and object-oriented entity model, engaging all problem's parameters, and providing additional switches for controlling the simulation

settings and environment. Several intermediate variations of the model based on the multi-resolution modeling technique has been tested to reach the desired stochastic behavior and realistic outcomes. Concerning the consistency criterion, the simulation model has been calibrated based on empirical evidence (frequency distributions) of relevant studies from the field of time series analysis (Raja, Hale, & Hale, 2009; Shariat Yazdi, Angelis, Kehrer, & Kelter, 2016). In principle, the simulation model imitates the stochastic nature of the actual maintenance process by incorporating developers' stochastic characteristics such as experience and learning rate as well as other random factors like uncertainty of scenarios' probabilities, alternate maintenance scenarios, non-repeated application patterns, the actual code size of interventions, code aging issues, etc.

Several intermediate results computed by the simulation model have been compared against formal models' deterministic predictions under the hypothesis testing of non-significant difference. T-test and correlation inferences are based on single (one-time) simulation while error rate assessment is based on multiple (repeated Monte Carlo) simulations per sample instance. The results demonstrate a high coefficient of correlation (near to 0.96) providing sufficient statistical evidence of formal model's decision-making reliability. Furthermore, the conducted hypothesis tests provide statistical evidence of formal models' long-term accuracy in terms of absolute effort predictions which, however, is a weak inference due to the lack of a strictly validation of the simulation model against actual (real world) estimations. Most importantly, the results showed that the formal models provide reliable decisions among design alternatives with an overall long-term error-rate about 8% with only 2% of it being critical in terms of significant wasted effort.

### 1.6.5 Alternate Computer-Aided Model Implementations

Alternate computer-aided implementations of the introduced models with the assistant of VENSIM tool are presented in chapter 7. This software tool can simulate physical and other phenomena and systems through the analysis of their key variables and integration of their change rates. This tool can represent both continuous and event-driven models while provides a variety of capabilities for representing and comparing the results of simulations. More specifically, the introduced discrete (event-driven) formal models in chapter 3, the continuous formal models in chapter 4, and the event-driven simulation model in chapter 6, concerning the CVP vs CIBI general problem, are implemented in VENSIM tool. The tool is demonstrated on the example of GUI implementation as an instance of the CVP vs CIBI general problem.

### 1.6.6 Design Decisions Under Uncertainty & Horizon Analysis

Several alternate and future perspectives of the introduced modeling method are presented in chapter 7. More specifically, the introduced modeling method and derived formal models are further analyzed to support decision-making under partial of full uncertainty. Thus, when software designers are unable to forecast the scenarios' probabilities in a precise manner. The technique is demonstrated on the formal models of the CVP vs CIBI general problem through further integration on intervals of probabilities factors. Furthermore, the horizon analysis technique is analyzed. This technique separates the entire maintenance period to subperiods, where for each subperiod different scenarios' probabilities are applied. In particular, the derived formal models of CVP vs CIBI problem are repeatedly applied on a specific instance of the general problem for different scenarios' probabilities. The technique is demonstrated on the example of Interpreter implementation as an instance of the CVP vs CIBI general problem.

### 1.6.7 Designers and Developers (Practitioners) Perspective

In a practical level, designers and developers can repeatedly use the derived formal modes in different instances of each general design problem to efficiently support decision-making among design alternatives, and thus develop more maintainable software of higher quality. Under this perspective, the formal models have been already derived for significant,

general, and frequently tackled design problems while they are offered as ready to use solutions. However, designers and developers can also analyze and derive formal models for alternate design problems in a practical level through the introduced modeling method for direct or future use.

### 1.6.8 Researchers and Accademia Perspective

In a theoretical level, researchers can analyze and derive formal models for alternate design problems through the introduced modeling method mainly for future use in a practical level by designers and developers. Furthermore, researchers can explore and adapt the modeling method under different factors, metrics, quality requirements, etc. In addition, the derived formal models can be statistically validated concerning their decision-making reliability by properly adapted simulation models. However, the statistical validation of formal models is not necessary since this thesis provide strong indications about the reliability of the introduce modeling theory in general.

### 1.6.9 Project and Quality Managers Perspective

In principle, project and quality managers are responsible for the efficient and effective development of quality software. Maintainability is one of the most important quality attributes requirements as discussed in this chapter. Under this perspective, project and quality managers are interested to adopt relevant methods that ensure software maintainability as part of their quality policies. The derived formal modes provide reliable early design-decision for general design problems leading to more maintainable software of higher quality.

### 1.6.10 Novelty and Critical Evidence

The introduced innovations of this thesis are briefly listed below:

- The introduced SMC metric, relayed on architectural behavior of the used design patterns, is an innovative measurement approach. It expresses required effort mainly for comparison purposes among design alternatives, thus not necessarily connected to accurate real-world observations. Hence, it's a suitable and versatile measure able to support early evaluation and selection among design alternative before code development.
- The progressive analysis through differential and probability- weighted equations is an innovation of the proposed modeling method, compared to other existing approaches. Maintainability is assessed under the sight of possible maintenance scenarios and their probabilities as a dynamic and progressive evolution process that gradually affects basic design attributes of the system.
- The structural evolution and expansion of the used design patterns per design alternative constitutes an insightful and innovative approach of software evolution during maintenance. This approach is in accordance with the confirmed increasing trend of software size during maintenance.
- The effort predictions of the introduced formal and simulation modes do not represent the entire general problem in a universal way. Instead, they are sensitive to several design characteristic (parameters) of the addressed design problem. In other words, they fragment the problem in distinct instances by classifying their outcomes with regard to the parameters of each problem's instance. This differentiation per sample instance is not limited only to the effort assessments but also extends to their variability degree.

Several aspects regarding the beneficial contribution of this thesis are briefly listed below:

- The average beneficial contribution (gain) from the repeated use of the derived formal models (in terms of avoided maintenance effort) concerning the entire design space

(possible instances) of each general problem lies between 25% and 90% of the optimal required effort which is a considerable performance.

- Early design decisions supported by formal models are particularly reliable regarding the risk taken by designers or else the possibility of an incorrect selection of a less maintainable design alternative, demonstrating a long-term average error rate 8% with only 2% of it being critical in terms of significant wasted effort even under assumptions of high variability.
- The introduced modeling theory provides formal models based on continuous integration through differential analysis, adequately describing an event-driven (distinct) phenomenon such software evolution during maintenance.
- The conducted error rate assessment based on simulated outcomes reveals the pattern or the traces of non-critical and critical error occurrences.
- The statistical validation of the formal models' decision-making ability is a strong indication that the introduced modeling method trustworthy describes the software evolution during maintenance process, delivering reliable formal models of limited decision-risk.

## 1.7 Thesis Organization

The remainder of this thesis is organized as follows:

- Chapter 2 overviews the existing works and relevant approaches concerning the analysis of design patterns, the general design problem of part-whole aggregations, effort estimations during software evolution, and maintainability assessment during early design stage.
- In chapter 3 the SMC metric and the quantitative (distinct) analysis of the general design problem of part-whole aggregations are introduced. The derived formal models are applied in several instance of the general problem. A step-step modeling methodology for deriving formal models for alternate design problems is presented.
- In chapter 4 the modeling method based on software (structural) evolution and continuous integrations is proposed. The method is demonstrated upon the general design problem of part-whole aggregations. A computer-aided modeling framework is presented that facilitates model derivation process.
- In chapter 5 three extended general design problems are modeled by attaching the design patterns of Decorate, Observer, Mediator, Abstract Factory, and Prototype. The derived formal modes are applied on indicative instances of each general problem. The exploration of almost the entire design space of each general problem is attempted to assess their overall contribution in terms of avoided wasted effort.
- Chapter 6 a simulation model that replicates the underlying activities and stochastic nature of actual software evolution during maintenance is proposed. The returned classified observations are statistically compared with formal model outcomes for a representative sample of one thousand problem's instances to evaluate the reliability of the introduced modeling theory.
- Chapter 7 discusses alternate techniques and uses of derived formal models such as decision-making under uncertainty and horizon analysis. Several alternate computer-aided implementations of formal and simulation models are presented.
- Chapter 8 discusses the overall conclusion of this thesis. In addition, the contribution of the study in the domain of architectural design and future work are discussed.

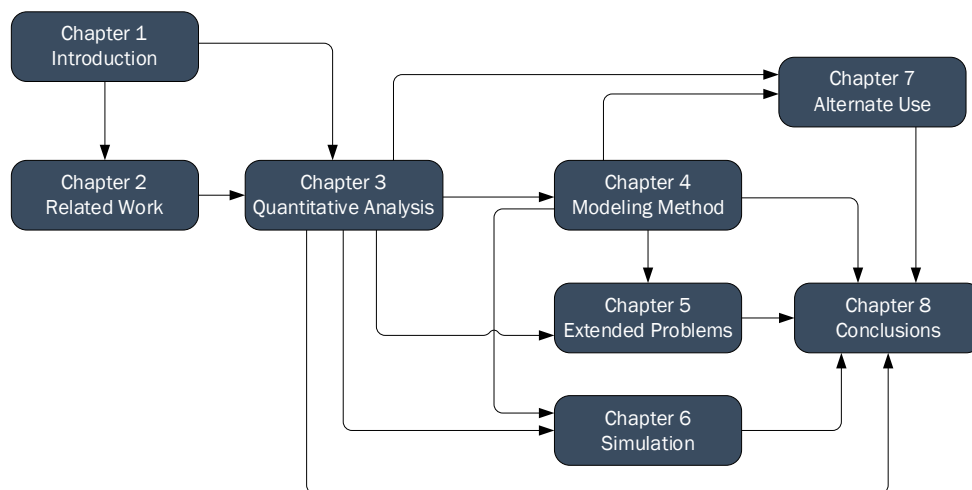


Figure 1-7: Dependence graph of chapters.

- The chapter 9 includes the appendix presenting technical details, supplementary diagrams, and evaluation results.

Figure 1-7 shows a dependence graph of the chapters. Priority should be given in chapter 3 in which the principal concepts of this thesis are introduced.

## 2 Related Work

### 2.1 Analysis of Design Patterns

There have been several research efforts on design pattern analysis. Gamma et al. (Gamma et al., 1994) introduced 23 general design patterns that have become popular in object-oriented programming community. Many of these have been used in finite element systems (Fenves, McKena, Scott, & Takahashi, 2004; Liu, Tong, Wu, & Lee, 2003; Mackie, 2002), or in compiler development (Neff, 2004). Also some of these design patterns have been used to identify best practice in object-oriented finite element program design (Heng & Mackie, 2009). Some of these like Visitor design pattern have been analyzed and developed through template libraries (Dascalu, Hao, & Debnath, 2005; B. C. d. S. Oliveira, Wang, & Gibbons, 2008). An approach for detecting these design patterns through a meta-model for design patterns representation has been proposed in (Bernardi & Di Lucca, 2010). A design pattern density metric that measures how much of an object-oriented design can be understood and represented as instances of design patterns has been proposed in (Riehle, 2009) through four real-world case studies. Most common object oriented design measures, methods and tools, like (Giuliano Antoniol, Fiutem, & Cristoforetti, 1998; Chidamber & Kemerer, 1994; N. E. Fenton & Pfleeger, 1998; Gibbons, 2006; Keller, Schauer, Robitaille, & Pagé, 1999; Srinivasan & Devi, 2014), quantify properties of lower level design units such as classes, attributes, methods etc. in order to evaluate the code structure. Briand et al (L. C. Briand, Daly, Porter, & Wüst, 1998; L. C. Briand, Daly, & Wüst, 1998, 1999; L. C. Briand, Wüst, Ikononovski, & Lounis, 1999), studied coupling and cohesion measures that quantify object oriented design quality. Also various object oriented design metrics and quality indicators have been proposed in (Bandi, Vaishnavi, & Turk, 2003; Victor R. Basili, Briand, & Melo, 1996; Li & Henry, 1993) in order to predict maintainability and maintenance performance. Almost all of them are based on individual characteristics of the source code. These methods are not focused on selecting proper design pattern combinations during the early stages of software development (software design). Therefore, they are only helpful for analysis and quality control of the produced source code and do not target explicitly the reduction of future software maintenance effort and cost.

A case study that analyses 39 version of an evolving object oriented software code is presented in (Bieman, Jain, & Yang, 2001). In that study, several perspectives such as relationships between design patterns, other design attributes, and the number of changes are analyzed. Its purpose is to extract evidence of improvements in adaptability by using design patterns and other design structures. That method captures design patterns through a set of software metrics and then links them to several future changes in the program. In this way, it reveals and relates some basic quality metrics (that capture design patterns) with several future changes based on case study evidence. The method discussed in (Bieman et al., 2001) also tries to determine the relationship between design structures and external quality factors such as reusability, maintainability, testability, and adaptability. Although the objective of the method discussed in (Bieman et al., 2001) is similar to the method proposed in this thesis, this method does not provide a general prediction of future changes on specific design pattern combinations during the software architecture design phase (before code development).

### 2.2 Visitor Design Pattern vs Inheritance-Based Implementation

Several object-oriented metrics and methods, like (Giuliano Antoniol et al., 1998; Bandi et al., 2003; Victor R. Basili et al., 1996; Bernardi & Di Lucca, 2010; Bieman et al., 2001; L. C. Briand, Daly, Porter, et al., 1998; L. C. Briand, Daly, & Wüst, 1998; L. C. Briand, Daly, et al., 1999; L. C. Briand, Wüst, et al., 1999; Chidamber & Kemerer, 1994; Dascalu et al.,

2005; N. E. Fenton & Pfleeger, 1998; Fenves et al., 2004; Gibbons, 2006; Heng & Mackie, 2009; Keller et al., 1999; Li & Henry, 1993; Liu et al., 2003; Mackie, 2002; Neff, 2004; B. C. d. S. Oliveira et al., 2008; Riehle, 2009; Srinivasan & Devi, 2014), have been proposed to help designers choose among Visitor and inheritance-based implementation or other design pattern combinations. Almost all of them are based on individual characteristics of the source code through source code analysis. In the proposed context, it seems that none of these methods can support the early evaluation (during software design and before code development) of these design patterns through a formal mathematical way, especially when a problem that has an extensible set of operations and an extensible Composite structure is addressed. Thus, through the result of the proposed context, software designers and developers can not choose between Visitor design pattern and inheritance-based implementation combination over a part-whole representation in an early stage, during software design phase.

The use of these patterns has been also widely discussed for the well-known Expression Problem which was coined by (P. Wadler, personal communication, 1998) and extended by (Zenger & Odersky, 2005) in order to illustrate modular extensibility issues in software evolution, especially when involving recursive data structures like Compositions. Expression Problem refers to a fundamental dilemma of programming: to which degree can an application be structured in such a way that both the data model and the set of virtual operation over it can be extended (extensibility in both dimensions), without the need to modify existing code, without the need of code repetition and without runtime type errors. An extra requirement added by (Zenger & Odersky, 2005) which predicts the possibility for combinations of independently developed extensions that can be used jointly. Several solutions have been proposed (Krishnamurthi, Felleisen, & Friedman, 1998; B. C. Oliveira, 2009; B. C. d. S. Oliveira & Cook, 2012; Torgersen, 2004) which are referred to the Expression Problem, all of them mainly focused on presenting a variety of object oriented programming methods and technics over Composite, Visitor design patterns and inheritance based implementation. These solutions present variations of these design patterns by using features such as generics, templates, type-parametrization, and subtyping that are available in many (mainly advanced) object-oriented languages, attempting to satisfy Expression Problem requirements while improving the extensibility of programs. Most of these solutions introduce high code complexity which finally deforms the initial design pattern structures. Although the proposed context satisfies the Expression Problem requirements in such a way that both the data model and the set of virtual operations over it can be extended using Composite, Visitor design patterns and inheritance-based implementation, it does not propose any method for selecting proper design pattern combinations for a given problem. Usually, software designers must choose specific design pattern combinations; otherwise, high code complexity could be introduced. In addition, developers often prefer to follow the initial simple structure of design pattern avoiding major reforms. Moreover, modifications or additions during software maintenance usually take place on the initial code (modules), without satisfying all Expression Problem requirements. Most of the pre mentioned concerns remain even for the latest proposed solution in (Wang & Oliveira, 2016) which presents a simple solution of Expression Problem.

The proposed method of this thesis is not intended as a solution for the Expression problem since it is not attached to the Expression Problem requirements. Furthermore, the proposed method tries to eliminate all the pre mentioned concerns and cover the need for a formal method for choosing between specific design pattern alternatives. Furthermore, the proposed method can be applied for early selecting proper design pattern combinations (like Visitor or inheritance-based implementation) even if any of the Expression Problem solutions is adopted in a later stage.

Perhaps the most related case study which compares the Visitor pattern with the Interpreter pattern (an inheritance-based implementation) is presented in (Hills et al.,

2011). In that study, two nearly equivalent versions of an interpreter for a programming language (one using Visitor and other using Interpreter pattern) are compared on their maintenance characteristic and execution efficiency. The method tries to measure the impact of future extensions for new expressions (nodes) or new operations through maintenance scenarios, for both versions, introducing a new metric called computational complexity. This new metric tries to measure not only the effort to transform the system, but also the effort to analyze it before applying any transformations. In other word computational complexity can be considered as the complexity of maintenance or as the required maintenance effort for specific maintenance scenarios. Summarizing, this model estimates maintenance effort by counting various individual actions (performed by the developer) for different maintenance scenarios. Although the results of this method are depended on source code analysis for a specific case study, the computational complexity metric has been used as an alternate metric for the proposed method in this thesis. Again, this case study cannot support the early evaluation (during software design and before code development) of these design patterns through a formal mathematical way, especially when a problem that has an extensible set of operations and an extensible Composite structure is addressed.

### 2.3 Effort Estimation During Software Evolution and Maintenance Process

Software evolution and maintenance process have been extensively discussed in the literature (Arbuckle, 2011; Chapin, Hale, Khan, Ramil, & Tan, 2001; Demeyer, Mens, & Wermelinger, 2001; Kemerer & Slaughter, 1999; Meir M. Lehman & Ramil, 2002; Tom Mens & Demeyer, 2001). Several general frameworks, methods and models (Dubey & Rana, 2011; Granja-Alvarez & Barranco-García, 1997; Heitlager, Kuipers, & Visser, 2007a; Kumar, Krishna, & Rath, 2017; Land, 2002) have been proposed toward effort estimation required during software maintenance or evolution process. In the proposed context, none of these methods can support the early comparison (during software design before code development) among visitor design pattern and inheritance-based approaches through a formal mathematical way, especially when a problem that has an extensible set of operations and an extensible composite structure is addressed.

Furthermore, several approaches and models (Harald Gall, Hajek, & Jazayeri, 1998; Ramil & Lehman, 2000, 2001) have been proposed toward software evolution prediction mostly through the use of low-level code metrics, ISO metrics, object-oriented specified metrics, and so forth. None of these measures and techniques seem to capture code's structural behavior during software evolution and maintenance, frequently leading to general probabilistic (Bakota, Hegedűs, Körtvélyesi, Ferenc, & Gyimóthy, 2011; Bakota et al., 2012) or static (Ahn, Suh, Kim, & Kim, 2003; Aloysius & Arockiam, 2012; Bandi et al., 2003; Bansiya & Davis, 2002; Bartosz & Pawel, 2010; Dagpinar & Jahnke, 2003; Fioravanti & Nesi, 2001; Hayes, Patel, & Zhao, 2004; Hayes & Zhao, 2005; Rizvi & Khan, 2010) models for predicting or measuring maintainability degree, delivering suboptimal estimations. Moreover, many of these studies analyze software maintenance after it has been occurred, more as a retrospective than as predictive approaches.

In general, the most commonly used metrics and factors toward actual effort estimation are source code measurements, and people related metrics, as shown by Wu et al. (Wu, Shi, Chen, Wang, & Boehm, 2016) Systematic Literature Review results. Additionally, there is not much evidence on the effectiveness, accuracy, and validation of software maintainability prediction techniques and models (Riaz et al., 2009a; Shepperd & MacDonell, 2012). Although there are several works in the literature regarding design pattern recognition and analysis (Giuliano Antoniol et al., 1998; Bernardi & Di Lucca, 2010; Dascalu et al., 2005; Keller et al., 1999; Riehle, 2009), there is no particular theory or method for deriving formal comparison modes based on the structural behavior/evolution of the engaged design patterns. Even the prediction models and



techniques based on time series analysis (G. Antoniol, Casazza, Di Penta, & Merlo, 2001; Raja et al., 2009; Shariat Yazdi et al., 2016) for predicting the evolution of various software aspects such as size, clones, bugs, defects, and changes, are not closely related to system’s design patterns and attributes. In the proposed context, none of these methods can support the early comparison among design (pattern) combinations alternatives in a formal mathematical way. This gap is even more obvious in the case of early evaluation during the crucial design stage before code development.

## 2.4 Assessing Maintainability During Early Design Stage

A method for the prediction of software maintainability during software design phase is presented by Bengtsson and Bosch (Bengtsson & Bosch, 1999). The method takes the requirement specifications and the design of the architecture as input and generates a prediction of the average effort for a maintenance task. Scenarios are used by the method to concretize the maintainability requirements and to analyze the architecture for the prediction of the maintainability. Although the method introduces individual scenario probabilities and distinct affected volumes for each component and scenario combination in a simple linear equation, it does not take under account several issues such as the change rate of the component instances. Furthermore, the system components and maintenance scenarios are strongly linked to minor specifications and system’s functionality, away from design pattern logic. This approach requires at least a detailed architectural design for all functionalities, which is not a strictly early evaluation. Moreover, without analysis of design pattern behavior, the number of potential maintenance scenarios is especially large, making the scenario selection and their weights a subjective and time-consuming process. In any case, the method seems that cannot support early comparison between design pattern combinations alternatives in an efficient and unbiased way. In addition, methods, which are not related to well-known design pattern combinations for common problems, are not easily reusable or adaptable.

A revised technique for analyzing the optimal maintainability of software architecture based on a specified scenario profile is presented by Bosch and Bengtsson (Bosch & Bengtsson, 2001). The new equation of optimal maintenance effort engages additional corrective factors as productivity measures under several conditions to improve the model’s prediction ability. However, this technique also suffers from the same concerns as the previous method (Bengtsson & Bosch, 1999).

## 3 Quantitative Analysis of Design Problems

### 3.1 Chapter Overview

In this chapter, a comparison model concerning the application of the object-oriented Visitor Pattern (VP) and Inheritance-Based Implementation (IBI) approaches on structures based on the Composite design Pattern (CP) is discussed. The proposed model is independent on specific code implementations and run-time behavior. So, it can be implemented in a very early stage, during software design, before code development to help designers make the right decisions and reduce the required effort and cost of software maintenance. The proposed model introduces a probabilistic approach for basic maintenance scenarios, based on specific design pattern behavior and individual problem characteristics or design attributes such as initial number of composition's elements and operations. It analyses software maintenance as a progressive evolution process and estimates maintenance effort by introducing the Structural Maintenance Cost (SMC) metric. The proposed SMC metric concentrates on the assessment of maintainability characteristic through the estimation of required maintenance effort. The metric focuses on the number of interventions as well as on the concentration degree (locality) of those interventions for each maintenance scenario as a specialization of software entropy concept (Bakota et al., 2012). Through the SMC metric, maintainability is related to the ease of future maintenance of software code in terms of numbers of interventions or changes as well as the locality of those interventions expressed by the numbers of classes (or code units in general) that are affected. SMC metric provides proportional equivalent effort assessments per design alternative mainly for comparison purposes, thus without the need of accurate assessments in terms of real-world maintenance cost. The progressive analysis is an innovation of the proposed method, compared to other existing approaches. The proposed approach provides a formal model of the behavior of CIBI (CP+IBI) and CVP (CP+VP) design combinations even when the structure and the operations' set are both extendable. Moreover, the model's computations and graphs can be easily implemented in software.

The model has been implemented and tested on many real cases or instances of the significant part-whole representation problem, three of which are presented as motivation and implementation examples in Table 1-1. Furthermore, the computational pattern of the model has been compared with two similar metrics derived from the evidence of related works (Hills et al., 2011; Tom Mens & Eden, 2005). The results of the comparison showed that the assessments of the proposed model converge to a significant degree to the evidence of related works. Also, it has been proven that the proposed model can be reliably used at a very early stage, before code development, for the comparison and selection among CIBI and CVP design alternatives, focusing on software maintainability while delivering reliable results.

The model provides a series of graphs which represent a full-scale visual illustration of entire solution or design space of CIBI and CVP comparison. Also, it has been induced that CVP implementation is preferable when companies dedicate more experienced resources (e.g., senior developers) during software maintenance. Overall, IBI is easier (than VP) to be understood and applied since it is preferable or easier to maintain by less experienced developers.

Furthermore, this chapter introduces a methodology on how the proposed model can be used for comparing the impacts on maintainability for similar or different design pattern combinations and general design problems. For this purpose, an analytical step-by-step description of the suggested methodology, including requirements and limitations, is presented.

The context of this chapter is based on the motivation examples in chapter 1, and related work in chapter 2. The rest of this chapter is organized as follows. Subsection 3.2 provides a brief presentation of the general principles that support the proposed model. Subsection 3.3 presents the general decision problem and the used design patterns under evaluation. Subsection 3.4 defines structural maintenance cost metric. Subsection 3.5 quantifies individual maintenance cost. Subsection 3.6 presents the quantitative analysis for a single future addition. Subsection 3.7 introduces progressive analysis based on scenario probabilities and presents a graphical analysis for any number of future additions. Subsection 3.8 presents application examples of the model and a comparison of three existing related measures. Subsection 3.9 suggests a step-by-step methodology and an extension example. Finally, in subsection 3.10, the model's validity challenges, limitations, future research issues, and conclusions are presented.

## 3.2 General Principles

### 3.2.1 General Architectural Design Principles

In general, the activities of the architectural design entails a series of design decisions (Bass et al., 2012). During these activities, several architectural patterns and tactics representing the available body of knowledge, practices, methods, and techniques are applied by software engineers and designers to address all the functional and quality attribute requirements of a system. To conclude on the most proper and beneficial architectural patterns and tactics, software engineers are constantly faced with design decisions. Many of these design decisions are critical since they are usually irreversible and their impact on systems performance is significant (e.g., concerning maintainability perspective). Such critical design decisions need to be made after systematic evaluation of the impact of each available option otherwise the entire design process is subject to high risk and likelihood to conclude on suboptimal solutions.

A typical flow diagram of the architectural design activities is presented in Figure 3-1. The design process begins with the analysis of system's specifications and requirements which entails several sub activities such as defining patterns of general problems, specifying logical entities that represent functional requirements, specifying modules and responsibilities to satisfy functional requirements, and considering quality attribute requirements that must be met. Next, a design model, usually in the form of UML class diagrams, is derived base on established design patterns form the available body of knowledge. These architectural design patterns are properly selected to fit general problem's requirements and quality attribute requirements. This activity is repeatedly performed until all functional requirements has been satisfied. However, in most of the cases, there are many architectural patterns that address the same requirements in different ways leading to several design alternatives (or artifacts) under consideration.

Furthermore, a design model could be further refined to satisfy the pursued quality requirements. Towards this directions, architectural tactics from the available body of knowledge are applied to a design model to improve its design structure. For example, tactics like splitting or rearranging responsibilities are used to increase cohesion and reduce coupling among the model's logical entities in a try to improve the quality attribute of maintainability. However, the critical point is whether these changes actually improve the pursued quality of a design model. Thus, during the next activity, a set of stimulus of possible events that may occur in the future are selected. At the same time, the response, or the effect of these stimulus on the design model is assessed. To be more precise, that response is quantitatively expressed through a response measure which captures the effect of the stimulus on the design model under the view of the pursued quality attribute. This measurement approach is assisted by evaluation models and techniques that allows designers to decide whether the applied tactics are actually improve the pursued quality of a design model. This kind of design decisions are critical and there is an increasing need for evaluation models able to efficiently and trustworthy support such decisions.

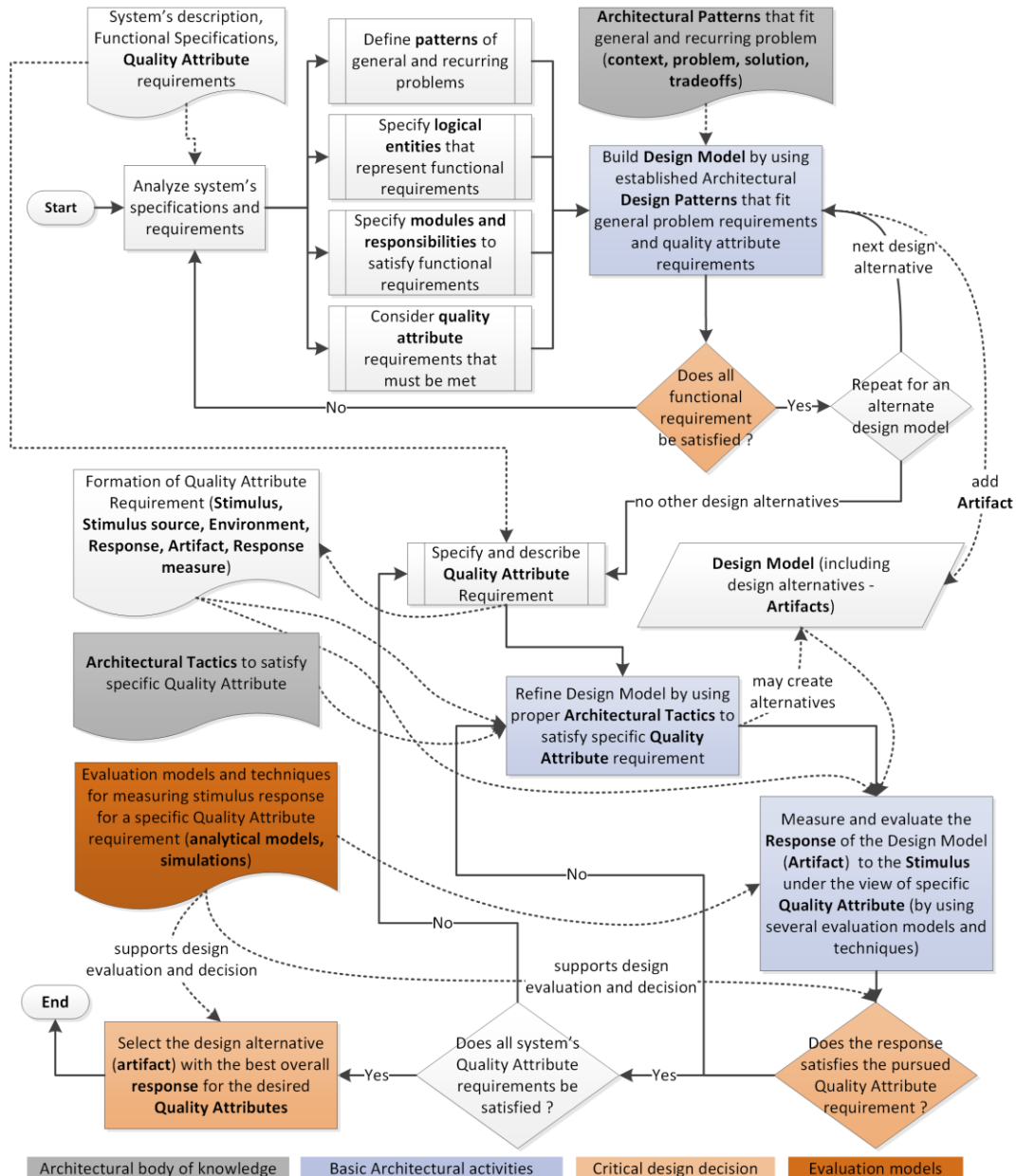


Figure 3-1: Typical flow diagram of architectural design activities.

Finally, designers have to evaluate several design alternatives and select among them the most beneficial from the point of view of the pursued quality requirement. Again, such design decisions are assisted by evaluation models and techniques that allows designers to decide whether the pursued quality requirement of a design alternative is better than others. Once more, this kind of design decisions are critical and there is an increasing need for evaluation models able to efficiently and trustworthily support such decisions especially for general, significant, and frequently tackled design problems.

The introduced in this chapter modeling method and derived formal models are offered as a suitable response measure and comparison (or evaluation) base among design alternatives with regards to their maintainability perspective.

### 3.2.2 Object-Oriented Design Principles

To address difficult design problems using computer systems, each problem should be firstly analyzed and designed before the development of the source code. These tasks are

performed by Software Engineers and designers through scientific theories, methods, tactics, and tools. One of the main tasks of object-oriented modelling is the Object-Oriented Design task during which software engineers focus on the conceptual solution (concerning software and hardware perspectives) for each problem rather than the solution itself. An important responsibility of object-oriented designing process is the definition of the relationships between the basic structural elements of the software as well as the selection of the proper Design Patterns which satisfy software quality requirements and specifications (Gamma et al., 1994). One of the most important concerns of object-oriented methods is to develop software and systems that are easily adaptable and maintainable. More details about software engineering as well as object-oriented and architectural design are presented in (Larman, 2004; R. S. Pressman, 2001; Sommerville, 2010). This analysis places the decision of selecting proper design patterns early into the general life cycle of software development while it highlights the importance and the impact of these decisions in the design process. Software designers can directly use the design patterns during system's design as part of a wider software life cycle model such as traditional sequential development, waterfall or agile.

Design patterns are suitable for addressing well specified, general, and significant design problems which usually are repeatedly addressed during the stage of software design. The design patterns that are being described in this thesis are modelled object-oriented descriptions of interrelated objects and classes, which are properly parameterized to address general design problems. They represent the relationships between objects participating to the solution and describe their collaborations and associations. By facilitating reuse of proven solutions, design patterns help improving software quality and reduce development and maintenance time, effort, and cost. Furthermore, object-oriented design patterns are especially dedicated on improving adaptability, since patterns generally increase the complexity of an initial design to ease future enhancements. For the design patterns to provide benefit in terms of maintainability, they must reduce the cost of future adaptations. In this thesis design patterns are graphically represented through class models of UML diagrams (Larman, 2004).

### 3.3 General Decision – Design Problem

#### 3.3.1 Recursive Hierarchies of Part-Whole Representations

During the phase of software design, several general and significant implementations which have solutions based on recursive hierarchies of part-whole representations/aggregations through composite structures of objects are confronted. In many of these cases, a set of different operations are performed on the objects (elements) of these structures, where different classes of objects are handled in a unique way by each distinct operation. Implementations with such characteristics are common in software design and constitute a general and significant design problem, searching for widely acceptable solutions. Classical paradigms of part-whole representation/ aggregation problems are computer aided design software and compilers (Aho, Lam, Sethi, & Ullman, 2006; Baldwin, 2003; Cooper & Torczon, 2011; Neff, 1999). Compilers usually form compositions of objects to internally represent intermediate representations (IRs). In all these implementations, it is common that each distinct operation (like drawing or filling for design software or like type checking for compilers) is applied in a unique way for every distinct class of objects (like line, rectangle, block, window objects for design software or like identifiers, nonterminal/terminal tokens for compilers). So, during the phase of architectural design, software engineers are responsible to design software of high-quality standards focusing on its maintainability perspective. Toward this direction, designers use combinations of well-known design patterns to represent part-whole representations in an efficient and straightforward way in order to implement different operations onto distinct type of nodes.

### 3.3.2 Engaged Design Patterns

A typical problem faced by software engineers, is the choice of a proper pattern as well as the choice of a flexible and systematic way for applying operations on distinct elements (class objects) of a composition. Over the years, various object-oriented design patterns have been proposed in order to solve the above problems. Composite, Visitor, and inheritance-based implementation are three of the most widely used design patterns, which have been proposed by software engineers through their experience in the field of software design. Composite and Visitor design patterns that have been used in this work have been introduced by Gamma et al. and are fully presented and analyzed in (Gamma et al., 1994).

- The Composite design pattern (CP) is an inheritance composition of classes used to represent the part-whole hierarchy of different types of objects.
- The inheritance-based implementation (IBI) is a simple and straightforward object-oriented approach that takes advantage of the inheritance attribute of object-oriented languages and as explained later can be (almost) matched with the Interpreter design pattern. Based on this similarity, the inheritance-based implementation is often referred as Interpreter design pattern or even as a naïve design pattern.
- The Visitor design pattern (VP) is slightly more complex, using distinct classes for visitor operations, while again it takes advantage of the inheritance attribute.

All patterns take advantage of the virtual method and polymorphism attributes of Object Oriented languages during the calling of their methods with different object reference (pointer) types, as presented in (Schildt, 2002). In most of the cases, the Inheritance-based implementation (IBI) and Visitor design patterns (VP) are combined with the Composite design pattern (CP).

Furthermore, there are several other well-known design patterns, like Interpreter, Iterator, and Flyweight, also presented in (Gamma et al., 1994), that could be combined with the pre mentioned design patterns to offer additional functionalities. However, CP, IBI and VP are the most important and basic design patterns that determine the quality characteristics of the software.

#### 3.3.2.1 Composite Design Pattern

In a typical hierarchy of classes, the inheritance relationship usually represents a generalization between classes rather a part-whole representation. The main intent of Composite design pattern (CP) is to compose objects into tree structures to represent part-whole hierarchies. CP lets clients treat individual objects and compositions of objects uniformly (Gamma et al., 1994). Referring to Figure 3-2, a typical structure of CP for a CAD system is presented. Point and Line objects are usually contained in objects of graphic Blocks which also can contain other Blocks' objects, etc. There is no need for a Point or Line object to have all the attributes and methods of a Block object. In this case, Blocks are compositions of other Blocks, Lines and Points. In order Block objects to contain Lines, Points, and other Block objects, in a part-whole hierarchy based on CP, all sub-classes (Block, Line and Point) should be declared as concrete (non-abstract). These classes should be generalized (through inheritance property) to a new abstract class (e.g., Structure\_Element). In this way a Point object with no content doesn't have the attributes of the Block class (like the vector of the contained sub objects, which in this case is a non-necessary memory cost). Now, clients and internal methods can treat to individual objects (such Lines and Points) and compositions of objects (such Blocks) uniformly through references to the abstract class (Structure\_Element) taking advantage of polymorphism property supported by object-oriented languages.

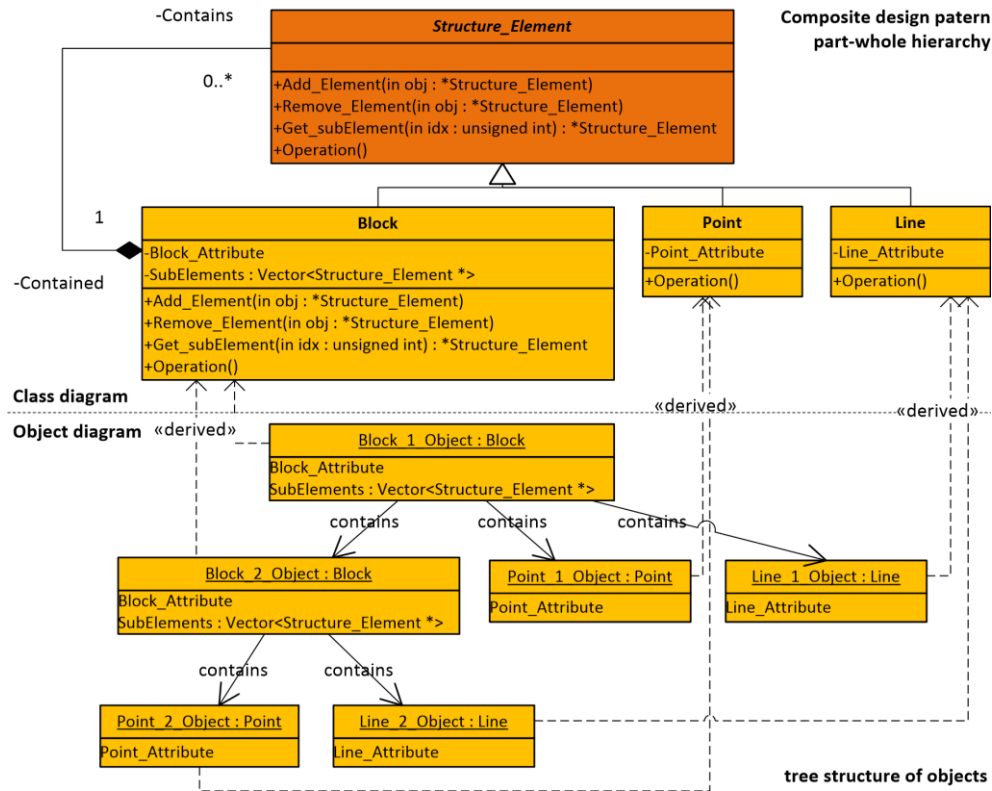


Figure 3-2: Example of class and object diagrams for typical hierarchies of objects.

As a general conclusion, in the Composite design pattern (Gamma et al., 1994), all classes except leaf classes of a hierarchy should be usually declared as abstract classes. This means that objects can be instantiated only through leaf classes. In this pattern the number of distinct node types, which can be represented, is equal to the number of leaf classes.

### 3.3.2.2 Inheritance-Based Implementation

In a Composite structure of a part-whole representation, where distinct operations (methods) are performed on different type (classes) of objects, the inheritance-based implementation can be used. This simple straightforward object-oriented approach is based on inheritance attribute of object-oriented languages and can be considered even as a naive design pattern. Furthermore, it is equated to the data-centered approach of the Expression Problem solutions. In addition, as presented in (Hills et al., 2011), can be (almost) matched with the Interpreter design pattern (Gamma et al., 1994). In this case, the interpreter operation (method) is implemented into the Composition pattern of the language through the inheritance-based implementation. In the general case in Figure 3-3, all distinct operations are declared as virtual methods inside the abstract root class of hierarchy. The implementation of every distinct operation (method) is placed in each distinct object (leaf) class of hierarchy. In this way, an operation (method) can be repeatedly called (e.g., recursively) through a general pointer type of the abstract root class (based on polymorphism). There is no need to know the specific pointer type of the (leaf) class from which an object has been instantiated.

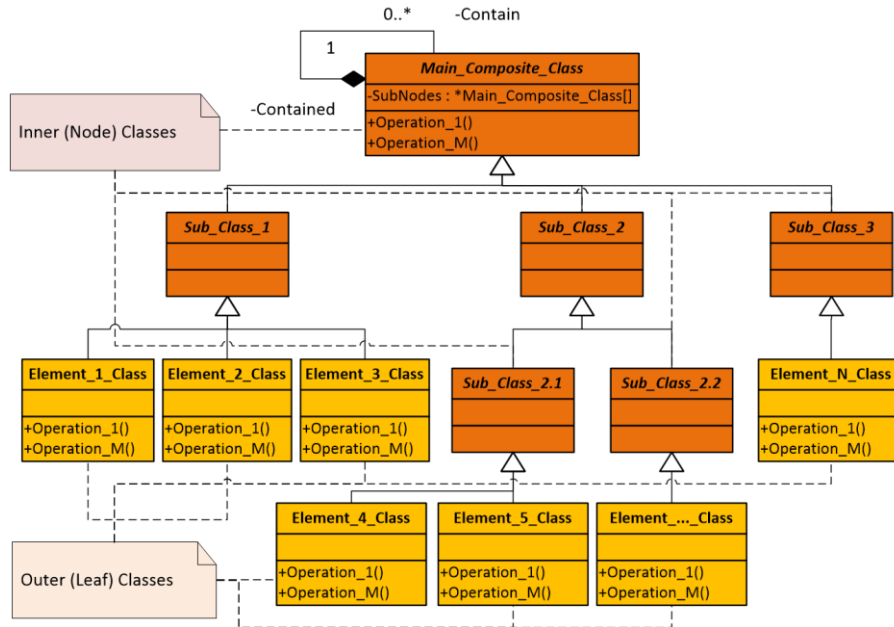


Figure 3-3: Example of class diagram for a typical structure based on Composite design pattern (CP) and inheritance-based implementation (IBI).

A disadvantage of the IBI is the fact that methods of a specific operation are distributed in different (leaf) classes. In this pattern, the addition of a new operation requires a new method implementation in every distinct (leaf) class. Contrariwise, the addition of a new type of node (leaf class) requires all new (operations) methods be placed in a single (new) class. This pattern makes adding new types of nodes (elements) easier (Gamma et al., 1994). This statement is still valid in case of modifying existing nodes (element) thanks to concentration (locality) of interventions. A typical code example based on CP and CIBI pattern is presented in Figure 3-5.

```

Composite_Design_Pattern.h
#ifndef COMP_DP
#define COMP_DP
class Main_Composite_Class {
private:
    List<Main_Composite_Class*> *SubNodes;
public:
    virtual void Operation_1();
    virtual void Operation_2();
    ...
    virtual void operation_M();
    void Add_SubNode(Main_Composite_Class*) {...};
};
class Sub_Class_1: public Main_Composite_Class {...};
class Sub_Class_2: public Main_Composite_Class {...};
class Sub_Class_3: public Main_Composite_Class {...};
class sub_class_2.1: public sub_class_2 {...};
class sub_class_2.2: public sub_class_2 {...};
#endif

Element_1_class_module.cpp
#include "Composite_Design_Pattern.h"
class Element_1_Class: public Main_Composite_Class {
public:
    void Operation_1() {
        // operation 1 code
    };
    void Operation_2() {
        // operation 2 code
    };
    ...
    void operation_M() {
        // operation M code
    };
};

Element_2_class_module.cpp
#include "Composite_Design_Pattern.h"
class Element_2_Class: public Main_Composite_Class {
public:
    void Operation_1() {
        // operation 1 code
    };
    void Operation_2() {
        // operation 2 code
    };
    ...
    void operation_M() {
        // operation M code
    };
};

Element_N_class_module.cpp
#include "Composite_Design_Pattern.h"
class Element_N_Class: public Sub_Class_3 {
public:
    void Operation_1() {
        // operation 1 code
    };
    void Operation_2() {
        // operation 2 code
    };
    ...
    void operation_M() {
        // operation M code
    };
};
    
```

Composite attributes:  
N initial Elements / M initial Operations

Figure 3-4: Typical code example based on Composite design pattern (CP) and inheritance-based implementation (IBI).

### 3.3.2.3 Visitor Design Pattern

Instead of using the inheritance-based implementation approach, the Visitor design pattern (VP) can be used. This is a more complicated design pattern which is also based on inheritance attribute of object-oriented languages. Furthermore, it is equated to the operation-centered approach of the Expression Problem solutions. In the general case in Figure 3-5, for every distinct type of node (leaf class) a new method is declared as virtual



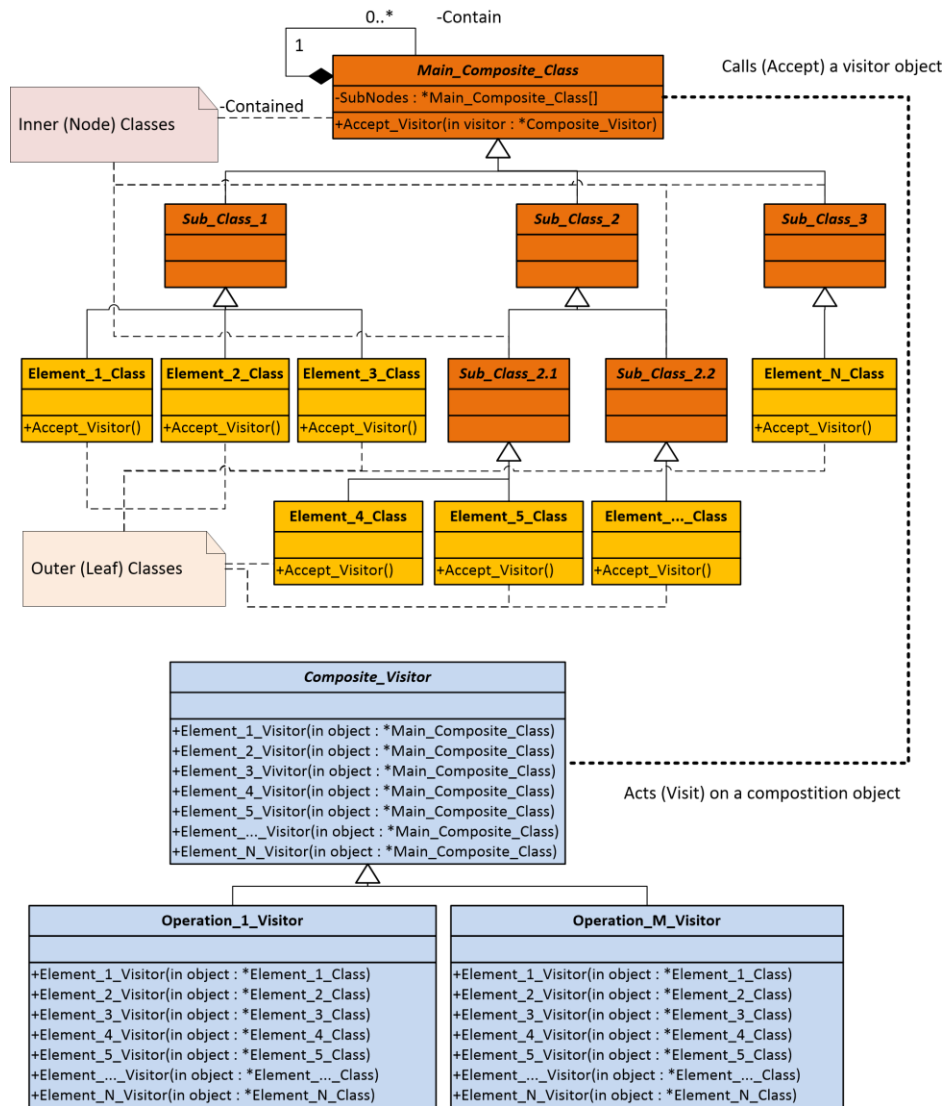


Figure 3-5: Example of class diagram for a typical structure based on Composite and Visitor design patterns (CP, VP).

method inside a root abstract class called Visitor. Also, for every distinct operation, a new sub-class is created which includes all the implementations of the methods of distinct node types for this specific operation. Furthermore, a single method, Accept\_Visitor(), is declared as virtual method inside the root abstract class of Composite hierarchy. The implementation of Accept\_Visitor() method is placed in each distinct object (leaf) class of hierarchy. The Accept\_Visitor() method usually contains very simple code; it gets a Visitor object reference (specific process) as argument and then calls the proper method (node type) for its node type by passing to it its object reference (pointer) as argument. In this pattern, a Visitor object (process) can be repeatedly implemented on Composite objects (e.g., recursively) through calls of its Accept\_Visitor() methods. There is no need to know the pointer type of specific Visitor sub-class or the pointer type of the (leaf) class from which objects have been instantiated. A full analysis and examples of the Visitor design pattern are presented in (Gamma et al., 1994; Palsberg & Jay, 1998; Santos Oliveira, 2007; Visser, 2001).

An advantage of the VP is that all methods of a specific operation are gathered in a single Visitor subclass. In this pattern, the addition of a new operation requires a new method implementation for every type of node be placed in a single Visitor subclass which is rather simple. Contrariwise, the addition of a new type of node (leaf class) requires all

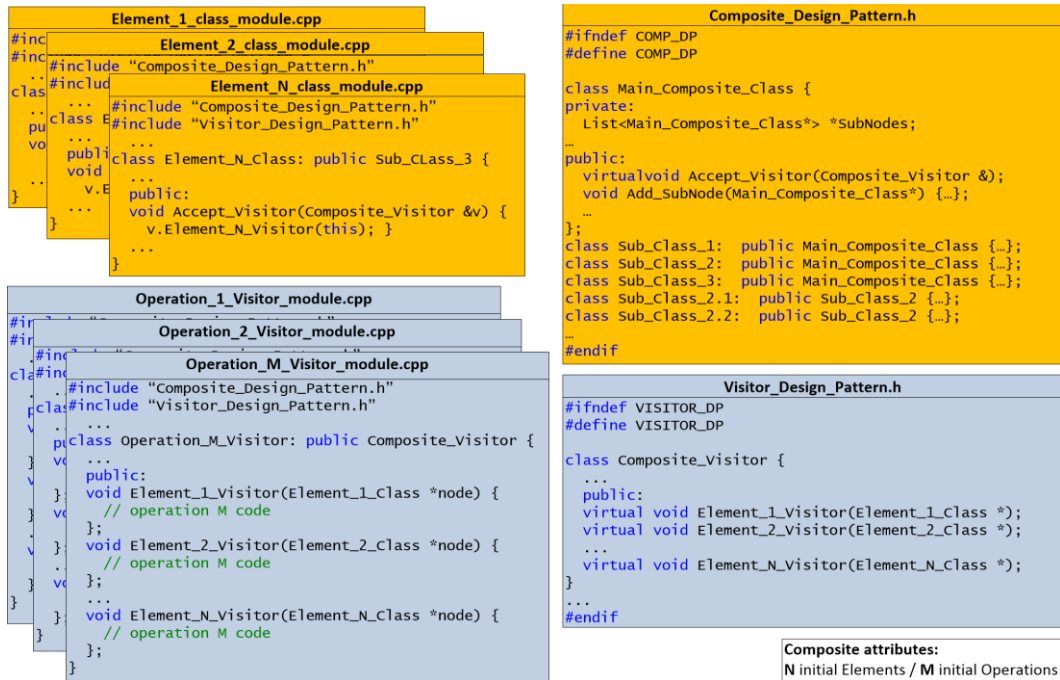


Figure 3-6: Typical code example based on Composite and Visitor design patterns (CP, VP).

new methods (for every node type) be placed in different Visitor subclasses (operation type). In contrast with IBI, it is obvious that this pattern makes adding new operations easier (Gamma et al., 1994). This is also truth in case of modifying an existing operation thanks to concentration (locality) of interventions. A typical code example based on CP and VP patterns is presented in Figure 3-6.

In practice, VP approach rearranges the methods of all distinct operations from CP sub-classes into the new visitor sub-classes. This kind of rearrangement is a typical architectural tactic (Bass et al., 2012) towards the improvement of maintainability which tries to increase cohesion of VP sub-classes considering that the operations will be more prone to changes during maintenance process. However, at the same time, the cohesion of CP sub-classes is decreased while their coupling degree with VP sub-classes is increased making this design variation less maintainable assuming that the elements will be more prone to changes during maintenance process. Usually, the application of an architectural tactic is subject to several trade-offs of conflicting pro and cons with regards to maintainability, mostly depending on the anticipated type of future events.

There are many other positive and negative characteristics of VP which are fully presented in (Gamma et al., 1994). This study mainly focuses on those characteristics which have a direct impact on the quality characteristics of maintainability and changeability.

### 3.4 Software Quality Measures

#### 3.4.1 Quality Measures

To evaluate design pattern combinations at an early stage, the proposed approach evaluates quality characteristics before code development based on the initial structure and attributes of the used design pattern combination. At this early stage, there is no source code for analysis except a design concept or a combination of design patterns, usually in the form of UML diagrams. However, even at this early stage, many quality characteristics of object-oriented design pattern combinations can be approximated.

The proposed model focuses on a major software quality aspect which in the case of CP, VP and IBI is the lowest required effort for extensions or modifications during software maintenance. In this way, the proposed model captures and eventually measures maintainability of specific design pattern combinations. Measuring maintainability requires that proper attributes and quantifiable metrics should be defined. Using these metrics, structural properties, which make a design pattern combination more maintainable and adaptable than one other, can be identified.

Maintainability as a software quality characteristic (defined in (ISO/IEC 25010, 2011; ISO/IEC/IEEE 24765, 2010)), is related to the ease of future modifications of software code and therefore is considered as a very important quality attribute requirement. The proposed approach mainly focuses on predicting software maintainability and its subdivisions, changeability, and modifiability.

### 3.4.2 Expressing Software Maintainability

A vast number of metrics, models, methods and case studies that focus on software maintainability definition, evaluation, and prediction are available in the literature. Many of them have been referenced in Section 2. In this subsection, the interrelation between different concepts/aspects of software maintainability assessment through existing literature, are discussed.

In (Bidve & Sarasu, 2016), the coupling degree among classes is related to maintainability assessment through existing object-oriented metrics. In (Aloysius & Arockiam, 2013), the code complexity is related to maintenance effort prediction through three existing object-oriented cognitive complexity metrics. In (Victor R. Basili et al., 1996), design complexity is related to fault pronounce and reliability assessment through existing CK (Chidamber & Kemerer, 1994) metrics. In (R. Subramanyam & Krishnan, 2003), code complexity is related to fault pronounce using existing CK and MOOD metrics. In (Aversano, Cerulo, & Di Penta, 2009), the number of defects in design pattern classes is related to scattering degree of their induced crosscutting concerns and maintainability aspect. In (Canfora, Cerulo, Di Penta, & Pacilio, 2010), the source code complexity is related to code disorganization or software entropy concept which is measured by using source code entropy. Software entropy was first introduced by (Jacobson, 1992), and according to (Hassan, 2009), it is directly related with the intuition that developers will have harder work keeping track of changes that are performed across many source files or any other code unit such as classes, methods, functions, code chunks respectively. In (Bakota et al., 2012), the code disorder (entropy) is related to maintainability through a probabilistic model based on software entropy concept. Also, in (Bakota et al., 2011), a probabilistic approach for computing high-level quality characteristic such as maintainability is presented. In (Heitlager et al., 2007a), a practical model for measuring maintainability is presented through which source code metrics can be related to quality characteristics (as defined in (ISO/IEC 25010, 2011)) such as maintainability. According to this model, a source code measure (such as cyclomatic complexity) indicates one or more source code properties (such as code complexity) which in turn influence system quality characteristics (such as maintainability). In (Riaz, Mendes, & Tempero, 2009b) review, it is concluded that maintainability and its sub-characteristics (as defined in (ISO/IEC 25010, 2011; ISO/IEC 25023, 2016)) are closely related to complexity and software size. Therefore, models using complexity and size metrics as predictors may be likely to be equally applicable to any maintainability sub-characteristic. By combining evidence of earlier works, an enhanced conceptual diagram for measuring maintainability is derived and presented in Figure 3-7. In this diagram, the interrelations between different concepts and aspects of software maintainability assessment are indicated in a more clear and classified way.

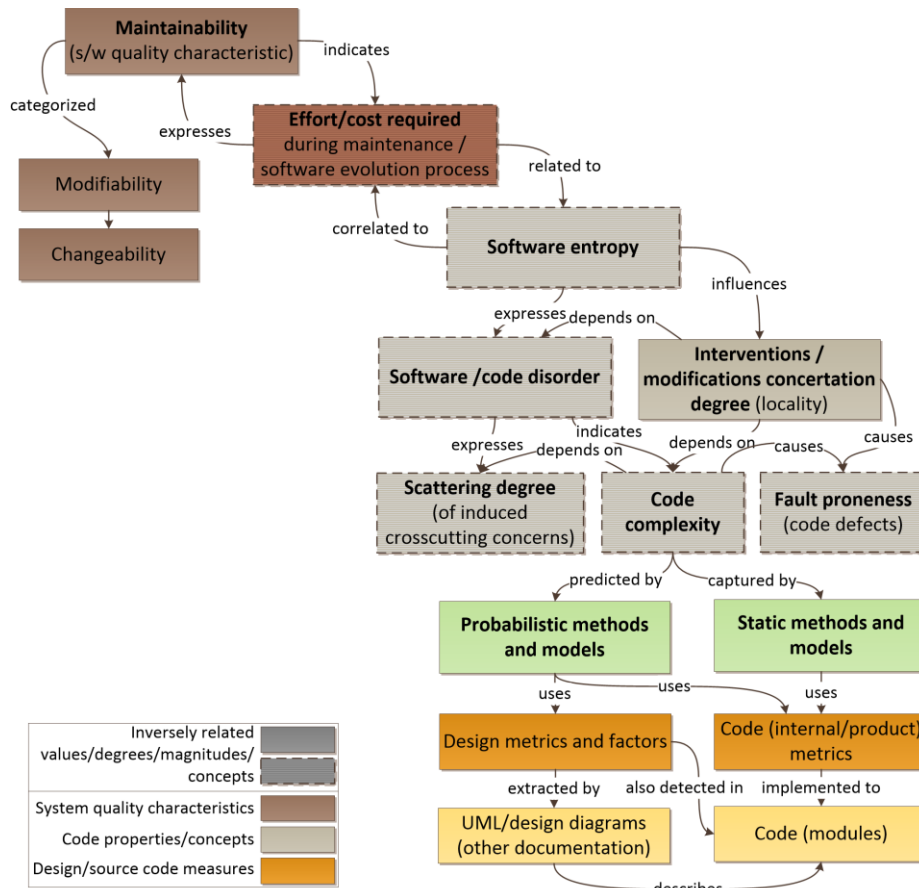


Figure 3-7: Conceptual diagram of interrelation of code properties regarding software maintainability assessment.

Figure 3-7 outlines the maintainability assessment process, starting from low-level code measures, passing to code properties, and ending to effort/cost estimation which expresses maintainability, as suggested by (Heitlager et al., 2007a). In nearly all the works in the literature, probabilistic (Bakota et al., 2011, 2012) and static (Aloysius & Arockiam, 2012; Bandi et al., 2003; Bansiya & Davis, 2002; Dagpinar & Jahnke, 2003; Fioravanti & Nesi, 2001; Rizvi & Khan, 2010) methods and models usually use low-level code measures (Bansiya & Davis, 2002; Chidamber & Kemerer, 1994; Li, 1998; Lorenz & Kidd, 1994) to capture code properties and through which to predict or estimate the required effort and relevant cost during maintenance process. Furthermore, low-level code metrics are usually used for capturing or estimating more concrete properties such as code complexity and crosscutting concerns (scattering degree). More abstract code properties like disorder, entropy and interventions locality are usually estimated indirectly through other more solid properties and/or through probabilistic models.

In the case of design pattern comparisons at an early stage before code development, there is no code for applying code measures except of an initial design structure. Thus, there is a need for a special metric that captures the behavior of specific design pattern combinations' structure during maintenance (evolution) process. This is an intelligent task and apparently requires human intervention. Traditional low-level code metrics (Chidamber & Kemerer, 1994; Li, 1998), cohesion metrics (Kayarvizhy, Kanmani, & Uthariaraj, 2013), ISO metrics (ISO/IEC 25023, 2016), etc. do not seem to capture structural behavior and evolution. This is a problem frequently leading to general probabilistic models which deliver suboptimal estimations and assessments. For example, in the (Bakota et al., 2012) model, maintainability (software entropy/disorder) assessment is based only on the size of the source code at a specific time, measured in lines of code

(LOC). Using the LOC metric in a probabilistic model is rather a rough magnitude although authors (Bakota et al., 2012) state that their proposed model predicts development cost with high accuracy.

Furthermore, design pattern comparison and analysis do not require an absolute maintenance cost prediction regarding available resource e.g. labor cost, operational expenditures, capital costs, etc., like (Bakota et al., 2012) model does. Instead, design pattern comparison can be achieved through a proportionally equivalent prediction of maintenance effort for each implementation alternative under comparison, even if these predictions are subjective concerning absolute cost magnitudes. Moreover, with no absolute cost prediction, other factors like change rate over time are unnecessary and can also be omitted. Additionally, the necessary maintenance cost for the actual code can be considered as neutral. This is because the business logic or actual code is the same for all implementations under comparison and, thus the required effort for its maintenance can be ignored.

The proposed model, free from the need for absolute cost predictions, is concentrated on capturing the locality degree of the required interventions for the anticipated changes that will take place during software maintenance process. Although locality is an abstract property, hard to be captured by common measures, it is selected due to peculiar case related to comparison of design patterns. The key concept is behind the knowledge about design pattern behavior and evolution during maintenance process. Knowing the individual pattern architectures and their behavior, the number and perhaps the extend of future interventions is possible to be determined in a far more precise way than in similar probabilistic models such as in (Bakota et al., 2012). Furthermore, the locality of those interventions is also possible to be determined in a precise way regarding their allocation in separate code units such as classes, modules, and files. Thus, the concept of interventions' locality, is directly related with the intuition that developers will have harder work keeping track of changes that are performed across many source files or any other code unit (Hassan, 2009). Furthermore, interventions' locality property is inversely related to software entropy and code disorder properties as indicated in Figure 3-7. As (Bakota et al., 2012) claims, higher code disorder requires more effort for modifications to be performed, and so maintainability can be interpreted as a measure of the disorder (entropy) of the source code. Moreover, the locality should be referred to class instead to file (code) units since classes' code normally exists in separate files or it is widespread in large files.

Summarizing, the proposed model captures software maintainability by estimating the required effort during maintenance process based on specific design patterns' behavior, focusing on two major (structural) measurement aspects: a) the estimated size of future interventions through the number of methods (code units) under maintenance, and b) the locality of those interventions through the number of classes (code units) that are affected.

### 3.4.3 Deriving Maintainability Effort

To be meaningful, a metric must provide a numerical value to a software attribute that is of genuine interest. The proposed approach uses measurement theory (Baker et al., 1990; N. Fenton & Melton, 1990; Melton, Gustafson, Bieman, & Baker, 1990) and related work (Offutt, Abdurazik, & Schach, 2008) for this purpose.

The distinct design pattern combinations (or implementations or design alternatives) under comparison are defined in the first step. Referring to the significant problem of recursive hierarchies of part-whole representations in subsection 3.3, the proposed model focuses on two design pattern combinations:

- CVP : the combination of CP with VP
- CIBI : the combination of CP with IBI

To measure maintainability, the basic or major maintenance scenarios have been described and analyzed with respect to the modification cost. The characteristics and the criteria of major maintenance scenarios in the context of the proposed method are discussed in subsection 3.4.5. The proposed approach analyzes the two most important maintenance scenarios for each design combination:

- $n_e$  : adding or modifying a new node type (or element) on the CP
- $n_p$  : adding or modifying a new operation (or process) on the VP or on the CP through IBI

The proposed model focuses on the additions and modifications for the operations and elements. These are the most relevant and anticipated maintenance scenarios since CVP and CIBI exclusively target on the efficient implementation of a set of operations over the composition's elements. Normally, during maintenance, developers are called to manage additions and modifications of operations or elements. Moreover, these are exactly the scenarios for which CVP and CIBI have different and conflicting quality characteristics regarding their maintainability perspective as analyzed in subsection 3.3.2. Although the proposed model is described in terms of additions and modifications for operations and elements, other types of interventions such as deletions are considered as similar to the above scenarios. For example, removing an operation from a design pattern combination has the same impact (in terms of number and locality of interventions) like adding an operation. Normally, during e.g. an operation's deletion, developers should modify the software at the same locations that were modified during its addition. In general, during deletions, developers should reversely perform almost the same interventions as for adding or modifying operations and elements which are extensively described in subsection 3.3.2.

New metrics that quantify specific properties, related to maintainability, have been derived based on earlier related metrics such as Maintainability Index (Coleman, Ash, Lowther, & Oman, 1994; Oman & Hagemester, 1994), SIG Maintainability Model (Heitlager et al., 2007a), Evolution Complexity (Eden & Mens, 2006; Tom Mens & Eden, 2005) and Computational Complexity or Complexity of Maintenance as defined in (Hills et al., 2011).

To measure maintainability or the maintenance effort for each of the maintenance scenarios, the proposed approach derives two basic and simple aspects of measures as analyzed in subsection 3.4.2:

- $a_m$  : the number of modifications or interventions on distinct methods which is a straightforward measure representing the direct estimation of maintenance cost or effort (method aspect), and
- $a_c$  : the number of modifications or interventions on distinct classes which is a measure that captures a non-obvious aspect of maintenance cost or effort relate to the locality degree of previous interventions (class aspect)

To capture maintainability, the number of distinct classes that will be modified (or added) should be included in the measured maintenance cost or effort. In this way, the maintenance cost does not only capture the number of future modifications or additions on distinct methods but also capture a major quality property related to the locality of these adaptations regarding their place in separate classes as analyzed in subsection 3.4.2. Thus, class interventions or locality degree is a non-obvious measurement aspect since it can be revealed only through analysis of the internal structure and behavior of the design pattern combination under comparison. This measure may also capture other quality characteristics of the software such as extensibility and modularity since high locality of maintenance implies minor interventions at the source code including module level.

In general, maintainability ensures that future software modifications will be implemented in the same or in as few as possible classes or even in the same module of

code. This makes the maintenance (including debugging) process easier for the developers because future modifications during maintenance should be performed only in specific classes and/or modules. Moreover, the compilation of the source code usually becomes more efficient since only some of their modules will be re-compiled.

Hence, less maintenance cost means better quality characteristics for maintainability, changeability, and adaptability. Object-oriented design patterns are especially used to improve adaptability since patterns generally increase the complexity of an initial design in order to ease future enhancements. It is important that the proposed metrics are useful and can be applied early even if any of the existing Expression Problem solutions is adopted at a later stage. In these cases, although extensions based on Expression Problem solutions leave the initial code module unchanged, the maintenance cost for these extensions persist, since design pattern architecture remains either the extensions are performed in the same or separate code modules.

It is noteworthy that the proposed metrics target on a consistent comparison of maintenance cost between different design pattern combinations and not on accurate cost assessment which is subjective as analyzed in subsection 3.4.2.

The metric Structural Maintenance Cost is defined as follows.

*Definition of Structural Maintenance Cost (SMC):* represents the effort required to adjust a specific design pattern combination (implementation or artifact) in the event of a particular maintenance scenario (stimulus) from a specific structural aspect, expressed in terms of number of applied interventions and number of affected code units.

From the architectural design perspective, the SMC metric corresponds to the Response Measure that quantitatively expresses the Response (or the required changes) of an Artifact (design implementation) in the event of a particular Stimulus (maintenance scenario) during maintenance, as discussed in subsection 3.2.

The definitions, terminology, and notation of the section are summarized in Table 3-1.

Table 3-1: Maintenance Cost Terminology and Notation

Terminology	Range / Values
Design pattern combination (implementation or artifact)	$D = \{ CVP, CIBI \}$
Maintenance Scenario or Stimulus	$S = \{ n_e, n_p \}$
Structural Aspect	$A = \{ a_m, a_c \}$
Structural Maintenance Cost (SMC) metric	$c_m^{S,A} = (S, D, A)$

For example, the notation  $c_m^{S,A} = (n_e, CVP, a_c)$  corresponds to the maintenance cost or effort that is required to adjust CP and VP combination in the event of a new element from class aspect. Alternatively, refers to the number of modifications on distinct classes which are necessary in order an element be added on CP and VP combination. The proposed model uses the metric of structural maintenance cost to evaluate and compute the total modification cost for each extension scenario and for each design combination as analyzed next.

#### 3.4.4 Considerations upon Other Quality Characteristics and Properties

In this subsection, we answer the question whether the proposed measurement approach is affected by or affects other quality characteristic of the software such as reusability, complexity, performance, or issues like resourcing and debugging. Furthermore, the characteristics of different types of maintenance activities are discussed.

*Reusability and Complexity:* At an early stage of software development there is no source code for evaluating and analyzing its complexity. Thus, regarding the embedded complexity of the used design patterns, it is an acceptable cost in exchange for other benefits or advantages such as reusability and extensibility. Thus, design patterns' complexity and reusability can be considered as neutral characteristics at this early stage of evaluation. However, the number of interventions and their locality degree, captured by the proposed metric, conceive an aspect of reusability since high reusability degree means,

in a sense, that the required interventions during maintenance should be allocated in a limited number of code units or entities.

*Performance and Resourcing:* Focusing on the inner performance and efficiency of design patterns, two major aspects arises: a) run-time performance and b) memory consumption. Although VP and IBI are based on recursive calls of its operations, VP has lower performance and increased memory consumption due to the double dispatching on visitor calls. More specifically, memory consumption of VP is higher than IBI, because VP double-dispatching causes double calls when a visitor method is engaged. This extra memory cost is unavoidable and is proportional to the depth of recursive calls which in this case is the depth of the (composition) object tree. Since usually, the depth of object trees has a logarithmic order of magnitude on the number of nodes, the double-dispatching memory cost is considered as negligible. Respectively, run-time performance of VP is lower than IBI, again due to VP double-dispatching. During run-time, each method call consumes a standard process time. Although this extra process time is intuitively small, in this early stage, it is almost impossible to be compared with the process time of the actual (business logic) code. However, in (Hills et al., 2011), the run-time results of four case studies showed that in most cases the performance differences between VP and Interpreter design pattern (IBI) is not substantial. Thus, based on (Hills et al., 2011) results, the run-time performance cost for VP is also considered as negligible.

*Debugging issues:* A good design pattern combination should reduce debugging effort during software maintenance. However, the low debugging effort is ensured through maintainability and low complexity discussed above.

### 3.4.5 Characteristics and Criteria of Major Maintenance Scenarios

The main goal of the proposed model is to derive metrics as simple as possible and oriented to specific and fundamental quality characteristics of the used design patterns. The selection of proper types of maintenance scenarios (i.e., new element and new operation) is a critical process since defines the outcome of the measurement approach. In general, the method is focused on those maintenance activities or events that are aligned with the architectural advantages or disadvantages of the used design patterns and the pursued quality requirement of maintainability which is the primary selection criterion among design alternatives under evaluation and comparison. However, maintenance activities or events varies significantly in respect to many conflicting properties and characteristics. A map that represents the tradeoffs between various types of maintenance events is presented in Figure 3-8.

During maintenance, the software specifications change to adapt software functionality to the new users' requirements. Such descriptions and requirements are usually referred and translated to specific maintenance events which are oriented to user's perspective and minor or specific functional requirements, thus their potential number of types are huge and difficult to be modeled, as illustrated in Figure 3-8. To facilitate the modelling approach, specific events should be grouped or classified in more general families or classes of resembling events that have similar properties and likelihood to occur during maintenance. Thus, general types of events are fewer in number (possible overlapped with specific events) and have higher probability to occur. Eventually, the analysis should be focused on even more broad types of anticipated events oriented to the architectural structure and attributes of the used design patterns, their design attributes, and the pursued quality attribute requirement. Thus, basic, or major types of anticipated events are even fewer in number (possible overlapped with general events) and have even higher repeatability and probability to occur.

Software engineers should have in mind that the goal of the method is to conclude on those types of maintenance events for which their occurrence probability could be estimated based on the characteristics of each specific instance of the general problem under study. Furthermore, the engaged design patterns demonstrate several pro and cons



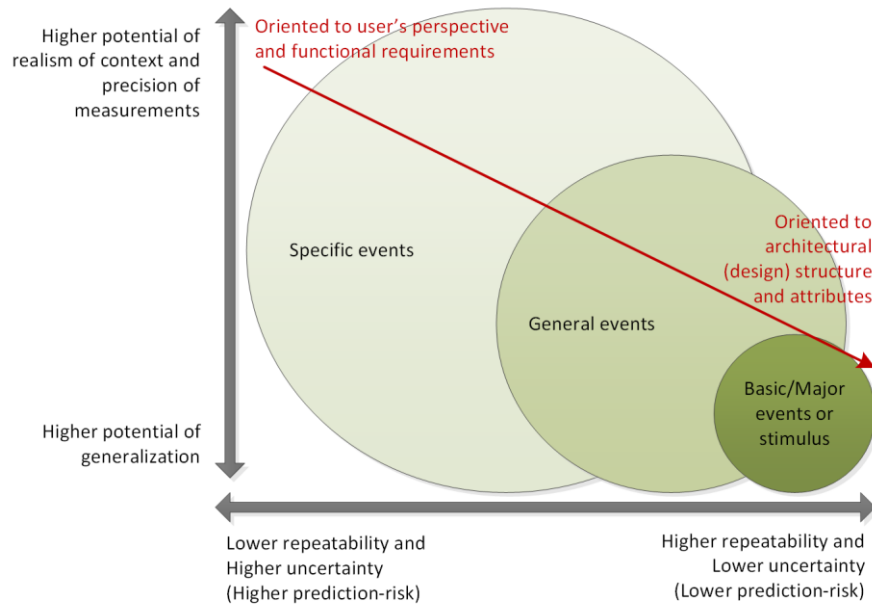


Figure 3-8: Tradeoffs between various types of arriving events during maintenance.

for different types of anticipated events during maintenance as discussed in subsection 3.3. These types of events are usually the most proper candidates to be considered as major maintenance scenarios. Design patterns implementations that are more suited to support the functional purposes of an application tend to be more prone to change during maintenance process (Aversano, Canfora, Cerulo, Del Grosso, & Di Penta, 2007). This modelling approach is in accordance with the ‘design for change’ principle (David Lorge Parnas, 1994), and the general architectural design principles in software engineering (Bass et al., 2012).

Examples of descriptions for each class of events in Figure 3-8 may be:

- *SPEFIFIC/MINOR* - Related to minor specifications and user’s requirements, e.g.: enhancement for dynamic resizing of selected block, including controls and other details.
- *GENERAL* - Related to broader specifications and user’s requirements, e.g.: new logical entity for block of design elements, new redesign process of all elements of block, new control elements in top and popup menus.
- *BASIC/MAJOR* - Related to the principal logical entities of the used Design Patterns, e.g.: new part-whole element in Composite Design Pattern, new recursive redesign operation in Visitor Design Pattern.

Nevertheless, each class of anticipated events have different and conflicting characteristic, while their tradeoffs are visualized as distinct dimensions in Figure 3-8. Moving from specific to major events, their repeatability and predictability increases, thus lowering the uncertainty degree and the risk to predict their occurrence probability. At the same time, the potential of generalization is increased too since broader type of events are more likely to be applied in a wider spectrum of possible instances of the general problem under study. However, the potential of realism of context and precision of measurements are decreased since broader type of events are away from real and specific functionalities while the quantification of their effect becomes more abstract and difficult to be measured in a precise manner, as suggested in (Stol & Fitzgerald, 2018).

In principle, an anticipated event (or class of resembling sub-activities) is characterized as major maintenance scenario or stimulus when fulfils the following criteria:

- has significant impact concerning the pursued quality attribute (i.e., maintainability). This criterion is satisfied either when:
  - its impact in terms of required effort significantly differs per design alternative, otherwise its effect will be common for all design alternatives and, thus neutral concerning the comparison and decision-making process. For example, a minor activity with the same effect across all design alternatives does not increase the reliability of the model concerning the comparison outcome while it adds unnecessary complexity, OR
  - it affects or changes the principal design attributes of the engaged design patterns based on which its effect or the effect of other major events in terms of required effort is computed,
- is neither too abstract nor too specific allowing its application on the early design stage before code development (i.e., encompasses various resembling sub-activities or changes such as adding, updating, and debugging concerning the maintenance of a discrete family of design elements with common characteristics),
- has recurring nature or considerable possibility to repeatedly occur during maintenance. For example, the probability of an extremely rare event is very difficult to be assessed while its impact (even if occur) would be negligible against the effect of the other more frequent events, and thus, it adds unnecessary complexity without significant benefits concerning the comparison outcome.

## 3.5 Analysis of Method

### 3.5.1 Deriving Problem's Characteristics and Attributes

Each design pattern combination has some attributes or characteristics which declare its initial state. This initial state can be an existing implementation under maintenance or the output of a code generation tool under adjustment. A maintenance scenario changes or shifts an initial implementation to a new state by updating its characteristics. To quantify the effort for each maintenance scenario, the proposed approach quantitatively derives these characteristics of the problem. Thus, for CVP and CIBI design alternatives, two major design characteristics or design attributes are derived:

- $N$  : the number of outer (leaf) classes of a Composite structure CP or the number of distinct elements (objects) that can be instantiated from a Composite structure, as indicated in Figure 3-9.
- $M$  : the number of operations that are performed on the objects (or element) of a Composite structure, as indicated in Figure 3-9.

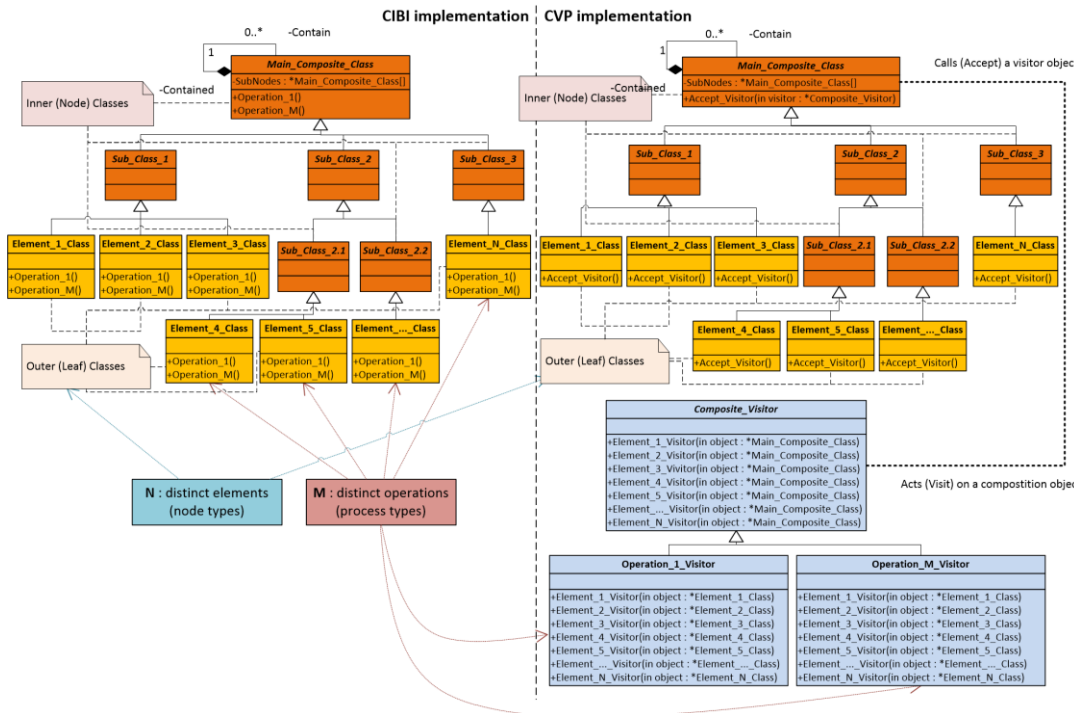


Figure 3-9: Design attributes of typical CIBI and CVP implementations.

For example, the  $n_e$  (add new element) maintenance scenario, increases the number of initial elements ( $N$ ) by one. Similarly, the  $n_p$  (add new operation) maintenance scenario, increases the number of initial operations ( $M$ ) by one. The mathematical notation and symbols of the analysis are presented in Table 3-2 and explained next.

Table 3-2: Maintenance Cost Terminology and Notation

Notation	Description
$N$	The number of outer (leaf) classes of a Composite structure OR The number of distinct elements (objects) that can be instantiated (created) from a Composite structure
$M$	The number of operations that are performed on the objects (or elements) of a Composite structure
$\mu$	$\mu = \frac{M}{N} \leftrightarrow M = \mu \cdot N$
$p_{nE}$	The probability for a new structure Element $p_{nE} = 1 - p_{nP}$
$p_{nP}$	The probability for a new Process/Operation $p_{nP} = 1 - p_{nE}$
$z$	$z = \left\lceil \frac{p_{nE}}{1 - p_{nE}} \right\rceil = \left\lceil \frac{p_{nE}}{p_{nP}} \right\rceil$ ( $\lceil x \rceil$ means roundup $x$ )
$\dot{z}$	$\dot{z} = \left\lceil \frac{1 - p_{nE}}{p_{nE}} \right\rceil = \left\lceil \frac{p_{nP}}{p_{nE}} \right\rceil$
$\lambda$	The number of future additions / modifications (new type node/element or new process/operation)

### 3.5.2 Asymptotic Evaluation of Structural Maintenance Cost

In this subsection, the previously defined SMC metric is used for evaluating and computing the maintenance effort or cost for each extension scenario and each design combination, based on the specific characteristic or attributes of the problem.

The asymptotic notation  $O(g(x))$  denotes the worst case or upper bound evaluation as analyzed in (Cormen, Leiserson, Rivest, & Stein, 2009). In the context of the proposed work, the asymptotic notation is used to describe an approximation of the maintenance effort after a specific action takes place. Moreover, this notation provides a more general bound for a large number of initial elements and simpler mathematical formulas for further analysis and computations. However, assuming that the conducted analysis is assisted by software tools (e.g., MATLAB), the asymptotic notation can be bypassed.

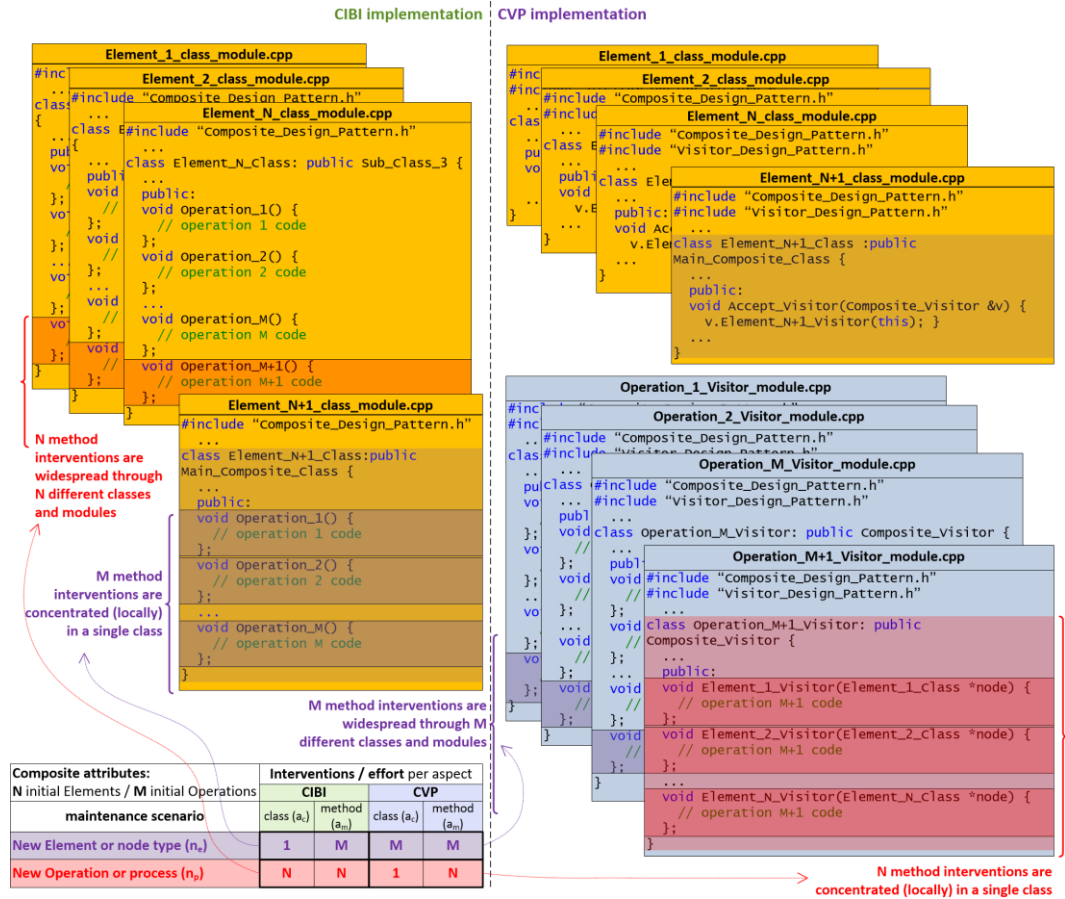


Figure 3-10: Typical code example after implementation of one new element and one new operation scenarios for CIBI and CVP implementation alternatives.

A visual representation of the interventions’ impact on a typical code example regarding each extension scenario, measure aspect, and combination under comparison is presented in Figure 3-10. More specific, the implementations of one new element (n<sub>e</sub>) and one new operation or process (n<sub>p</sub>) scenarios are simultaneously presented for both design pattern combinations (CVP, CIBI).

Based on SMC definition and the specific characteristics of the problem, four distinct cases are derived, one for each maintenance scenario and design combination.

*New element on CP and VP:* When a new element should be added by the developer, a wide range of modifications are needed. More specifically, for every new element, a new subclass definition in the CP with a new (accept) method are needed. Also, M new methods (operation code for the new element), one in every existing visitor (operation) subclass, should be created. Totally, 1+M method modifications into 1+M distinct classes are necessary to be made as indicated in Figure 3-10.

The asymptotic maintenance cost for adding a new element onto CP and VP combination considering the effect on distinct classes is given by the Equation (3-1).

$$c_m^{S,A}(n_e, CVP, a_c) = O(1 + M) = M \quad (3-1)$$

The asymptotic maintenance cost for adding a new element onto CP and VP combination considering the effect on distinct methods is given by the Equation (3-2).

$$c_m^{S,A}(n_e, CVP, a_m) = O(1 + M) = M \quad (3-2)$$

*New operation on CP and VP:* When a new operation should be added by the developer, a smaller range of modifications are needed. More specifically, for every new operation, a new method (for every existing element) in a new visitor class (for the new operation) should be created. Totally, N method modifications into one new class are necessary to be made as indicated in Figure 3-10.

The asymptotic maintenance cost for adding a new operation onto CP and VP combination considering the effect on distinct classes is given by the Equation (3-3).

$$c_m^{S,A}(n_p, CVP, a_c) = O(1) = 1 \quad (3-3)$$

The asymptotic maintenance cost for adding a new operation onto CP and VP combination considering the effect on distinct methods is given by the Equation (3-4).

$$c_m^{S,A}(n_p, CVP, a_m) = O(N) = N \quad (3-4)$$

*New element on CP and IBI:* When a new element should be added by the developer, a new subclass definition as well as new methods (one for every operation) should be created. Totally, M method modifications into one new class are necessary to be made as indicated in Figure 3-10.

The asymptotic maintenance cost for adding a new element onto CP and IBI combination considering the effect on distinct classes is given by the Equation (3-5).

$$c_m^{S,A}(n_e, CIBI, a_c) = O(1) = 1 \quad (3-5)$$

The asymptotic maintenance cost for adding a new element onto CP and IBI combination considering the effect on distinct methods is given by the Equation (3-6).

$$c_m^{S,A}(n_e, CIBI, a_m) = O(M) = M \quad (3-6)$$

*New operation on CP and IBI:* When a new operation should be added by the developer, one new method (for the new operation) in every existing element (type) class, should be created. Totally, N method modifications into N distinct classes are necessary to be made as indicated in Figure 3-10.

The asymptotic maintenance cost for adding a new operation onto CP and IBI combination considering the effect on distinct classes is given by the Equation (3-7).

$$c_m^{S,A}(n_p, CIBI, a_c) = O(1 + N) = N \quad (3-7)$$

The asymptotic maintenance cost for adding a new operation onto CP and IBI combination considering the effect on distinct methods is given by the Equation (3-8).

$$c_m^{S,A}(n_p, CIBI, a_m) = O(1 + N) = N \quad (3-8)$$

### 3.5.3 Merging Structural Maintenance Cost

In this subsection, the previously defined asymptotic maintenance costs are merged to represent the structural maintenance cost in a more compact and manageable way. Based on the structural maintenance cost of four previous cases, the maintenance cost is merged and defined as follows.

*New element on CP and VP:* The maintenance cost for adding a new element onto CP and VP combination considering all aspects is given by the Equation (3-9).

$$c_m^S(n_e, CVP) = \sum_{\forall A \in \{a_c, a_m\}} c_m^{S,A}(n_e, CVP, A) = 2M \quad (3-9)$$

*New operation on CP and VP:* The maintenance cost for adding a new operation onto CP and VP combination considering all aspects is given by the Equation (3-10).

$$c_m^S(n_p, CVP) = \sum_{\forall A \in \{a_c, a_m\}} c_m^{S,A}(n_p, CVP, A) = N + 1 \quad (3-10)$$

*New element on CP and IBI:* The maintenance cost for adding a new element onto CP and IBI combination considering all aspects is given by the Equation (3-11).

$$c_m^S(n_e, CIBI) = \sum_{\forall A \in \{a_c, a_m\}} c_m^{S,A}(n_e, CIBI, A) = 1 + M \quad (3-11)$$

*New operation on CP and IBI:* The maintenance cost for adding a new operation onto CP and IBI combination considering all aspects is given by the Equation (3-12).

$$c_m^S(n_p, CIBI) = \sum_{\forall A \in \{a_c, a_m\}} c_m^{S,A}(n_p, CIBI, A) = 2N \quad (3-12)$$

In Equations (3-9)-(3-12), individual costs for each aspect have been added by equal weights. Considering concentration degree as extra (penalty of) required effort, the proposed metric implies that this extra effort amount has equal weight to the number of method interventions. Furthermore, this additional effort could be captured with different weights by introducing a new factor over its magnitude. This is a very interesting perspective which discussed in subsection 3.9.2 as an extension example of the method.

### 3.5.4 Combining Maintenance Cost

The proposed model is further generalized through the combination of different maintenance scenarios  $n_e$  and  $n_p$ . This requires the introduction of probability analysis by engaging different maintenance scenarios. This is a step over the usual analysis level in comparison to the related existing approaches such as (Hills et al., 2011; Tom Mens & Eden, 2005). This provides the proposed model with greater flexibility and a wider field for further analysis and conclusions.

Based on specific characteristics of the design pattern combinations, two symmetrical probability factors are derived.

- $p_{ne}$  : the probability for a new element against the probability for a new operation
- $p_{np}$  : the probability for a new operation (process) against the probability for a new element

For example,  $p_{ne} = 0.8$  means that the probability for a new element is 80% against 20% probability ( $p_{np} = 1 - p_{ne} = 0.2$ ) for a new operation. Developers can specify these probabilities for a specific problem since system's specifications and developers' experience often offer a reliable prediction about their values. It is important that based on probability theory (Jaynes, 2003), the probability factors of independent events<sup>1</sup> should be complementary (as mutually exclusive). Also, the sum<sup>2</sup> of the probability mass function<sup>3</sup> for all events is equal to 1. Thus, for the case of two probability factors, they should be symmetrical. Furthermore, different maintenance scenarios (such as  $n_e$  and  $n_p$ ) could not be additive. Normally, new element and new operation are independent event types. A simultaneous addition of different event types is not manageable by the developers. In practice each event type usually begins after the completion of one other event type.

<sup>1</sup> Two events A, B are independent if and only if one's realization does not affect the probability of the other or  $P(A/B)=P(A)$ .

<sup>2</sup> The sum of the probability mass function is defined as  $\sum_{i=1}^m f_X(x_i) = 1$  where  $f_X(x_i) \geq 0$

<sup>3</sup> The probability mass function for a series of  $X=\{x_1, x_2, \dots, x_m\}$  distinct events is defined as  $f_X(x_i) = P(X=x_i)$

Based on structural maintenance cost and probability factors, the combined maintenance costs for each design pattern combination are derived as follows:

*Maintenance cost of CP and VP:* The maintenance cost for CP and VP combination based on probabilities is given by the Equation (3-13).

$$\begin{aligned} c_m(p_{ne}, CVP) &= p_{ne} \cdot c_m^S(n_e, CVP) + (1 - p_{ne}) \cdot c_m^S(n_p, CVP) \\ &= N(2\mu p_{ne} - p_{ne} + 1) + 1 - p_{ne} \end{aligned} \quad (3-13)$$

*Maintenance cost of CP and IBI:* The maintenance cost for CP and IBI combination based on probabilities is given by Equation (3-14).

$$\begin{aligned} c_m(p_{ne}, CIBI) &= p_{ne} \cdot c_m^S(n_e, CIBI) + (1 - p_{ne}) \cdot c_m^S(n_p, CIBI) \\ &= N(\mu p_{ne} - 2p_{ne} + 2) + p_{ne} \end{aligned} \quad (3-14)$$

Factor  $\mu = M/N$ , used for simpler mathematical representation and further analysis.

### 3.5.5 Summarizing Structural Maintenance Cost

The previously defined equations for structural maintenance cost evaluation are summarized in Table 3-3.

Table 3-3: Asymptotic Evaluation of Structural Maintenance Cost on Inheritance-Based Implementation and Visitor Design Pattern

Eq. 3-	Condition		Estimation of Structural Maintenance Cost for Design Pattern combination Composite (CP) with	
	Maintenance scenario	Modification effect / aspect	Visitor (CVP)	Inheritance based implementation (CIBI)
1, 5	New Element ( $n_e$ )	Modifications on distinct Classes ( $a_c$ )	$c_m^{SA}(n_e, CVP, a_c) = M$	$c_m^{SA}(n_e, CIBI, a_c) = 1$
3, 7	New Operation ( $n_p$ )		$c_m^{SA}(n_p, CVP, a_c) = 1$	$c_m^{SA}(n_p, CIBI, a_c) = N$
2, 6	New Element ( $n_e$ )	Modifications on distinct Methods ( $a_m$ )	$c_m^{SA}(n_e, CVP, a_m) = M$	$c_m^{SA}(n_e, CIBI, a_m) = M$
4, 8	New Operation ( $n_p$ )		$c_m^{SA}(n_p, CVP, a_m) = N$	$c_m^{SA}(n_p, CIBI, a_m) = N$
9,11	New Element ( $n_e$ )	Modifications on distinct Classes and distinct Methods ( $a_c \wedge a_m$ )	$c_m^S(n_e, CVP) = 2M$	$c_m^S(n_e, CIBI) = M + 1$
10, 12	New Operation ( $n_p$ )		$c_m^S(n_p, CVP) = N + 1$	$c_m^S(n_p, CIBI) = 2N$
13	New Element (possibility $p_{nE}$ ) or new Operation (possibility $p_{nO} = 1 - p_{nE}$ ) / ( $n_e \wedge n_p$ )	Modifications on distinct Classes and distinct Methods ( $a_c \wedge a_m$ )	$c_m(p_{nE}, CVP) = N(2\mu p_{nE} - p_{nE} + 1) + 1 - p_{nE}$	
14			$c_m(p_{nE}, CIBI) = N(\mu p_{nE} - 2p_{nE} + 2) + p_{nE}$	

Note: Eq. column refers to the number of equations

The asymptotic calculations of Equations (3-1)-(3-8) should be applied, having in mind a common confrontation for all individual scenarios. In equations (3-9)-(3-12), the initial basic asymptotic evaluations are merged in two possible maintenance scenarios. These equations confirm the opposite characteristics of IBI and VP discussed in subsection 3.3.2. This early confirmation of Gamma et al. claims can be considered as proof of validity and reliability of the proposed method and measures.

### 3.5.6 Maintenance Process

Design patterns implementations that are more suited to support the functional purposes of an application tend to change more frequently during maintenance process (Aversano, Canfora, Cerulo, Del Grosso, & Di Penta, 2007). Maintenance is a progressive software evolution process that is conceptually described in Figure 3-11. Each maintenance scenario  $S_i$  (such as the introduction of new element or operation) occurs based on its individual probability  $p_i$  and updates a design implementation  $D_j$  (such as CVP or CIBI) requiring a specific amount of effort or maintenance cost ( $c_m^S$ ). Each iteration of the maintenance process updates the characteristics of the attributes of a specific design implementation.

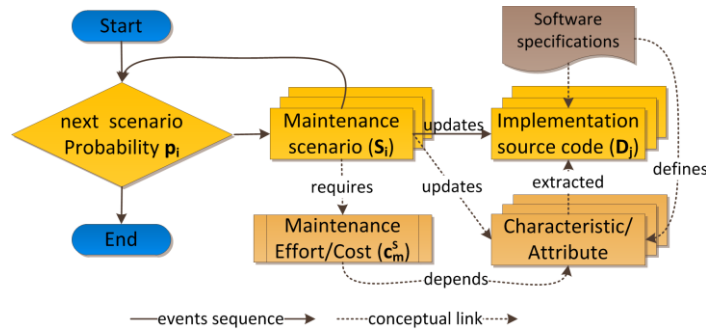


Figure 3-11: Typical conceptual flowchart of software maintenance process.

The analysis of the proposed model focuses on the maintenance process over CVP and CIBI design combinations (implementations). A flowchart that visually describes and summarizes the maintenance process for both design combinations and scenarios is presented in Figure 3-12. Each maintenance scenario ( $n_e$ ,  $n_p$ ) occurs based on its probability ( $p_{n_e}$ ,  $p_{n_p}$ ) and updates a design implementation such as CVP or CIBI requiring a specific amount of effort or maintenance cost,  $c_m^S(S, D)$ . At the same time, each iteration updates the attributes  $N$  and  $M$  of each design implementation.

During this repeated process, the continuously updated values of the design characteristics and attributes affect the intermediate computations of the maintenance cost. An important aspect at this point is the analysis of the behavior of the proposed metric during the maintenance process in a long-term perspective. The proposed approach addresses this aspect as presented in the next subsections.

### 3.5.7 Combined Analysis

In the rest of this subsection, the general case of arbitrary probabilities for maintenance scenarios is analyzed.

In case the probability for a new Composite element has an arbitrary value, equations (3-9)-(3-12) can be combined to return the maintenance costs of equations (3-13) and

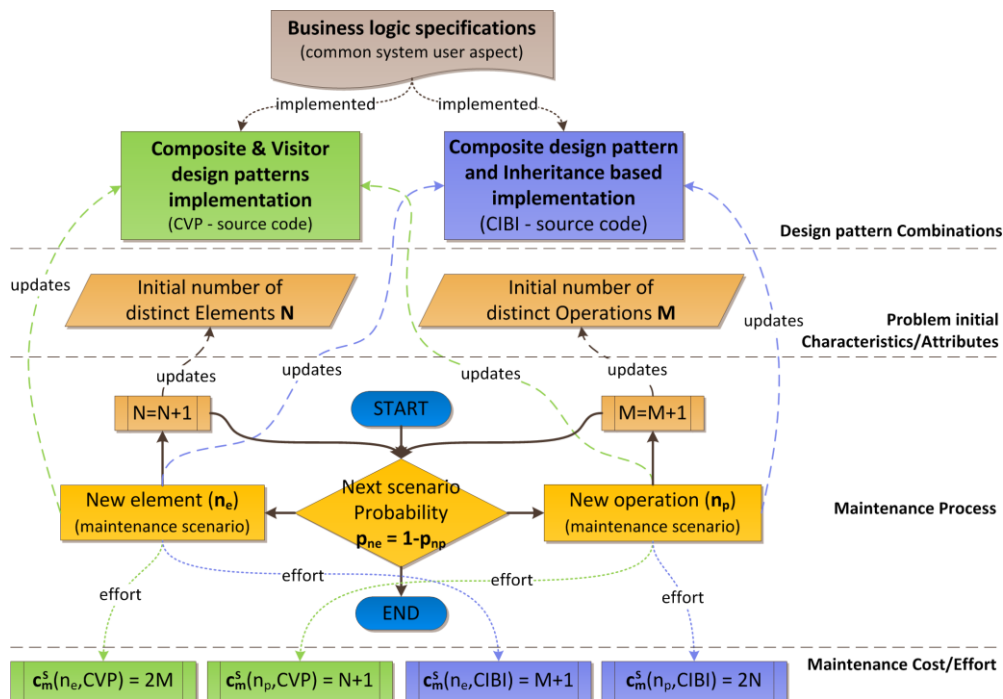


Figure 3-12: Maintenance process flowchart for CVP and CIBI implementations.



(3-14) as summarized in Table 3-3. By replacing  $M=\mu N$ , the factors  $(2\mu p_{nE} - p_{nE} + 1)$  and  $(\mu p_{nE} - 2p_{nE} + 2)$  become crucial. More specifically when  $\mu=(1-p_{nE})/p_{nE}$ , then  $(2\mu p_{nE} - p_{nE} + 1)=(\mu p_{nE} - 2p_{nE} + 2)$ , meaning that IBI and VP have equal maintenance cost. This analysis leads to the general conclusion that the choice between IBI and VP is clear only when factor  $\mu \neq (1-p_{nE})/p_{nE}$ .

Furthermore, when  $p_{nE}=(0..1)$  and assuming a normal arriving pattern between different types of maintenance scenarios, equations (3-13) and (3-14) are valid even in the case of more than one possible future additions. This statement is valid because in all cases, the normal arriving pattern increases  $N$  and  $M$  simultaneously, with a rate related to  $p_{nE}$  and  $p_{nP}$  probabilities. So, if initially  $\mu < (1-p_{nE})/p_{nE}$ , means that always  $\mu < (1-p_{nE})/p_{nE}$ , no matter how many future additions will take place. If the number of future additions becomes large enough, factor  $\mu$  tends to  $(1-p_{nE})/p_{nE}$  (balance case), where two patterns become equal in terms of required effort.

Equation (3-15) shows the limit of factor  $\mu=M/N$  when the number of future additions (factor  $n$ ) tends to infinity for  $p_{nE}, p_{nP}=(0..1)$  and  $p_{nP}=1-p_{nE}$ . Usually, a maximum number between  $n=10$  to  $20$  (future additions) is applied by developers. Equation (3-15) represents a mathematical model for the purpose of theoretical completeness, however, in practice it is less applicable.

$$\lim_{n \rightarrow +\infty} \left( \frac{M_{\text{Initial}} + n(1 - p_{nE})}{N_{\text{Initial}} + n(p_{nE})} \right) = \frac{(1 - p_{nE})}{p_{nE}} = \frac{p_{nP}}{p_{nE}} \quad (3-15)$$

This observation leads to the conclusion that given a Composite structure of  $N$  initial distinct elements and  $M=\mu N$  initial operations, choosing between VP and IBI, is independent of the number of additions (new elements or operations) and only depends on the individual probabilities for each future addition. In the worst case, after many additions, both patterns become equal having similar maintenance costs. Equations (3-13) and (3-14) are general and can substitute the equations (3-9)-(3-12) for  $p_{nE}=1$  or  $p_{nE}=0$ .

### 3.6 Quantitative Analysis

The impact of one future modification is analyzed and discussed in this subsection. First, a basic analysis is presented, followed by combined analysis. A series of graphs are provided representing a visual mapping of the model analysis.

#### 3.6.1 Basic Analysis

For the sake of completeness, in this subsection, a basic analysis for one future addition is presented. The graph in Figure 3-13 presents the maintenance cost of modifications on a Composition for a single future addition according to equations (3-9)-(3-12) as summarized in Table 3-3. The initial number of nodes has been randomly selected to be  $N=25$  since the behavior of linear equations is independent of the number of initial elements ( $N$ ). Axe  $z$  of the graph presents a logical range of factor  $\mu=(0, \dots, 3]$  or a range of  $M=(0, \dots, 3N]$ . Higher values of factor  $\mu$  means more initial distinct operations ( $M$ ) relative to initial distinct elements ( $N$ ).

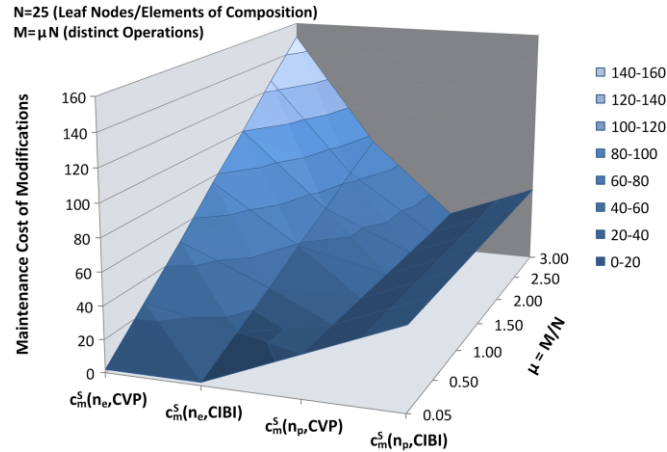


Figure 3-13: Graph of maintenance cost of modifications on a Composition for a single future addition referred to the inheritance-based implementation (IBI) and Visitor design pattern (VP).

The graph in Figure 3-13 confirms the opposite characteristics of IBI and VP discussed in subsection 3.3.2. More specifically, when a steady Composite structure and an extensible set of operations ( $n_p$ ) are addressed, the VP is preferred since it has better (less) maintenance cost independently of  $\mu$  value. Similarly, when a steady set of operations and an extensible Composite structure ( $n_e$ ) are addressed, the IBI is preferred since it has better (less) maintenance cost independently of  $\mu$  value. Also, the statements “*Visitor makes adding new operations easy*” and “*Adding new ConcreteElement classes is hard*” in (Gamma et al., 1994) about CVP are confirmed. This early confirmation of Gamma et al. claims can be considered as proof of validity and reliability of the proposed measure and method.

### 3.6.2 Combined Analysis

The left graph in Figure 3-14 presents the maintenance cost of a Composition for a single future addition according to the general equation (3-13) for VP. Axis x of the graph presents a logical range of factor  $\mu=(0,\dots,3]$  or else a range of  $M=(0,\dots,3N]$ . Axis z presents the full range of the factor  $p_{nE}=[0,\dots,1]$ . Because of  $p_{nE}=1-p_{nP}$ , when  $p_{nE}=0$  the elements of the structure are steady, and when  $p_{nE}=1$  the set of the operations is steady.

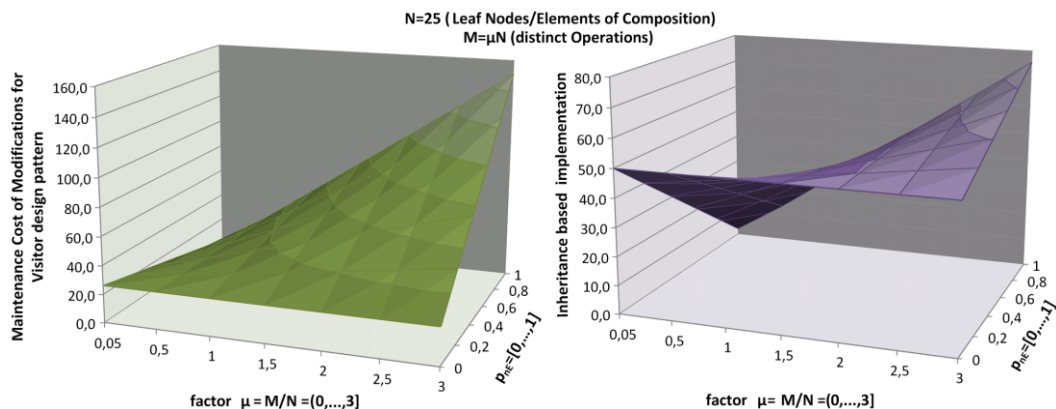


Figure 3-14: Graphs of maintenance cost of modifications on a Composition for a single future addition, related to  $\mu$  and  $p_{nE}$  factors, referred to the Visitor design pattern (VP) and Inheritance based implementation (IBI).

Left graph in Figure 3-14 shows how the maintenance cost of VP changes while  $p_{nE}$  factor shifts from 0 to 1. It clearly shows the stability of the pattern when  $p_{nE}=0$  (steady

structure) no matter how many operations ( $M, \mu$ ) exist. The right graph in Figure 3-14 presents the maintenance cost of a Composition for a single future addition according to the general equation (3-14) for IBI.

In general, the graphs in Figure 3-14 do not uniformly illustrate the distance or the gain of efficiency between pattern combinations. To prove the increased efficiency of VP against IBI, a graph of the distance of maintenance cost is presented in Figure 3-15. Axis y of the graph presents the distance of maintenance cost between VP and IBI given by simple equation (3-16).

$$c_{m(dist)}(p_{nE}) = c_m(p_{nE}, CVP) - c_m(p_{nE}, CIBI) = N\mu p_{nE} + Np_{nE} - N - 2p_{nE} + 1 \quad (3-16)$$

When the graph’s surface in Figure 3-15 is under zero level, the VP design alternative is preferred for the specific values of  $\mu$  and  $p_{nE}$  factors. Similarly, when the graph’s surface is above zero level, the IBI design alternative is preferred. The distance cost (absolute) value represents the gain between the patterns.

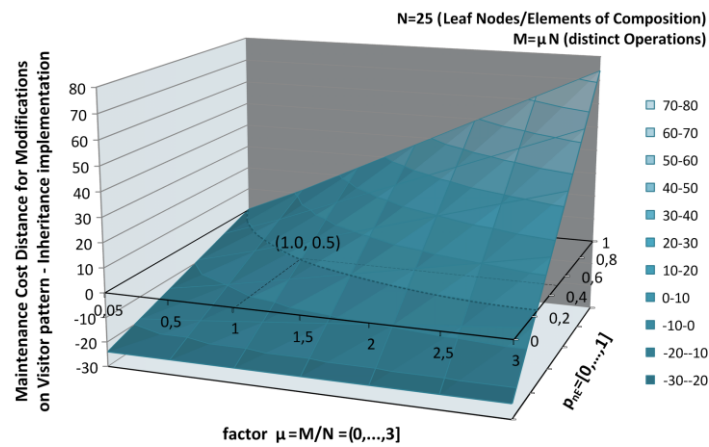


Figure 3-15: Graph of asymptotic cost differentiation  $C_{diff}$  (3) for modifications on a Composition for a single future addition, related to  $\mu$  and  $p_{nE}$  factors, referred to the Visitor design pattern (VP) and inheritance-based implementation (IBI).

The graph in Figure 3-15 illustrates a full representation of the solution or the design space about CVP and CIBI pattern comparisons. The section limit of the graph surface (representing the zero-cost level) in Figure 3-15 is a curved (dot) line that indicates all balance cases where VP and IBI have equal cost (or zero distance). In general, the balance line is close to the limit of equation (3-15) which is graphically presented in Figure 3-16.

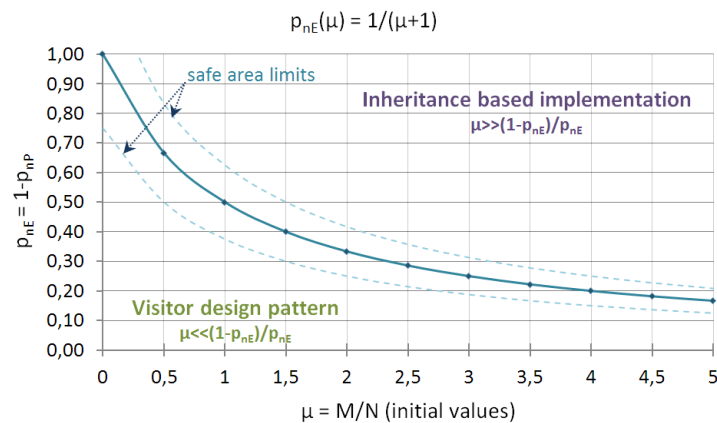


Figure 3-16: Graph of balance cases (equal maintenance cost) for Visitor design pattern (VP) vs Inheritance based implementation (IBI).

The graphs in Figure 3-15 and Figure 3-16 contain all the information on the behavior of VP and IBI on a Composition for a single future addition. They show which pattern should be selected based on the values of  $\mu=M/N$  and  $p_{nE}$  factors. In general, VP is preferred when factors  $\mu$  and  $p_{nE}$  have small values. IBI is preferred when factors  $\mu$  and  $p_{nE}$  have large values. For a safe choice, factor  $\mu$  should not be too close to the trace of balance cases (indicated by safe area limits in Figure 3-16). Furthermore, Figure 3-15 and Figure 3-16 clearly show that probability analysis over maintenance scenarios has a decisive role on maintenance cost estimation, proving the usefulness of the proposed theory and model.

### 3.7 Progressive Analysis

In this subsection, the progressive behavior of the proposed metrics during maintenance process is presented through a formal mathematical analysis.

#### 3.7.1 Deriving Progressive Maintenance Cost

In this subsection, the previously defined metric of SMC in Table 3-3 is used for evaluating and computing the total progressive maintenance cost for each design combination and for several applied maintenance scenarios.

As mentioned in subsection 3.5.6, maintenance is a progressive software evolution process, described in Figure 3-12. During this progressive process, the continuously updated values of the design characteristics and attributes affect the intermediate computations of the maintenance cost. Thus, to analyze the intermediate behavior of the proposed metrics, the progressive structural maintenance cost should be defined.

*Definition of Progressive Structural Maintenance Cost (PSMC):* represents the effort required to adjust a specific design pattern combination through the progressive implementation of several maintenance scenarios, based on their individual probabilities, during software maintenance process.

Based on the above definition and the merged maintenance costs as expressed in equations (3-9)-(3-12), the progressive maintenance cost is defined considering the maintenance as a progressive evolution process in Figure 3-12.

*Maintenance on D pattern combination:* The PSMC ( $pc_m$ ) for progressively implementing  $\lambda$  future maintenance scenarios on  $D \in \{CVP, CIBI\}$  pattern combination considering a known  $p$  (for  $p_{nE}$ ) probability is given by Equation (3-17).

$$pc_m(\lambda, p, D) = \sum_{j=1}^{\lambda} \left( \frac{1}{\left| \frac{p}{1-p} \right| + \left| \frac{1-p}{p} \right|} \left( \sum_{i=1}^{\left| \frac{p}{1-p} \right|} c_m^S(n_e, D) + \sum_{i=1}^{\left| \frac{1-p}{p} \right|} c_m^S(n_p, D) \right) \right) \quad (3-17)$$

The analysis of maintenance cost as a progressive evolution process, is another innovation of the proposed theory and model, compared to existing related work such as (Hills et al., 2011; Tom Mens & Eden, 2005).

The Equation (3-17) assumes a normal or cyclical arriving pattern of events based on their individual probabilities. This assumption is necessary to facilitate the mathematical representation and formulation of a dynamic phenomenon such as the arrival and application of various types of events during maintenance. In addition, a cyclical arriving pattern express the most probable sequence of events based on their probabilities and, thus it is offered as a safe and reliable approximation of an actual and random sequence of events. Furthermore, the  $z'$  and  $z$  factors in Table 3-2 convert the real probability factors (i.e.,  $p_{nE}$ ,  $p_{nP}$ ) to integer values, thus, allowing the event-driven and discrete analysis of the applied scenarios from a mathematical perspective. For example, if  $\lambda=200$ ,  $p_{nE}=0.25$  and  $p_{nP}=0.75$  then  $z = 1$  and  $z' = 3$ . Thus, each cycle of arriving events contains the application of 1 new Element event and next the continuous application of 3 new Operation events.

The total number of cycles or repetitions is given by the formula  $\lambda/(z + z') = 200/4 = 50$ . Summarizing, the total number of applied events (maintenance scenarios) is represented by 50 cycles or repetitions of  $(1+3)=4$  arriving events each, thus  $\lambda=50 \cdot 4=200$ . However, according to the progressive evolution process in Figure 3-12, each scenario application affects the values of design attributes (i.e., N and M). This is an aspect that Equation (3-17) misses and further analyzed in following subsections.

### 3.7.2 Progressive Maintenance Cost Computation

Using Visitor equations (3-9) and (3-10) and assuming a normal arriving pattern of events based on  $p_{nE}$  and  $p_{nP}$  propabilities, the progressive maintenance cost for all possible future additions ( $\lambda$ ) in a Composition with N initial nodes and M initial operations can be computed based on the general equation (3-17). Figure 3-17 presents the computation steps for  $\lambda$  future additions where  $p=p_{nE}$ , N=initial distinct elements, and M=initial distinct operations. Each row in Figure 3-17 represents a cycle of  $(z + z')$  arriving events based on their probabilities. Furthermore, the first column represents the new Element events and the second column the new Operation events for each cycle of arriving events. The intersection of columns and rows represents the continuous application of a specific event type for a specific cycle of arriving events. This representation gives emphasis and analyzes the intermediate increments of the design attributes (i.e., N and M) as they affected by the arriving (applied) maintenance scenarios (events).

$$\begin{array}{c}
 \frac{\lambda}{\lfloor \frac{p}{1-p} \rfloor + \lfloor \frac{1-p}{p} \rfloor} \\
 \text{times}
 \end{array}
 \left| \begin{array}{cc}
 \begin{array}{c} \lfloor \frac{p}{1-p} \rfloor \text{ times} \\ \\ \\ \end{array} & \begin{array}{c} \lfloor \frac{1-p}{p} \rfloor \text{ times} \\ \\ \\ \end{array} \\
 \hline
 \begin{array}{cc}
 (2M)_{n_E} + \dots + (2M)_{n_E} & + \left( (N + \lfloor \frac{p}{1-p} \rfloor) + 1 \right)_{n_P} + \dots + \left( (N + \lfloor \frac{p}{1-p} \rfloor) + 1 \right)_{n_P} + \\
 + \left( 2(M + \lfloor \frac{1-p}{p} \rfloor) \right)_{n_E} + \dots + \left( 2(M + \lfloor \frac{1-p}{p} \rfloor) \right)_{n_E} & + \left( (N + 2 \lfloor \frac{p}{1-p} \rfloor) + 1 \right)_{n_P} + \dots + \left( (N + 2 \lfloor \frac{p}{1-p} \rfloor) + 1 \right)_{n_P} + \\
 + \left( 2(M + 2 \lfloor \frac{1-p}{p} \rfloor) \right)_{n_E} + \dots + \left( 2(M + 2 \lfloor \frac{1-p}{p} \rfloor) \right)_{n_E} & + \left( (N + 3 \lfloor \frac{p}{1-p} \rfloor) + 1 \right)_{n_P} + \dots + \left( (N + 3 \lfloor \frac{p}{1-p} \rfloor) + 1 \right)_{n_P} + \\
 + \dots + & + \dots +
 \end{array}
 \end{array}$$

Figure 3-17: Computation of progressive maintenance cost for  $\lambda$  future additions/modifications on a Composite using Visitor design pattern (CVP).

Equation (3-18) is extracted from the steps in Figure 3-17. By replacing  $z$  and  $z'$  (as showed in Table 3-3) in equation (3-18), the equation (3-19) is derived, which computes the progressive maintenance cost for  $\lambda$  future additions on VP design combination.

$$\sum_{\varphi=1}^{\lambda} \frac{\lambda}{\lfloor \frac{p}{1-p} \rfloor + \lfloor \frac{1-p}{p} \rfloor} \left[ \sum_{i=1}^{\lfloor \frac{p}{1-p} \rfloor} 2 \left( M + (\varphi - 1) \lfloor \frac{1-p}{p} \rfloor \right) + \sum_{i=1}^{\lfloor \frac{1-p}{p} \rfloor} \left( \left( N + \varphi \lfloor \frac{p}{1-p} \rfloor \right) + 1 \right) \right] \quad (3-18)$$

$$\mathbf{pc}_m(\lambda, p_{nE}, \text{CVP}) = \left( 2zM + zN + z - 2zZ + \frac{3zZ}{2} \cdot \frac{\lambda + z + z'}{z + z'} \right) \cdot \frac{\lambda}{z + z'} \quad (3-19)$$

Similarly, the equation (3-20) computes the progressive maintenance cost for  $\lambda$  future additions on IBI combination.

$$\mathbf{pc}_m(\lambda, p_{nE}, \text{CIBI}) = \left( 2zN + zM + z - 2zZ + \frac{3zZ}{2} \cdot \frac{\lambda + z + z'}{z + z'} \right) \cdot \frac{\lambda}{z + z'} \quad (3-20)$$

Equations (3-19) and (3-20) represent the relative magnitudes of the total required maintenance effort or cost during software evolution after  $\lambda$  scenario's applications, and for each implementation alternative (CVP, CIBI). Since the factors  $\lambda$ , M, and N are simultaneously increased, both equations have a positive exponentially increased trend.

Thus, the equations (3-19) and (3-20) is in accordance with Lehman’s second law (Meir M. Lehman et al., 1997) since PSMC ( $pc_m$ ) metric is related to code complexity property as analyzed in subsection 3.4.2. Furthermore, both equations converge to Bakota’s relevant cost curve (Bakota et al., 2012) when the change rate is constant over time.

By replacing factors  $z$  and  $\acute{z}$  in equation (3-19), the following mathematical transformations (assuming  $p_{nE} > 1/2$ ) leads to an alternate formulation of the progressive maintenance cost of CVP expressed in terms of all problem’s parameters (i.e., initial  $N$  and  $M$ ,  $p_{nE}$ ,  $p_{nP}$ , and  $\lambda$ ).

$$\begin{aligned}
 pc_m(\lambda, p_{nE}, CVP) &= \left( 2zM + \acute{z}N + \acute{z} - 2z\acute{z} + \frac{3z\acute{z}}{2} \cdot \frac{\lambda + z + \acute{z}}{z + \acute{z}} \right) \cdot \frac{\lambda}{z + \acute{z}} = \xrightarrow{\acute{z}=1 (p_{nE} > 1/2)} \\
 &= \left( 2zM + N + 1 - 2z + \frac{3z}{2} \cdot \frac{\lambda + z + 1}{z + 1} \right) \cdot \frac{\lambda}{z + 1} = \xrightarrow{z+1 = \left\lceil \frac{p_{nE}}{1-p_{nE}} \right\rceil + 1 \approx \left\lceil \frac{1}{1-p_{nE}} \right\rceil} \\
 &= \left( 2 \frac{p_{nE}}{1-p_{nE}} M + N + 1 - 2 \frac{p_{nE}}{1-p_{nE}} + \frac{3}{2} \frac{p_{nE}}{1-p_{nE}} \cdot \left( \lambda + \frac{1}{1-p_{nE}} \right) \cdot (1-p_{nE}) \right) \cdot \lambda (1-p_{nE}) \\
 &= \left( 2p_{nE}M + N - Np_{nE} + 1 - 3p_{nE} + \frac{3p_{nE}}{2} \cdot \lambda(1-p_{nE}) + \frac{3p_{nE}}{2} \right) \cdot \lambda = \\
 &= \left( 2p_{nE}M + Np_{nP} + 1 - 3p_{nE} + \frac{3p_{nE}}{2} \cdot \lambda p_{nP} + \frac{3p_{nE}}{2} \right) \cdot \lambda = \\
 &= \left( 2p_{nE}M + Np_{nP} + 1 - \frac{3p_{nE}}{2} + \frac{3\lambda p_{nE} p_{nP}}{2} \right) \cdot \lambda = \\
 &= \frac{3}{2} \lambda^2 p_{nE} p_{nP} + 2\lambda p_{nE} M + \lambda p_{nP} N + \lambda - \lambda \frac{3}{2} p_{nE}
 \end{aligned}$$

Respectively, by replacing factors  $z$  and  $\acute{z}$  in equation (3-19), the following mathematical transformations (assuming  $p_{nE} < 1/2$ ) leads to a similar formulation of the progressive maintenance cost of CVP expressed in terms of all problem’s parameters (i.e., initial  $N$  and  $M$ ,  $p_{nE}$ ,  $p_{nP}$ , and  $\lambda$ ).

$$\begin{aligned}
 pc_m(\lambda, p_{nE}, CVP) &= \left( 2zM + \acute{z}N + \acute{z} - 2z\acute{z} + \frac{3z\acute{z}}{2} \cdot \frac{\lambda + z + \acute{z}}{z + \acute{z}} \right) \cdot \frac{\lambda}{z + \acute{z}} = \xrightarrow{z=1 (p_{nE} < 1/2)} \\
 &= \left( 2M + \acute{z}N + \acute{z} - 2\acute{z} + \frac{3\acute{z}}{2} \cdot \frac{\lambda + \acute{z} + 1}{\acute{z} + 1} \right) \cdot \frac{\lambda}{\acute{z} + 1} = \xrightarrow{\acute{z}+1 = \left\lceil \frac{p_{nP}}{1-p_{nP}} \right\rceil + 1 \approx \left\lceil \frac{1}{1-p_{nP}} \right\rceil} \\
 &= \left( 2M + \frac{p_{nP}}{1-p_{nP}} N + \frac{p_{nP}}{1-p_{nP}} - 2 \frac{p_{nP}}{1-p_{nP}} + \frac{3}{2} \frac{p_{nP}}{1-p_{nP}} \cdot \left( \lambda + \frac{1}{1-p_{nP}} \right) \cdot (1-p_{nP}) \right) \cdot \lambda (1-p_{nP}) \\
 &= \left( 2M - 2Mp_{nP} + p_{nP}N + p_{nP} + \frac{3p_{nP}}{2} \cdot \lambda(1-p_{nP}) + \frac{3p_{nP}}{2} \right) \cdot \lambda = \\
 &= \left( 2M - 2Mp_{nP} + p_{nP}N + p_{nP} + \frac{3p_{nP}}{2} \cdot \lambda p_{nE} + \frac{3p_{nP}}{2} \right) \cdot \lambda = \\
 &= \left( 2Mp_{nE} + p_{nP}N + p_{nP} + \frac{3p_{nE}}{2} + \frac{3\lambda p_{nE} p_{nP}}{2} \right) \cdot \lambda =
 \end{aligned}$$

$$\begin{aligned}
&= \frac{3}{2}\lambda^2 p_{nE} p_{nP} + 2\lambda p_{nE} M + \lambda p_{nP} N + \lambda p_{nP} + \lambda \frac{3p_{nE}}{2} = \\
&= \frac{3}{2}\lambda^2 p_{nE} p_{nP} + 2\lambda p_{nE} M + \lambda p_{nP} N + \lambda(1 - p_{nE}) - \frac{3}{2}\lambda p_{nE} = \\
&= \frac{3}{2}\lambda^2 p_{nE} p_{nP} + 2\lambda p_{nE} M + \lambda p_{nP} N + \lambda - \lambda p_{nE} - \frac{3}{2}\lambda p_{nE} = \\
&= \frac{3}{2}\lambda^2 p_{nE} p_{nP} + 2\lambda p_{nE} M + \lambda p_{nP} N + \lambda - \frac{5}{2}\lambda p_{nE}
\end{aligned}$$

Because  $3/2 \cdot p_{nE} (\text{for } p_{nE} > 1/2) \approx 5/2 \cdot p_{nE} (\text{for } p_{nE} < 1/2)$ , the latest expressions of the progressive maintenance cost of CVP are almost equivalent for any value of  $p_{nE} = [0, \dots, 1]$ . Similar mathematical transformations can be applied in equation (3-20) for the CIBI design alternative. This approach transforms equations with integer parameters (representing an event-driven evolution pattern) to equations with real (float) parameters, thus, implying that the underlying event-oriented evolution pattern may be also supported by continuous integration through integrals as explained in subsection 3.7.5.

### 3.7.3 Reverse Analysis (Verification)

In this subsection, a reverse analysis beginning from the progressive maintenance cost and ending to the fundamental equations of structural maintenance cost is presented. This is an attempt that verifies the previously conducted analysis. By replacing factors  $z$  and  $\acute{z}$  and setting  $\lambda=1$  in the general equations (3-19) and (3-20), the equations (3-13) and (3-14) are derived. Note that for  $p_{nE} > 0.5$ ,  $\acute{z} = 1$  and for  $p_{nE} < 0.5$ ,  $z = 1$ . As an example, the mathematical operations from equation (3-19) to equation (3-13) for VP are presented below (only for  $p_{nE} = p > 1/2$ ). Similar mathematical operations exist for  $p_{nE} = p \leq 1/2$  which end up to the same result.

$$\begin{aligned}
\mathbf{pc}_m(\lambda, p_{nE}, \text{CVP}) &= \left( 2zM + \acute{z}N + \acute{z} - 2z\acute{z} + \frac{3z\acute{z}}{2} \cdot \frac{\lambda + z + \acute{z}}{z + \acute{z}} \right) \cdot \frac{\lambda}{z + \acute{z}} \xrightarrow{\acute{z}=1 (p>1/2)} \\
&= \left( 2zM + N + 1 - 2z + \frac{3z}{2} \cdot \frac{\lambda + z + 1}{z + 1} \right) \cdot \frac{\lambda}{z + 1} \xrightarrow{z+1 = \left\lceil \frac{p}{1-p} \right\rceil + 1 \approx \left\lceil \frac{1}{1-p} \right\rceil} \\
&= \left( 2 \frac{p}{1-p} M + N + 1 - 2 \frac{p}{1-p} + \frac{3}{2} \frac{p}{1-p} \cdot \left( \lambda + \frac{1}{1-p} \right) \cdot (1-p) \right) \cdot \lambda(1-p) \\
&= \left( 2pM + N - Np + 1 - 3p + \frac{3p}{2} \cdot (\lambda - \lambda p + 1) \right) \cdot \lambda \xrightarrow{\lambda=1} \\
&= 2pM + N - Np + 1 - 3p + \frac{3p}{2} \cdot (2-p) \\
&= 2pM + N - Np + 1 + \frac{3p^2}{2} \approx \left[ \begin{array}{l} p = [1/2, \dots, 1] \\ \frac{3p^2}{2} = [3/8, \dots, 3/2] \end{array} \right] \\
&\approx 2pM + N - Np + 1 + p \\
&= N(2\mu p_{nE} - p_{nE} + 1) + 1 - p_{nE} = \mathbf{c}_m(p_{nE}, \text{CVP})
\end{aligned}$$

Also, looking for the sign of distance in equation (3-21) the equation (3-16) is extracted. This means that the trend (sign) of the distance of PSMC of equation (3-21) is almost identical to equation (3-16).

$$\mathbf{PC}_{\mathbf{m}(\text{dist})}(\lambda, p_{nE}) = \mathbf{PC}_{\mathbf{m}}(\lambda, p_{nE}, \text{CVP}) - \mathbf{PC}_{\mathbf{m}}(\lambda, p_{nE}, \text{CIBI}) = (z\mu N + \acute{z} - \acute{z}N - z) \cdot \frac{\lambda}{z + \acute{z}} \quad (3-21)$$

Furthermore, considering that for  $p_{nE} > 0.5$ ,  $\acute{z} = 1$ , and  $z + 1 \approx 1/(1 - p_{nE})$  then

$$\frac{z}{z + \acute{z}} = \frac{z}{z + 1} \approx \frac{z}{\frac{1}{1 - p_{nE}}} \approx \frac{z(1 - p_{nE})}{1} = p_{nE}$$

$$\frac{\acute{z}}{z + \acute{z}} = \frac{1}{z + 1} \approx \frac{1}{\frac{1}{1 - p_{nE}}} = (1 - p_{nE}) = p_{nP}$$

Respectively, for  $p_{nE} < 0.5$ ,  $z = 1$ , and  $\acute{z} + 1 \approx 1/(1 - p_{nP})$  then

$$\frac{z}{z + \acute{z}} = \frac{1}{1 + \acute{z}} \approx \frac{1}{\frac{1}{1 - p_{nP}}} = (1 - p_{nP}) = p_{nE}$$

$$\frac{\acute{z}}{z + \acute{z}} = \frac{\acute{z}}{1 + \acute{z}} \approx \frac{\acute{z}}{\frac{1}{1 - p_{nP}}} \approx \frac{\acute{z}(1 - p_{nP})}{1} = p_{nP}$$

Thus,  $\frac{z}{z + \acute{z}} \approx p_{nE}$  and  $\frac{\acute{z}}{z + \acute{z}} \approx p_{nP}$  for any decimal value of  $p_{nE}$  and  $p_{nP}$  factors. Now, the similarity of equation (3-21) with the equation (3-16) can be demonstrated as follows:

$$\begin{aligned} \mathbf{PC}_{\mathbf{m}(\text{dist})}(\lambda, p_{nE}) &= (z\mu N + \acute{z} - \acute{z}N - z) \cdot \frac{\lambda}{z + \acute{z}} = \lambda(N\mu p_{nE} + p_{nP} - Np_{nP} - p_{nE}) = \\ &= \lambda(N\mu p_{nE} - N(1 - p_{nE}) + p_{nP} - p_{nE}) = \lambda(N\mu p_{nE} + Np_{nE} - N + p_{nP} - p_{nE}) \approx \\ &\approx \lambda(N\mu p_{nE} + Np_{nE} - N - 2p_{nE} + 1) \end{aligned}$$

### 3.7.4 Graph of Progressive Maintenance Cost

A graph of the distance of PSMC based on equation (3-21) is presented in Figure 3-18. The  $\lambda$  factor has been set to a large value ( $\lambda=30$ ) in order to show the stability of graph's shape for almost any number of future additions.



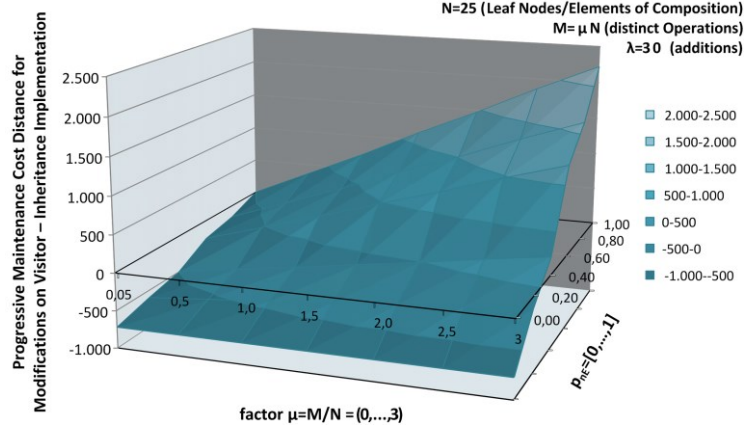


Figure 3-18: Graph of progressive maintenance cost differentiation  $pc_{cm(dist)}(p_{nE})$  for modifications on a Composition for  $\lambda$  future additions, related to the  $\mu$  and  $p_{nE}$  factors, referred to the Visitor design pattern (VP) and Inheritance based implementation (IBI).

The graph surfaces in Figure 3-15 and Figure 3-18 are almost identical, indicating that the impact of a single addition ( $\lambda=1$ ) and the progressive analysis of maintenance costs are matched. It is proven that the choice of the proper design pattern combination can be made directly by using the single addition equations (3-13) and (3-14) or the distance equation (3-16) or its graph in Figure 3-15.

### 3.7.5 Integrated Maintenance Cost

Equations (3-19), (3-20), and (3-21) are the result of a quantitative analysis returning the progressive or total required effort based on the analysis of repeated and distinct scenario's applications. In this subsection, a more typical mathematical perspective about the progressive maintenance cost based on calculus and integration concept is presented.

Equations (3-13), (3-14), and (3-16) compute the maintenance cost for CVP and CIBI pattern combinations based on  $N$ ,  $M$  design attributes and  $p_{nE}$  probability factor for a single addition ( $\lambda=1$ ). Since  $N$  and  $M$  attributes are continuously updated during the progressive or repeated implementation of different maintenance scenarios in respect to their probabilities ( $p_{nE}$ ,  $p_{nP}=1-p_{nE}$ ), it is possible to rewrite the equations (3-13), (3-14), and (3-16) for  $N$  and  $M$  values that are based on  $\lambda$ th future addition. Thus, equation (3-16) is transformed to the equation (3-22).

$$c_{m(dist)}(p_{nE}) = N\mu p_{nE} + Np_{nE} - N - 2p_{nE} + 1 = Mp_{nE} + Np_{nE} - N - 2p_{nE} + 1 \quad (3-22)$$

$$= (\lambda(1 - p_{nE}) + M^{initial})p_{nE} + (\lambda p_{nE} + N^{initial})p_{nE} - \lambda p_{nE} - N^{initial} - 2p_{nE} + 1$$

$$\text{Where current } N = \lambda p_{nE} + N^{initial} \text{ and current } M = \lambda(1 - p_{nE}) + M^{initial}$$

Equation (3-22) computes the distance of maintenance cost only for the  $\lambda$ th maintenance scenario. The distance of PSMC can be derived through the continuous integration of equation (3-22) on  $\lambda$  factor. Thus, the distance of PSMC of equation (3-21) can also be expressed by the integral in the general equation (3-23).

$$\begin{aligned} pc_{m(dist)}(\lambda, p_{nE}) &= \int c_{m(dist)}(p_{nE}) d\lambda \\ &= \lambda(M^{initial} p_{nE} + N^{initial} p_{nE} - N^{initial} - 2p_{nE} + 1) + C \end{aligned} \quad (3-23)$$

$$\text{Where current } N = \lambda p_{nE} + N^{initial} \text{ and current } M = \lambda(1 - p_{nE}) + M^{initial}$$

Equation (3-23) has a simpler mathematical representation than equation (3-21) and returns similar results for  $C=0$ . For a different set of measures, the equations (3-19)–(3-21), and (3-23) could be more complex. The approach of continuous integration is a more direct and easier way for deriving formal models and computing progressive costs when mathematical tools such Matlab<sup>4</sup> or MS Mathematics<sup>5</sup> are used. On the other hand, it hides a significant aspect of the distinct analysis or the event-driven (cyclical) pattern of applied scenarios during maintenance process.

### 3.8 Application of the Proposed Model

In this subsection, a diagram which summarizes the (structural) maintenance cost equations, an application flowchart, three application examples of the proposed model, and the comparison of the proposed measurement approach with two other relevant existing approaches are presented.

#### 3.8.1 Summarizing Maintenance Cost of the Model

A diagram summarizing the (structural) maintenance cost equations of the proposed model is presented in Figure 3-19. The graph displays all the relations between the proposed maintenance metrics and equations for both single addition and progressive analysis. Figure 3-19 can also be used as a computational pattern of the model for the analysis of other general and significant design problems.

#### 3.8.2 Classification and Application Flowchart of Proposed Model

The proposed model is classified based on the diagram in Figure 3-7. The interrelations between the proposed model and different concepts and aspects of software maintainability assessment are presented in Figure 3-20.

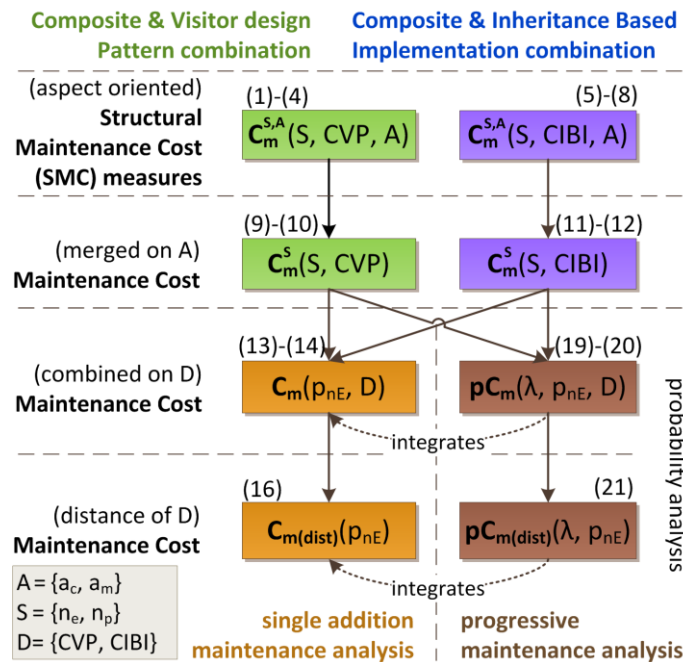


Figure 3-19: Computational pattern of Structural Maintenance Cost.

<sup>4</sup> Licensed mathematical suite on <http://www.mathworks.com/>

<sup>5</sup> Freeware equation solver on <https://www.microsoft.com/en-us/download/details.aspx?id=15702>

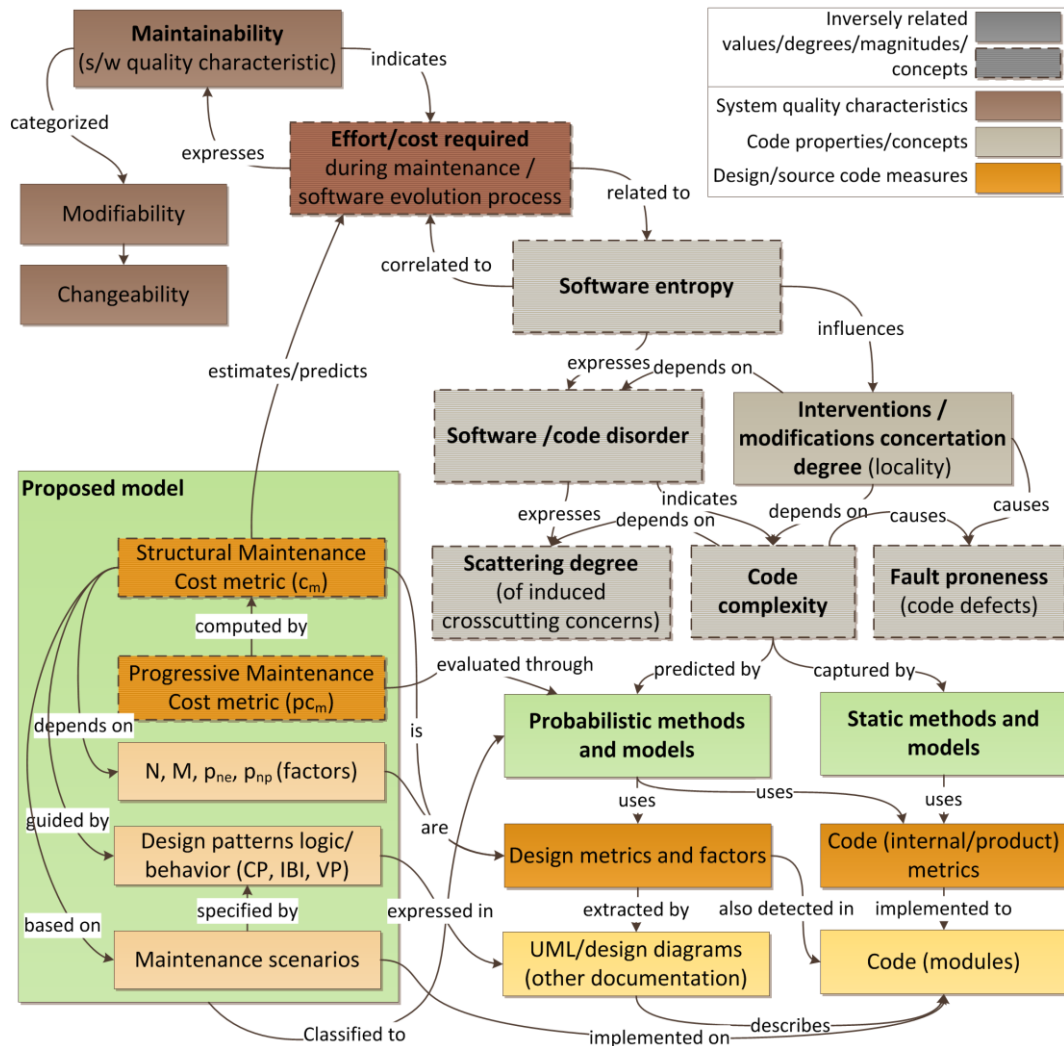


Figure 3-20: Classification diagram of the proposed model.

Figure 3-21 presents a simple application flowchart of the proposed model which shows the proper use of CP, IBI, VP and Iterator design patterns through simple steps of sub-decisions. The Iterator design pattern is usually combined with the other patterns and is fully analyzed in (Gamma et al., 1994; Gibbons & Oliveira, 2009).

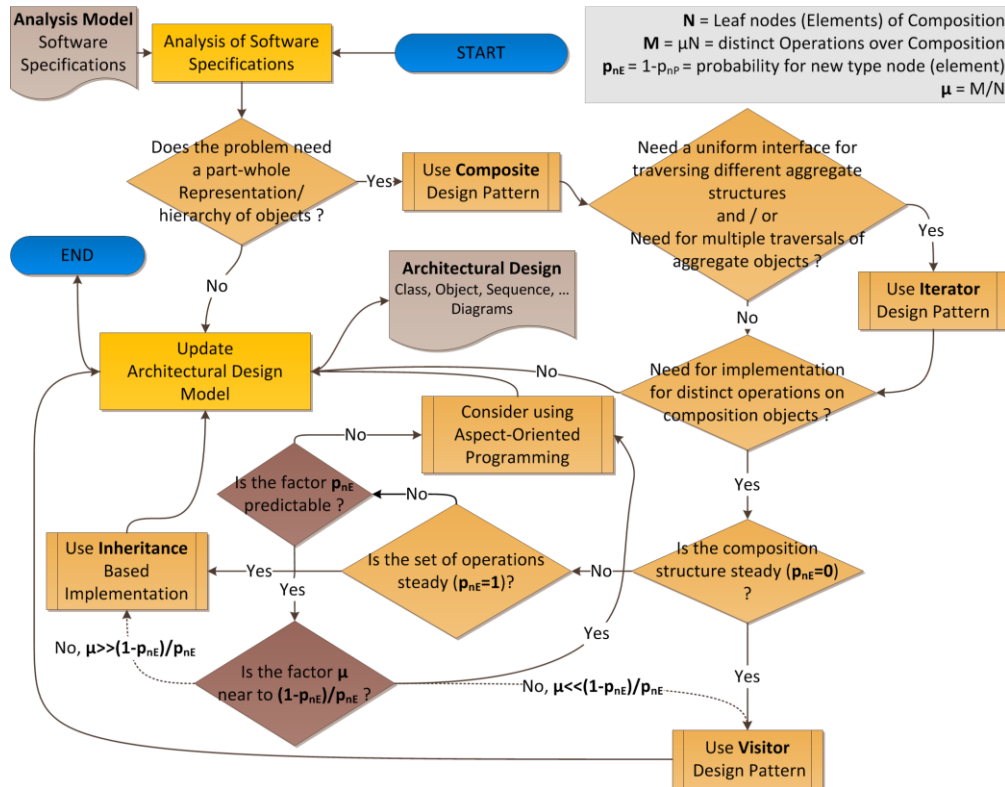


Figure 3-21: Application flowchart on the use of Composite, Visitor, Iterator design patterns and inheritance-based implementation.

As shown in subsections 3.4 and 3.5, a safe choice between IBI and VP can be made with no concern about the number of future additions, especially when a good estimation of  $p_{nE}$  probability can be made. Aspect-oriented programming (AOP) technology is a possible next step, and presented in (Elrad, Filman, & Bader, 2001; Filman, Elrad, Clarke, & Aksit, 2004). However, AOP technology, as a generative approach, usually depends on Domain Specific Languages (DSLs) which produce source code for known high-level languages. Thus, the requirement for mastering an extra language and the extra compilation stage puts AOP under consideration.

### 3.8.3 Application Examples

In this subsection, three case studies of the application of the proposed model based on progressive analysis are presented.

*Problem Descriptions:* Based on the specifications of the case studies of the motivational examples in Table 1-1, the design attributes of each description are derived and presented in Table 3-4. Each of these examples corresponds to an instance of the general design problem (i.e., CVP vs CIVI) under study. Because all these examples are complex problems representing critical systems, maintainable software which can be easily modified should be produced. It is assumed that it was decided to use the CP for the structure representations, but a decision on IBI or VP for the implementation of the operations has not been made. Moreover, both operation and element sets could be extended during software maintenance.

Table 3-4: Characteristics and attributes of individual problem descriptions

Problem description	N: number of initial distinct Elements of Composition structure	M: number of initial Operations over composition's elements	$p_{ne}$ : probability for a new element during software maintenance process
Compiler implementation for the standard C89 high level language	155	20	0.10 (10%)
Interpreter implementation for a new custom (extendable) DSL language	40	10	0.50 (50%)
GUI implementation for a simple graph designing tool	15	14	0.70 (70%)

Note: values of each attribute have been derived from individual specifications in Table 1-1.

*Model Application:* The approach described in this chapter has been selected for the development of targeted implementations. All necessary data for model application are available in Table 3-4. Simply by looking at the graph in Figure 3-16, proper pattern combinations can be safely selected as showed in Figure 3-22.

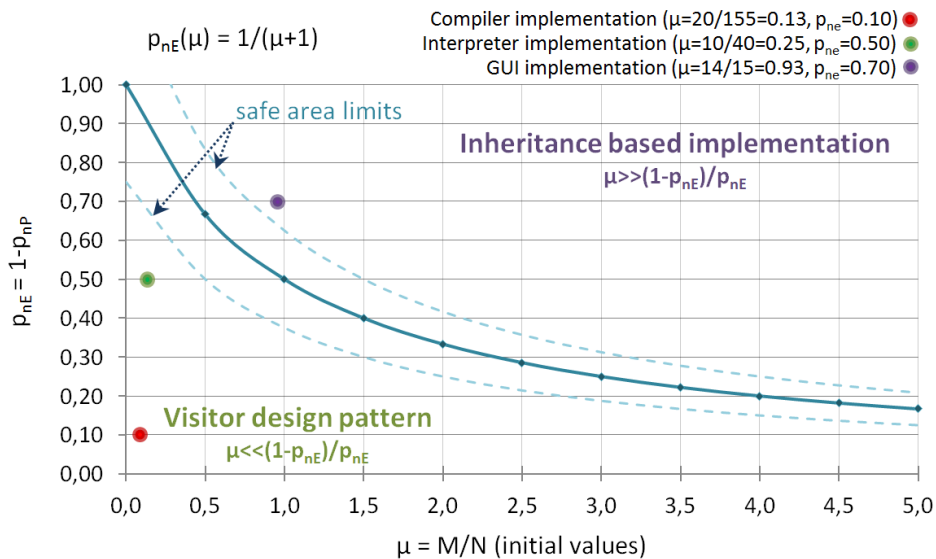


Figure 3-22: Graph of balance cases (equal maintenance cost) for CVP vs CIBI.

Hence, for the problems described in Table 3-4, it is concluded that the VP can be safely selected for Interpreter and Compiler implementations and IBI can be safely selected for GUI implementation due to their lower overall maintenance cost. Among the three case studies, Compiler is the clearest as expected due its structure stability. For Interpreter and GUI, the model clarifies the advantage of maintenance cost for each pattern combination although their structures and operation sets are extendable.

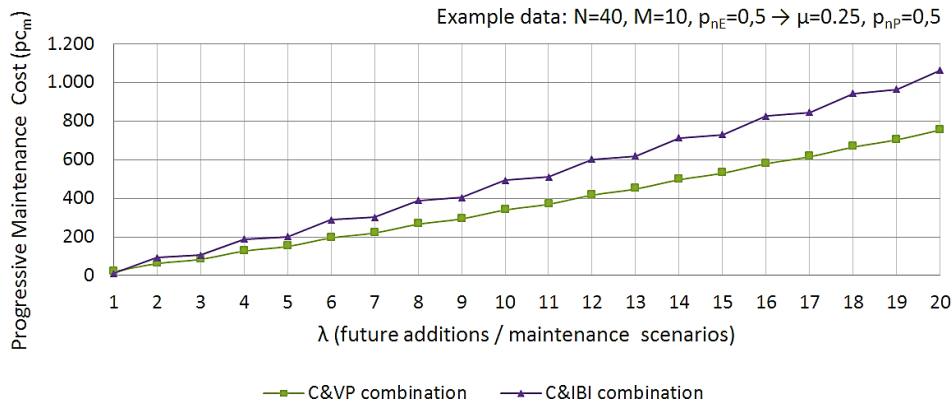


Figure 3-23: Example: computation of progressive asymptotic cost for inheritance-based implementation (IBI) and Visitor design pattern (VP).

Furthermore, equations (3-19) and (3-20) can compute the exact PSMC for any number of future modifications, as presented in Figure 3-23 for the Interpreter implementation example. Alternatively, based on the flowchart in Figure 3-21, the use of VP can be concluded since  $0.25 = \mu \ll (1-p_{nE})/p_{nE} = 1$ .

Although the proposed model implementation seems to be simple, the detailed mathematical analysis provides a complete and ready to use evaluation model regarding CVP and CIBI comparison. This model including its formal equations and the detailed graphs such as those presented in Figure 3-16, Figure 3-18, and Figure 3-21, provide a visualization for almost all the solutions or the design space of the significant decision problem of recursive hierarchies of part-whole aggregations.

### 3.8.4 Alternate Maintenance Measures

In this subsection, the maintenance measures of the proposed model are compared with similar metrics that have been proposed in the literature in Table 3-5.

Table 3-5: Correlation of Model's Measures with Related Models

Eq. 3-	Maintenance scenario, Implementation	Correlated Measures		
		Structural Maintenance Cost (this method)	Computational Complexity	Evolution Complexity
9	$c_m^S(n_e, CVP)$	$2M$	$N+2M+1$ <sup>(1)</sup>	$M$ <sup>(3)</sup>
10	$c_m^S(n_p, CVP)$	$N+1$	$2+N$ <sup>(2)</sup>	$1$ <sup>(3)</sup>
11	$c_m^S(n_e, CIBI)$	$M+1$	$2M+11$ <sup>(1)</sup>	$1$ <sup>(4)</sup>
12	$c_m^S(n_p, CIBI)$	$2N$	$3+3N$ <sup>(2)</sup>	$N$ <sup>(4)</sup>

Source: Computational Complexity (Hills et al., 2011), Evolution Complexity (Tom Mens & Eden, 2005).

Note: Related work measures have been correlated with proposed model measures through matching of similar attributes and scenarios.

<sup>1</sup> derived from analysis of scenario S1 – add two new expression operators, adapted for one new operator/element (Hills et al., 2011)

<sup>2</sup> derived from analysis of scenario S4 – add outline (new operation) (Hills et al., 2011)

<sup>3</sup> derived from analysis of case study 1: Visitor (Tom Mens & Eden, 2005)

<sup>4</sup> derived based on analysis of case study 1: Visitor correlated to Inheritance based implementation, focusing only on distinct class modifications (Tom Mens & Eden, 2005)

Equations (3-9)-(3-12) represent the merged SMC of four basic maintenance scenarios derived from the proposed metrics. Based on these equations and through the computational pattern in Figure 3-19, the progressive maintenance cost can be computed for any set of related metrics. Thus, single addition and progressive analysis of the proposed approach can be implemented for different or similar metrics. This reveals the usability and the broad application perspectives of the proposed model considering different metrics and quality characteristics.

### 3.8.5 Comparison to Relevant Metrics

In this subsection, the comparison between the proposed metric and two relevant existing metrics is presented. The purpose of this comparison is to demonstrate the usability of the proposed model and progressive analysis as a more general framework. Furthermore, the behavior of different related metrics about maintenance effort under the progressive analysis of the proposed model is explored. More specifically, the maintenance measures of Table 3-5 are analyzed through progressive analysis of the proposed model. Next, the metrics derived from the progressive analysis, are implemented to the Interpreter example data in Table 3-5 as presented in Figure 3-24.

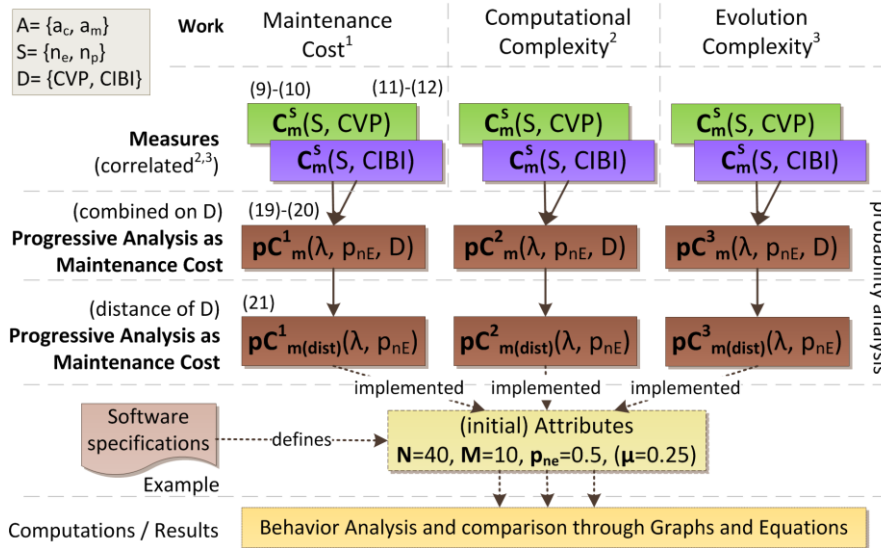


Figure 3-24: Diagram of methods comparison through progressive analysis of multiple measures.

For the SMC metrics of the proposed model, the equations (3-9)-(3-12) and (3-19)-(3-21) have been used. For Computational Complexity (Hills et al., 2011) and Evolution Complexity (Tom Mens & Eden, 2005), the correlated metrics in Table 3-5 have been used by which the PSMC  $pc^2_m$  and  $pc^3_m$  have been derived through the general equation (3-17). The results of the progressive analysis in the form of graphs such as in Figure 3-18 and Figure 3-23 are presented in Figure 3-25. These graphs show all the progressive computations and their distance produced for each distinct measure set under comparison.

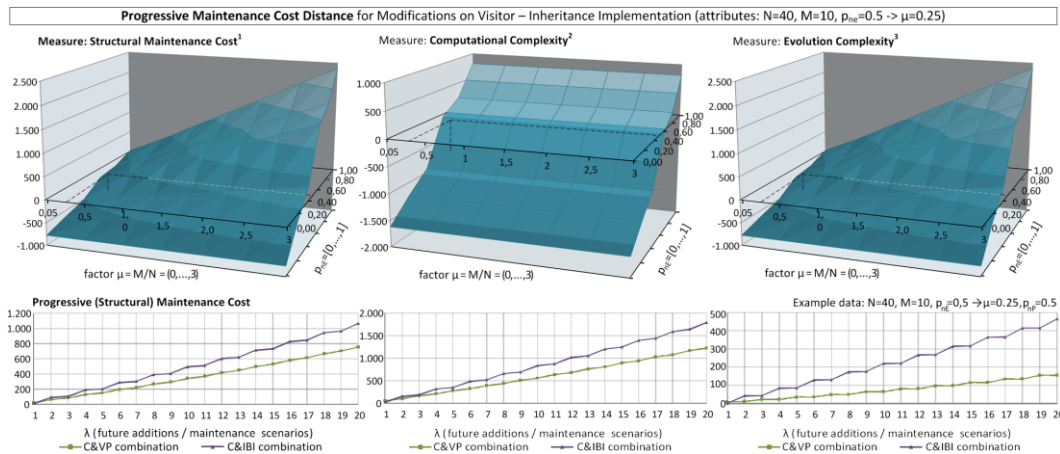


Figure 3-25: Results of metrics comparison.

Computations for new metrics and implementation data can be easily performed by using mathematical environments such as Matlab or MS Excel. All computations for this comparison have been performed through a custom function which is available online<sup>6</sup> for demonstration and further tests.

### 3.8.6 Discussion

By analyzing the previous comparison results in Figure 3-25, several interesting conclusions can be extracted. It is obvious that CVP pattern combination is preferred for all measures since it has smaller progressive cost with an increasing differentiation compared to CIBI combination. In a first glance, all metrics are correlated to a significant degree despite the variations of computed values. Although two-dimensional graphs offer enough information for selecting a proper pattern combination for the specific problem’s instance, they do not provide any information about the general behavior of the metrics referred to initial problem attributes. This information is provided by three-dimensional graphs which present the distance of progressive maintenance cost. Referring to these graphs, some major conclusions can be drawn about metrics behavior: a) all metrics are consistent regarding their outcome for the marginal values  $p_{ne}=\{0,1\}$  meaning that CVP is always recommended when  $p_{ne}=0$  and CIBI is always recommended when  $p_{ne}=1$ , confirming the opposite characteristics of IBI and VP that have been discussed in subsections 3.3.2 and 3.5.5; b) the first and third metrics are correlated, referring to distance, despite the variations of individual values; c) the second metric’s behavior is independent of initial values of attributes N and M and depends only on  $p_{ne}$  probability factor.

One other conclusion is that the first SMC (this model) and third (Tom Mens & Eden, 2005) Evolution Complexity metrics seem to be more “sensitive” than the second Computational Complexity (Hills et al., 2011) metric since they have different behavior with respect to all attributes of the problem. Also, the similarity of results between this model SMC metric and Evolution Complexity (Tom Mens & Eden, 2005) metric which is based on a visitor implementation case study, confirms the validity of the proposed model. However, the proposed SMC metric has a significant advantage compared to Evolution Complexity (Tom Mens & Eden, 2005) metric since it captures the locality degree of the applied interventions. The previews comparison results can be safely considered as confirmation about the validity and reliability of the proposed measures and model.

Summarizing, the progressive analysis on different metrics can provide a full-scale model about measuring behavior during the maintenance process. Furthermore, it indicates that different measure aspects have a significant impact on progressive computations with respect to initial attributes of a problem and should be carefully defined.

## 3.9 Methodology Determination

### 3.9.1 Methodology Description

In this subsection, a methodology based on the introduced theory is proposed and discussed through a step-by-step description in Table 3-6. Through this methodology, alternate comparison models for similar or different design pattern comparisons or for other significant and general design problems can be generated.

Table 3-6: Methodology for deriving comparison models

n	Description	Requirements / Limitations	Relation to proposed comparison model
1	Define design pattern combinations under comparison	The defined design pattern combinations should have structural nature over the solution. Also, both design pattern combinations should be targeted on solving the same (general) problem through different design architectures.	$D = \{ CVP, CIBI \}$

<sup>6</sup> Online implementation of the proposed model available in <https://www.chriskaranikolas.gr/CIBIvsCVP/>



n	Description	Requirements / Limitations	Relation to proposed comparison model
2	Define quality characteristics under evaluation	Conceptually, the defined quality characteristics should be objectively measurable. Also, they should be focused on the evaluation of design pattern architecture itself.	Maintainability Changeability
3	Define specific quantitative problem attributes as factors	The defined problem attributes should be measurable (e.g. through UML class diagrams (Lavazza & Agostini, 2005) or explicit software specifications). Also, they should be general and focused on design patterns architecture itself.	N: initial number of leaf classes of composition M: initial number of operations over composition element
4	Define major maintenance scenarios and its distinct probability factors	The defined maintenance scenarios should be selected based on specific design pattern characteristics. Probability factors should be complementary (as mutually exclusive), having total sum equal to 1. The total number of maintenance scenarios is suggested to be limited.	$S = \{ n_e, n_p \}$ Adding structure element ( $n_e, p_{ne}$ ) Adding operation/process ( $n_p, p_{np}$ )
5	Define aspects of measures	The defined aspects of measures should be focused on capturing quality characteristics defined in step (2). Also, they should be focused on the evaluation of design pattern architecture itself. Modifications on distinct methods and classes are considered as fundamental maintainability quality aspects for early design patterns assessment.	$A = \{ a_m, a_c \}$ number of modifications on distinct methods ( $a_m$ ) and distinct classes ( $a_c$ )
6	Define metrics, other factors for quantifying quality characteristics (defined in step 2)	An equation should be derived for each design pattern combination (defined in step 1), maintenance scenario (defined in step 4) and aspect of measure (defined in step 5) in correspondence to general Equations (3-1) to (3-8). Totally $ D  \times  S  \times  A $ equations should be derived by involving attribute factors (defined in step 3) as independent variables and other constant factors.	$C_m^{S,A}(S, D, A)$ Equations: (3-1) to (3-8) $ D  \times  S  \times  A  = 2^3 = 8$ Existing factors: M, N
7	Derive the merged/combined equations and graphs	The equations of the previous step should be merged / combined using probability factors (defined in step 4). Also, new constant factors can be identified and used in equations to provide a premium/penalty onto distinct amounts regarding their individual weight on quality assessment.	Factors: $p_{ne}, p_{np}$ New factors: $\mu = M/N$ Equations: (3-9)-(3-12), (3-13), (3-14), (3-16), (3-19)-(3-21), (3-23) Graphs : Figure 3-13 to Figure 3-16, and Figure 3-18 Computational pattern: Figure 3-19
8	Seek for convergence limits of the attribute factors (defined in step 3)	Convergence limit can be estimated based on probability factors of maintenance scenarios (determined in step 4). A convergence limit (if exists) expresses the basic balance cases and could detect the convergence limit of specific factors during maintenance process while the individual attributes (defined in step 3) being updated.	Equation: (3-15)
9	Decision making	Helps to infer the most maintainable design pattern combination by using the derived equations and graphs (defined in step 7). Mathematical tools such as MATLAB, Excel can be used to facilitate computations and graphs generation.	Application examples (subsection 3.8.3) Demonstration page : <a href="http://www.chriskaranikolas.gr/CIBIvsCVP">www.chriskaranikolas.gr/CIBIvsCVP</a>
10	Save the new results for future use	Alternatively, define (update) the new (existing) software quality policy plan regarding specific problem family. Also, flowcharts can be derived and used by designers during repeated implementation of step 9.	Flowchart: Figure 3-21

Referring to steps 6 and 7 in Table 3-6, other factors can be derived from combinations of existing factors targeting computational simplifications and easier graphical representations. Also, new constant factors or parameters can be derived based on specific requirements of the software quality policy plan. Multivariate linear model (MEMOOD) can be helpful for equation derivations as proposed by (Rizvi & Khan, 2010). Moreover, as an intuitive conclusion, metrics that have balance (zero distance decision) cases similar to the convergence limit (in step 8), provide safe and permanent decisions and usually are considered as particularly reliable.

The efficiency and the degree to which the methodology proposed in Table 3-6 can be used to extract alternative comparison models for similar or different design pattern solutions can be further explored. For example, one other comparison case could be the Decorator design pattern (Gamma et al., 1994) which attaches additional responsibilities to CP objects dynamically against the common extension of CP through inheritance. In this case, basic maintenance scenarios could be a new element, new operation, and new responsibility. Furthermore, various comparison cases could be interesting considering the simultaneous implication of VP for the operations. Under this perspective, the proposed formal model for comparing CIBI and CVP implementation alternatives could be considered as the first step in using and testing the suggested methodology.

### 3.9.2 Example of Weighted Effort Measurement

At this level of analysis, many other decision factors can be investigated. As an example, which reveals the flexibility and extensibility of the described methodology, the following model variation is presented.

In most of the cases, the data of part-whole representation problems are declared inside CP classes. In practice, developers usually manage methods' code which acts directly on the data of its class like operations' code in CIBI implementation. This is a common programming technique which is manageable even by less experienced developers. However, in CVP implementation, operations' code which is in the extra VP classes acts indirectly on the data of CP classes. Indirect access on the data of CP classes is achieved through the double dispatching calls of accept method as referred in subsection 3.3.2.3. This is a more complex programming technique which is difficult to be managed by less experienced developers. Furthermore, many other concerns such as debugging issues are introduced because of the use of double dispatching calls. In general, the indirect access on data of different classes through double dispatching calls introduces an extra effort during maintenance. Consequently, it can be assumed that the locality of method interventions, in the same or separate classes/modules, has different (or higher) significance only for CVP implementation.

In the proposed metric (subsection 3.5.3 and step 7 of methodology in Table 3-6), the numbers of methods interventions and affected class (expressing the locality of interventions) for CVP have equal weights (equations (3-9), (3-10)). Instead of this, a new constant factor can be defined based on previous discussion. For example, the equation (3-9) could be reformed as equation (3-24).

$$c_m^S(n_e, CVP) = c_m^{S,A}(n_e, CVP, a_m) + w \cdot c_m^{S,A}(n_e, CVP, a_c) = M + w \cdot M \quad (3-24)$$

The introduced factor  $w \geq 1$  represents the weight of class aspect magnitude which captures the penalty or the extra maintenance effort required for CVP. Thus, for  $w=1$ , the equation (3-24) and (3-9) are equivalent meaning that developers can equally manage the widespread interventions for both IBI and CVP implementations. When  $w > 1$ , the new metric reflects the locality of interventions in a more significant weight for CVP implementation. So, CIBI implementations are preferred (have lower maintenance cost) than CVP implementations due to  $w$  factor presence. By following the other steps of the described methodology, a new set of equations (e.g., (3-10), (3-13), (3-14), (3-16), (3-21), (3-23)) and graphs (e.g., Figure 3-16 to Figure 3-18) can be derived. As a final stage on this example, Figure 3-22 is updated to Figure 3-26. In Figure 3-26, new curved lines have been added which show balance cases (zero distance) for different values of  $w = \{1.0, 1.5, 2.0, 3.0, 4.0\}$  factor. Hence, different values of  $w$  factor cause a relocation of the curve of balance cases.

Referring to Figure 3-26, some interesting observations and interference can be derived about  $w$  factor: a) for  $w=3$ , alternate GUI implementation decision (light purple dot) has switched side and IBI is preferred than VP, thus different comprehension degree of CVP (e.g. defined by  $w$  factor) leads the model to different results; b) as  $w$  factor increases, the balance curve shifts to the left with a deceleration rate. This graph clearly demonstrates that CIBI implementation expands for greater values of  $w$  factor or when comprehension degree of CVP is smaller. In contrast, CVP implementation expands for lower values of  $w$  factor or when comprehension degree of CVP is greater.

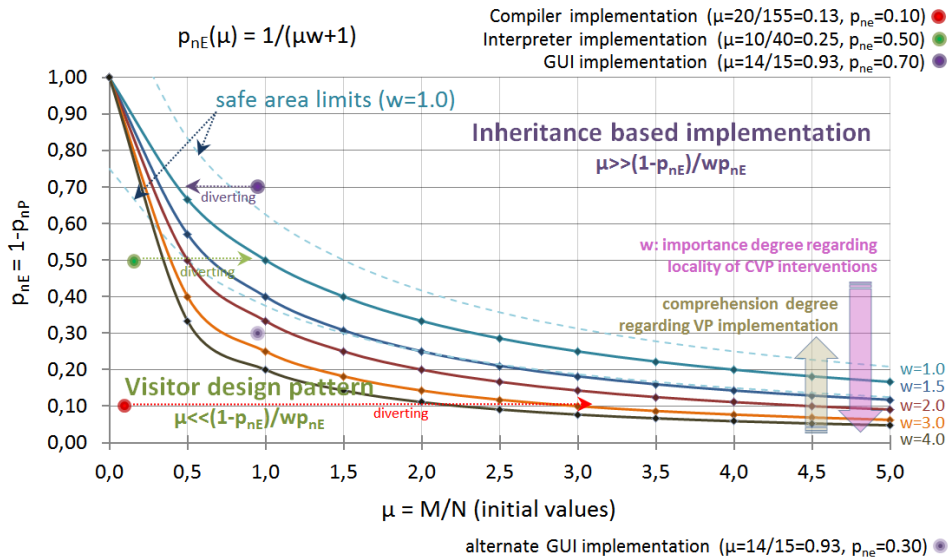


Figure 3-26: Graph of balance cases (equal maintenance cost) for CVP vs CIBI based on  $w$  factor.

Another interpretation about  $w$  factor is related to the concept of software disorder (entropy) and particularly the erosion factor introduced by (Bakota et al., 2012). More specific erosion factor represents the amount of “damage” (decrease in maintainability) caused by changing one line of the code. In a similar manner,  $w$  factor can be considered as the amount of extra effort (increment in maintenance cost) that required in order the method interventions for CVP to be performed concentrated in one class during a particular maintenance scenario. Alternatively, the  $w$  factor could be named as CVP dispersion factor. Following the same logic,  $w$  factor normally is smaller for senior developers who have significant experience and thus they handle widespread interventions of CVP in a better and easier way, dedicating less effort. Also,  $w$  factor normally is smaller for developer teams which use more advanced resources (developing/versioning suites, etc.) and thus they also handle widespread interventions of CVP in a better way, dedicating less time and effort. Normally software companies should try to reduce  $w$  factor by dedicating better resources during software maintenance, reducing overall maintenance cost and effort. Thus, another noticeable conclusion is that CVP implementation expands when companies dedicate more experienced resources (e.g., senior developers) during software maintenance. Consequently, it can be induced that VP is harder (than IBI) to be understood and applied since VP is less suitable or maintainable by less experienced developers. This conclusion confirms the intuition and perhaps the concerns of some developers that VP implementation is harder because of the use of double dispatching calls or the extra visitors’ classes. Apparently, this perception discourages many developers for using VP. Thus, designers and developers should insist on VP comprehension and use it whenever it is recommended in order to avoid the required extra effort during maintenance.

### 3.9.3 Further Discussion

Referring to the previous example of weighted effort measurement, there is a key point. Equation (3-15) still stands since it is independent of  $w$  factor. Thus, during the maintenance process, the design attributes  $N$  and  $M$  increase and factor  $\mu$  tends to limit  $p_{nP}/p_{nE}$ . Therefore, each decision point in the new graph in Figure 3-26 is shifting or diverting horizontally toward the limit of equation (3-15), represented by the initial curve of balance cases for  $w=1$ . Although in this case, the diversion has an increasingly decelerating rate (due to limit’s behavior), a paradox seems to arise which also looks unavoidable. More specifically, for  $w>1$ , a pattern combination may be initially preferred

for a certain number of future additions. However, after the application of several additions, this combination will no longer be the proper choice. That kind of decisions need extra caution, and perhaps an assessment of the range of future additions can be helpful toward proper selection.

The later discussion indicates that the SMC metrics in equations (3-9)-(3-12) ( $w=1$ ) are stable and dominant since the curve of balance cases in Figure 3-16 and Figure 3-22 is identical to the limit in equation (3-15) which represents the convergence of individual characteristics (i.e.,  $N$ ,  $M$ ,  $\mu$ ) during maintenance process. This equivalence is exactly the reason why the proposed model provides straightforward and safe choices independently on the number of future additions that will take place during maintenance process, as showed in subsection 3.5.7. Thus, the proposed approach and its SMC metric can be considered as particularly reliable. As a more general and intuitive conclusion, metrics that have similar balance equations to the above convergence limit are stable, provide safe and permanent decisions, and usually considered as particularly reliable.

## 3.10 Conclusions

### 3.10.1 General Requirements and Limitations

In this subsection, the basic limitations, and some potential threats to validity regarding the proposed approach are briefly discussed.

To derive realistic equations of asymptotic maintenance effort or cost, it is necessary to analyze and precisely understand the way that each design pattern evolves, behaves, and reacts with other patterns in the event of future maintenance scenarios. Furthermore, it is essential to conclude on those maintenance scenarios and design characteristics that are the most influential regarding the pursued quality requirement (i.e., maintainability or modifiability). Without this envision, the model's estimations and consequent design decisions could be subjective or unrealistic. If this is the case, then every possible type of future action (e.g., additions or modifications) as well as the consequences or the impact of these actions can be described in an objective way. Furthermore, the required effort for every possible action type or other quality characteristics can be estimated as well. These quality characteristics include method/code or data locality in source code or in memory, cyclomatic complexity, interface complexity, run or compile time performance estimations, data/memory consumption estimations, design or module complexity, depth of class inheritance, friendly methods, overwritten methods, any other known quality metric (e.g., ISO 9126/25000, McCabe) which can be statically computed by the structure of design pattern, etc. As an example of the variety of such measures is the computational complexity metric (Hills et al., 2011) which tries to measure not only the effort to transform the system, but also the effort to analyze it before applying any transformation.

The asymptotic or static cost of all the characteristics that have been mentioned can be positive or negative or even multiplied by factors and depends on estimations about anyone of them. Furthermore, intermediate, or partial costs can be combined and merged to more general actions, deriving general equations. During this process, reliable predictions or probabilities of different maintenance scenarios can be involved in these equations. In general, the accuracy of this method depends on the accuracy of the distinct asymptotic or static cost estimations and predictions.

It is important that the conclusions and the results of this method are general and based on static predictions and structure analysis of specific design patterns. They are independent to specific program implementations and run time behavior. They are based on static estimations about future modifications and additions, while they capture specific quality characteristics like maintainability and changeability. Therefore, they should be used during the design phase of software before code development to ensure reduced time, effort, and relevant cost of future code modifications, updates, and maintenance.

Another important point is that the proposed method evaluates, in a formal way, two or more combinations of design patterns that have been proposed to address the same significant problem in a different (design) way. Thus, absolute maintenance cost assessments or effort estimations for individual implementations are out of the scope of the proposed approach. Analysis is limited on knowledge of the used design pattern behavior through which distinct maintenance scenarios are derived and analyzed. The main goal of the proposed model is to deliver reliable and proportionally equivalent effort estimations per design alternative for comparison purposes, with no concern regarding the accuracy of the effort magnitudes itself.

Furthermore, the selected maintenance scenarios contribute mainly on the evolution or extension of the system features, retaining initial design (pattern) architecture. Maintenance scenarios that target on structure reformations change the design architecture and the maintainability degree of the system and they are out of scope of the suggested approach.

In the proposed model, the maintenance events are involved in probabilistic analysis considering a repeated (cyclical) arriving pattern, since this is the most common case in prediction models. Thus, a possible threat to validity of the proposed model could be the special case when maintenance scenarios are performed by the developers in a way that significantly deviates from a repeated arriving pattern.

### 3.10.2 Extensions and Further Research

The proposed formal model can be adapted for similar or other design problems. For example, additional cost can be added if more additions or modifications are necessary for a specific structure or when other quality characteristics or metrics should be captured. The proposed approach or an adapted version can be combined and analyzed in conjunction with other methods on case studies (e.g. in (R. S. Pressman, 2001)) to determine the relationship between static asymptotic cost of design patterns and external quality factors such as reusability, maintainability, testability and adaptability. The adaptation and implementation of the proposed method in other problems and patterns depends on many factors such as the type of the problem, scale of structure or/and operations, complexity of operations, quality or other standards-requirements, other collaborated patterns, time, cost plan, etc.

Furthermore, the proposed formal model could be used as a guide by many tools, including aspect-oriented programming tools that generate code, templates or libraries from a higher level language or through a visual environment (e.g. (Dascalu et al., 2005; B. C. d. S. Oliveira et al., 2008; VanDrunen & Palsberg, 2004)) to compare and propose appropriate design pattern combinations. An example is ANTLR (Parr, 2013), a powerful and flexible tool for scanning and parsing formal languages that by default generates initial code for CP, VP and Listener design pattern (a differentiation of VP) based on grammar elements. In this case the use of CIBI or other design patterns could be suggested through the use of the proposed model based on specific grammar attributes. In addition, the proposed mathematical approach can be used as a general framework for estimating and comparing similar or different design patterns especially on the field of Pattern Languages of Programs.

Moreover, the proposed model could be used to evaluate and propose efficient pattern combinations based on code's structure through real-time coding intelligence tools during code development process. Furthermore, the proposed method can be used for source code generation in the generic field of product line engineering (Völter, 2003). More specifically, the used design pattern and specific characteristics of a problem could be derived through a properly formatted model (e.g. class diagram, XML) or DSL (Zdun & Strembeck, 2009) or through a metamodel of the software specifications. In this case, the proposed method can be implemented during the process of model analysis and code generation.

### 3.10.3 Overall Assessment

Selecting between Visitor design pattern and inheritance-based implementation on a part-hole representation during software architecture design is crucial since software maintenance should adapt to the initial architecture. Decisions made at software architecture definition stage heavily affect maintainability and changeability of software and related time, effort, and cost of software maintenance.

The proposed approach suggests a documented methodology to support object-oriented design pattern analysis. Software quality metrics can be derived and computed directly from design descriptions of well-known design pattern combinations by analyzing their structural behavior and evolution pattern. The model returns effort estimations well fitted and sensitive to specific design characteristics and attributes that distinguish a specific system as an instance of the general and significant problem. Using the proposed model, specific characteristics such as design attributes of given problems are considered allowing selection of proper design pattern combinations at an early stage of the design process before code development.

The analysis of the proposed model indicates that different design pattern combinations have a significant effect on software quality and its maintenance perspectives. Furthermore, the progressive and probabilistic analysis verify the same significant effect for a large number of future modifications during software maintenance. Moreover, the analysis proves that different metrics aspects and weights have a significant impact on progressive computations with respect to initial attributes of a design problem and should be carefully defined. Also, it has been indicated that probability analysis over maintenance scenarios has a decisive role on maintenance cost and effort estimation.

The proposed model can be easily implemented in software to support behavior analysis and relevant design decisions among Composite, Visitor design patterns and inheritance-based implementation, providing a visualization for almost the complete solutions or design space of the significant problem of recursive part-whole aggregations. The application of the proposed approach reveals the usability and extensibility of the suggested methodology considering different or alternate metrics, design alternatives, and quality characteristics.

## 4 Modeling Software Evolution

### 4.1 Chapter Overview

In this chapter, a systematic modeling method for the derivation of formal comparison models of different implementation alternatives is discussed. The models can be used for the early evaluation of the design alternatives with regards to maintainability. The method is based on a novel theory focusing on the progressive evolution of a system during the maintenance process. The proposed approach analyzes the software’s expansion tendency through the formulation of system’s size change rates using differential equations in a two-level integration process. More specifically, the analysis focus on change rates of individual structural attributes of each design combination under consideration. Furthermore, the method predicts the required effort during the maintenance process by using the adaptable Structural Maintenance Cost (SMC) metric (Karanikolas, Dimitroulakos, & Masselos, 2017), introduced in chapter 3. This metric is mostly inspired by software entropy concept (Bakota et al., 2012). The metric captures the structural expansion behavior of each design combination under evaluation. The structural expansion is quantitatively expressed in terms of the number of method interventions and the number of classes/modules that are affected, for basic maintenance scenarios and their probabilities. Moreover, an alternative technique based on the ‘ripple effect’ concept (Turver & Munro, 1994) for deriving basic SMC metrics is discussed. In this way, the required effort is predicted in a formal and deterministic way, limiting the ambiguity imposed by the stochastic nature of the maintenance process. The generated formal models are general and reusable, while they can be easily implemented in software and repeatedly applied during the early stage of software design before code development.

The proposed modeling method has been illustrated and evaluated on the important and frequently tackled decision problem between the design combinations of Visitor design pattern and Composition design pattern for data structures (both serving recursive part-whole aggregations) as described in subsection 3.3. In this case, the two major types of maintenance scenarios effect on i) composition’s elements and ii) different operations over these elements. The numbers of initial elements and operations are considered as quantitative structural attributes. The applicability of the derived formal modes is demonstrated over the practical examples of Interpreter and Graphic User Interface (GUI) implementations as defined in subsection 1.2.3.

Furthermore, a detail modeling framework of the introduced modeling method has been implemented in the form of MATLAB code and data structures. This framework accelerates and supports the derivation of formal models as well as the generation of relevant graphs in a dynamic way through properly parametrized scripts. The effectiveness of the framework is demonstrated on the general problem of recursive part-whole aggregations.

The context of this chapter is based on the motivation examples in chapter 1, the related work in chapter 2, and the significant design problem of part-whole representations in chapter 3. The rest of this chapter is organized as follows. Subsection 4.2 discusses the theoretical background of the significant design problem under study. Subsection 4.3 introduces the proposed approach and the modeling method. Subsection 4.4 presents a general and formal implementation of the modeling method in MATLAB structures. Subsection 4.5 provides the validation evidence that support the introduced modeling method and derived formal models. Finally, in subsection 4.6, the model’s validity challenges, limitations, future research issues, and conclusions are presented.

## 4.2 Background of General Decision Problem

An example of a general, significant, and frequently tackled design problem is referred to the recursive implementation of various types of operations upon part-whole aggregations of different types of elements as presented in Table 1-1 of chapter 1, and Table 4-1. This kind of structures is encountered in a wide spectrum of critical systems such as compilers, interpreters, GUI, hierarchical presentation frameworks, DSL, and CAD as their principal design background. Usually, each design alternative is a combination of well-known design patterns. Visitor and Composite are examples of established design patterns which combined can provide implementations of part-whole aggregations. The evolution of software during its maintenance is strongly related and mostly determined by the behavior of the engaged design patterns in future changes or major maintenance scenarios. Referring to the general problem of part-whole aggregations, events like adding or updating or debugging a new or existing type of element constitute a major maintenance scenario. Similar events referred to a new or existing type of operation is another major scenario. In principle, an event is characterized as major maintenance scenario when fulfills the following criteria: a) has significant impact concerning the pursued quality attribute (i.e., maintainability), b) affects or changes the principal design attributes of the engaged design patterns (e.g., number of elements or operations of a structure), c) is neither too abstract nor too specific allowing its application on the early design stage before code development (i.e., encompasses various resembling sub-activities or changes such as adding, updating, and de-bugging concerning the maintenance of a discrete family of design elements with common characteristics such as structure elements, and operations), and d) has recurring nature or considerable possibility to repeatedly occur during maintenance, as further discussed in subsection 3.4.4. In this general design problem, the number of initial elements and operations are conceived as basic design attributes which define a specific problem as an instance of the general problem. Such design attributes are usually referred to problem’s logical entities which are represented by design patterns’ components such as methods, classes, and modules. Furthermore, during maintenance, several of those initial design attributes are updated according to the behavior of the engaged design patterns based on the individual probabilities of major maintenance scenarios. Scenarios’ probabilities are assessments according to the scope of each specific problem.

Table 4-1: Example of Interpreter Software Specifications for the General Problem of Part-Whole Representations.

<b>Analysis of the general problem</b>	<b>Indicative, practical example of a specific problem as an instance of the significant and general problem referred to the recursive implementation of various type of operations upon different types of elements of part-whole aggregations (compositions)</b>
<b>Initial design attributes</b> <ul style="list-style-type: none"> <li>• number of initial elements</li> <li>• number of initial operations conceived as basic design attributes which define a specific problem as an instance of the general problem</li> </ul>	<b>Interpreter</b> implementation for a new custom (extendable) DSL language <ul style="list-style-type: none"> <li>▪ <b>40 initial types of elements</b> (parse-tree nodes derived from a custom BNF grammar such as terminal–nonterminal symbols, identifiers, etc.), and</li> <li>▪ <b>10 initial operations</b> (type checking, code generation, executing, etc.) acting on elements of parse-tree,</li> </ul>
<b>Major maintenance scenarios</b> <ul style="list-style-type: none"> <li>• Adding/updating/debugging a new/existing Element</li> <li>• Adding/updating/debugging a new/existing Operation</li> </ul> <b>Probabilities</b> of major scenarios as assessments according to the scope of each specific problem	since DSL is custom and extendable, both structure and operations could be extended during maintenance by equal probabilities (50%-50%)
Description of the <b>change impact for major maintenance scenarios</b> during maintenance  A possible <b>wrong decision</b> during design stage, before code development, has a serious impact	A <b>wrong selection</b> of the <b>Inheritance-based implementation</b> into Composite’s elements (design combination) requires: <ul style="list-style-type: none"> <li>▪ 40 new methods in 40 different classes for a single operation addition and</li> <li>▪ 1 new class with 10 new methods for a single element addition</li> <li>▪ which <b>overall requires more maintenance effort</b></li> </ul> <b>Instead</b> of the most beneficial alternative of <b>Visitor</b> over Composite’s elements (design combination) which requires:



<b>Analysis of the general problem</b>	<b>Indicative, practical example of a specific problem as an instance of the significant and general problem referred to the recursive implementation of various type of operations upon different types of elements of part-whole aggregations (compositions)</b>
on system quality evaluation, concerning either: (a) the <b>increased (wasted) effort/cost</b> during maintenance, or	<ul style="list-style-type: none"> <li>■ 40 new methods to be placed on a single class for a single operation addition and</li> <li>■ (only) 10 new methods to be placed in 10 different classes for a single element addition</li> </ul> This is a <b>difficult choice</b> due to structure and operation expandability and there is no clear advantage for arbitrary scenarios' probabilities
(b) the <b>costly</b> setback, after code development, in the design stage of software lifecycle which requires <b>redesign and refactoring of the existing</b> code that usually <b>demand a significant amount of extra (wasted) effort</b>	<ul style="list-style-type: none"> <li>■ 10 methods for each of 40 (composite) classes, thus totally 400 methods (including their references and calls) be revised and moved in 10 different (visitor) classes in groups of 40 methods</li> </ul>

Initial source code will be generated by a parser tool such as Bison or ANTLR (Parr, 2013). Interpreter implementation is a real case description

The practical example of the Interpreter implementation in Table 1-1 and Table 4-1, as a specific instance of the significant and general problem of recursive part-whole aggregations, highlights the negative consequences of wrong design selections as well as the necessity for a rigorous modeling method that can support early selection among design alternatives regarding their maintainability perspective.

The Inheritance Based Implementation into Composition (CIBI) and Visitor upon Composition (CVP) design combinations have opposite characteristics regarding their maintainability perspective, and thus they have been discussed in the context of many studies, like the well-known Expression Problem (B. C. d. S. Oliveira & Cook, 2012; Torgersen, 2004; Wang & Oliveira, 2016; Zenger & Odersky, 2005). Therefore, the selection between CIBI and CVP is rather a crucial and challenging decision since many major systems like compilers, interpreters, high-level synthesis, Domain Specific Languages, Intermediate Representations, GUIs, hierarchical frameworks, are designed over recursive composite structures. For the sake of completeness and uniformity with the context of chapters 1 and 3, a brief presentation of the theoretical background of the general selection problem, as presented in Table 4-1 follows. A summary of the relevant concepts, components, and terms of the analysis in correspondence to the significant problem of part-whole aggregations is presented in Table 4-2.

Table 4-2: Correspondence of Concepts, Components, and Terms to the General Decision Problem of Part-Whole Representations.

Basic concepts, components, and terms of the analysis		Correspondence to the significant general problem, referred to the recursive implementation of various types of operations upon different types of elements of part-whole aggregations (compositions)	
		Description of specialized concept	Notation
1) Introduction of Modeling Method	<b>Distinct design patterns</b> engaged in design combinations	<ul style="list-style-type: none"> <li>■ <b>Composite</b> (design Pattern) represents recursive part-whole aggregations or structures of elements</li> <li>■ <b>Visitor</b> (design Pattern) links operations to different type of elements of a Composition</li> <li>■ <b>Inheritance-Based Implementation</b> incorporates operations inside Composition's elements</li> </ul>	CP VP IBI
	<b>Combinations of design patterns</b> (as solution alternatives of the general problem)	<ul style="list-style-type: none"> <li>■ <b>Visitor</b> over Composite's elements</li> <li>■ <b>Inheritance-based</b> implementation into Composite's elements</li> </ul>	CVP = (CP+VP) CIBI = (CP+IBI)
	Problem's <b>initial design attributes</b>	<ul style="list-style-type: none"> <li>■ Number of <b>initial Elements</b> of Composition</li> <li>■ Number of <b>initial Operations</b> (or Processes) acting on Elements</li> </ul>	N M
	<b>Basic classes</b> of changes or <b>maintenance scenarios</b> and their individual probabilities	probability of <b>Adding</b> /updating/debugging a new/existing <ul style="list-style-type: none"> <li>■ <b>Element</b> type</li> <li>■ <b>Operation</b> (or <b>Process</b>) type</li> </ul>	$p_{nE} = 1 - p_{nP}$ $p_{nP} = 1 - p_{nE}$
	<b>Number of future scenarios' applications</b> Returned predictions of <b>expected maintenance effort</b> (per design alternative)	<ul style="list-style-type: none"> <li>■ Number of maintenance scenarios' applications during software evolution/maintenance</li> <li>■ <b>Total required effort</b> prediction returned by formal model per design alternative</li> </ul>	$\lambda$ $C_m(CVP, N, M, p_{nE}, 1 - p_{nE}, \lambda)$ $C_m(CIBI, N, M, p_{nE}, 1 - p_{nE}, \lambda)$
2) Illustration through	a. Derivation of <b>fundamental effort metrics</b> (for each design alternative and scenario type for a single [ $\lambda=1$ ] scenario application)	<ul style="list-style-type: none"> <li>■ New Element (<math>p_{nE}=1.0, p_{nP}=0.0</math>) on CVP for <math>\lambda=1</math></li> <li>■ New Operation (<math>p_{nE}=0.0, p_{nP}=1.0</math>) on CVP for <math>\lambda=1</math></li> <li>■ New Element (<math>p_{nE}=1.0, p_{nP}=0.0</math>) on CIBI for <math>\lambda=1</math></li> <li>■ New Operation (<math>p_{nE}=0.0, p_{nP}=1.0</math>) on CIBI for <math>\lambda=1</math></li> <li>■ CVP design combination (solution)</li> <li>■ CIBI design combination (solution)</li> </ul>	$C_m(CVP, N, M, 1.0, 0.0, 1)$ $C_m(CVP, N, M, 0.0, 1.0, 1)$ $C_m(CIBI, N, M, 1.0, 0.0, 1)$ $C_m(CIBI, N, M, 0.0, 1.0, 1)$ $C_m(CVP, N, M, p_{nE}, 1 - p_{nE}, \lambda)$ $C_m(CIBI, N, M, p_{nE}, 1 - p_{nE}, \lambda)$

<b>Basic concepts, components, and terms of the analysis</b> b. Derivation of <b>formal prediction models</b> (for each design alternative and any number of applied scenarios $\lambda$ ) c. <b>Application</b> of the derived formal models on <b>practical examples</b> of specific problems	<b>Correspondence to the significant general problem, referred to the recursive implementation of various types of operations upon different types of elements of part-whole aggregations (compositions)</b>	
	<b>Description of specialized concept</b> ■ <b>Interpreter</b> (N=40, M=10, $p_{nE}=0.5$ ) ■ <b>GUI</b> (N=15, M=14, $p_{nE}=0.7$ )	<b>Notation</b> $C_m(CVP/CIBI, 40, 10, 0.5, 0.5 \lambda)$ $C_m(CVP/CIBI, 15, 14, 0.7, 0.3 \lambda)$

**CP:** The main intent of the Composite design Pattern is to compose objects into tree structures to represent part-whole hierarchies. CP lets clients treat individual objects and compositions of objects uniformly (Gamma et al., 1994). CP is the basis of both design combinations and presented on both sides in Figure 4-1. The number of distinct nodes/objects or element types, which can be instantiated or represented by CP, is equal to the number of leaf classes of the hierarchy, denoted as N in Figure 4-1.

**CIBI (IBI in CP):** In a Composite structure, the Inheritance Based Implementation (IBI) can be used, as presented on the left side in Figure 4-1. This straightforward object-oriented approach is based on the inheritance attribute and can be considered even as a naïve, and similar to Interpreter design pattern (Gamma et al., 1994; Hills et al., 2011). In the general case, all distinct operations, denoted as M in Figure 4-1, are declared as virtual methods in the abstract root class of the hierarchy. The implementation of every distinct operation (method) is placed in each distinct object (leaf) class of the hierarchy. This pattern combination makes adding new types of nodes (elements) easier (Gamma et al., 1994) thanks to the concentration (locality) of the related interventions in a single class.

**CVP (VP over CP):** Visitor design pattern (VP) can be used over CP as presented on the right side in Figure 4-1, and further analyzed in (Alexandrescu, 2001; Gamma et al., 1994; B. C. d. S. Oliveira et al., 2008; Palsberg & Jay, 1998; Visser, 2001). In the general case, for every distinct type of CP node, a new virtual method is declared in an abstract root class called Visitor. In addition, for every distinct operation, a new subclass is created which includes all the implementations of the methods of distinct node types for this specific operation. CVP approach rearranges the methods of all distinct operations from CP sub-

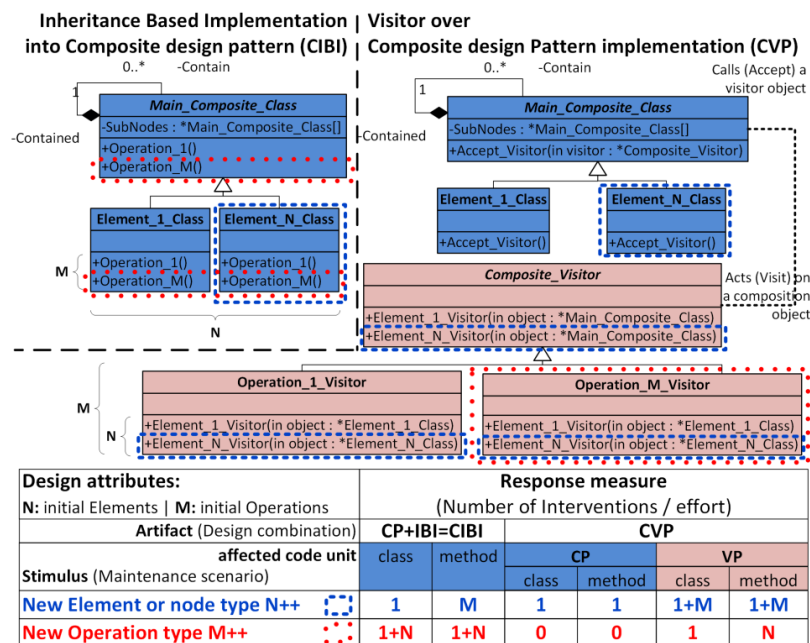


Figure 4-1: Conceptual UML class-diagram of CIBI (Inheritance-Based Implementation inside a Composition) and CVP (Visitor design pattern over Composition’s pattern) design combinations.

classes into the new visitor sub-classes. In contrast with IBI, VP makes adding new operations easier (Gamma et al., 1994) thanks to the increased cohesion or the concentration (locality) of the related interventions in a single visitor class.

The numbers of distinct element types and distinct operation types, denoted as N and M in Figure 4-1, are conceived as key design attributes. Through these attributes, the number of required method and class interventions can be quantitatively expressed in the event of major maintenance scenarios as summarized in memo table in Figure 4-1 and analyzed later. Thus, the selection of the most maintainable combination is rather a difficult decision problem, especially in the case where elements and operations are both extendable during the maintenance process as showed in Table 1-1 and Table 4-1.

## 4.3 Modeling Approach

### 4.3.1 Designs for Change Principle

The first step in controlling software changeability or maintainability degree is by applying the rule “design for change” as discussed by Parnas (David Lorge Parnas, 1994). Design for change can be achieved by trying to categorize the changes that are likely to occur over the “lifetime” of the product. Since actual changes cannot be precisely predicted, the assessments will be about classes of resembling changes. In principle, this design rule implies that logical entities that are most likely to change are “confined” to a small or grouped amount of code so that if those entities do change, only a small amount of code would be affected. This is exactly one of the fundamental reasons to use well-known design patterns in order to ease future enhancements (Bieman et al., 2001). Furthermore, since it is impossible to make everything equally easy to change, it is important to estimate the probabilities of each class of changes. However, even if these probabilities have been estimated, in most of the cases, it is not obvious how specific design combinations are evolved during software maintenance with respect to these probabilities. Hence, selection among design alternatives is usually left to experts’ intuition, thus introducing high-risk for suboptimal choices and low maintainability degree. Identifying what can change, what is the likelihood of the change, and what is the impact or cost of the change is in the core of architectural design process towards modifiability as suggested in (Bass et al., 2012).

### 4.3.2 Corresponding Architectural Design Principles

In general, the object-oriented (low-level) design is a sub-domain of the Software Architectural (high-level) design of systems (Bass et al., 2012). The general architectural design principles are roughly analyzed in subsection 3.2.1 and visualized in Figure 3-1. However, a direct correspondence exists regarding the notation, terms, and concepts between the (low-level) object-oriented design and (high-level) architectural design as illustrated in Figure 4-2. For example, the design for change principle corresponds to the Quality Attribute requirement of modifiability or maintainability, expressing the ability of the system to support changes. The arriving (classes of resembling) events or maintenance scenarios correspond to Stimulus, thus to the requested modifications which affect or change the Artifacts. The design combinations under assessment, usually represented by UML models of classes and methods, correspond to Artifacts generally represented by design models of system’s modules and components. The implementation of an arriving maintenance scenario or Stimulus corresponds to the Response, thus to the required specific modifications on an Artifact.

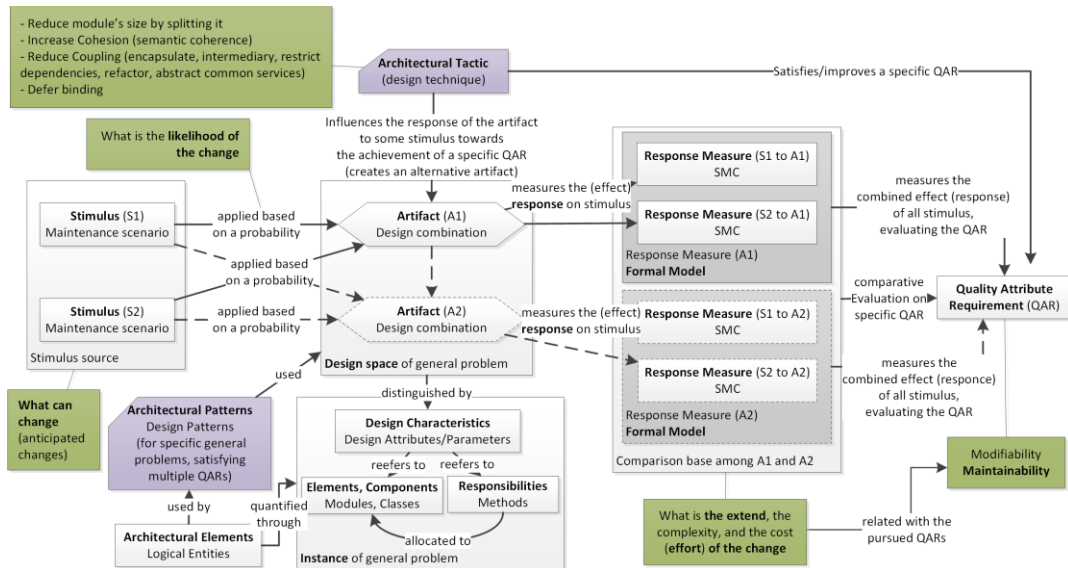


Figure 4-2: Conceptual representation of the architectural design principles connected to the proposed theory and Modeling Method

The Response Measure refers to the quantitative measurement of the effect or the Response of the Stimulus, focusing on the aspect of the pursued Quality Attribute Requirement. Modifiability or maintainability as a Quality Attribute Requirement is influenced by the application of Architectural Tactics such as splitting or rearranging responsibilities to increase cohesion and reduce coupling among the model's logical entities. The result of this architectural design process leads to alternate design models or Artifacts, aiming at improving their Response Measure to a set of Stimulus or maintenance scenarios. Architectural design is also about selecting among design alternatives or Artifacts the one with the most satisfactory overall Response Measure. Since Modifiability refers to the amount of code that should be affected, the Response Measure refers to the amount of code (e.g., number of entities) of an Artifact that would be affected as a result of an arriving Stimulus or maintenance scenario.

However, the overall quality assessment of an Artifact requires the measurement of the Response on a set of different Stimulus, each affecting the Artifact's entities in different and conflicting ways. Thus, a combined analysis of the effects (Responses) of all Stimulus based on their probabilities for each design alternative or Artifact is required. The Response of an Artifact to a Stimulus is quantitatively expressed by the Response Measure the outcome of which may be subject to several design characteristics and properties of the Artifact. As a result, the Response of an Artifact to a Stimulus may be also sensitive to specific model's design attributes such as the initial number of instances of an entity type. These design attributes of a model or Artifact identify a specific problem as an instance of the general problem. The proposed modeling method derives formal models for each design alternative or Artifact of a general design problem. This allows the selection of the Artifact with the best combined Response Measure to all possible major Stimulus for specific design attributes, concerning the pursued Quality Attribute Requirement of modifiability or maintainability.

From a different point of view concentrating on maintainability perspective, software architecture is in accordance with the 'design for change' principle. It tries to answer a) what can change or what are the anticipated changes or maintenance events, b) what is the likelihood or probability of each change, and c) what is the impact or the extend or the required effort of each change. The proposed modeling method and derived formal models help software engineers to resolve the trade-offs among design alternatives and find the sweet spot within the enormous architectural design space that satisfies the pursued QAR

by finding the design alternative with the optimal combined response (measure) for all major stimuluses.

### 4.3.3 Characteristics of SMC Effort Metric

In the context of this study, the Structural Maintenance Cost or SMC (Karanikolas et al., 2017), introduced in chapter 3, is used as fundamental effort metric. For the sake of completeness, the principal characteristics of the SMC metric are briefly discussed in this subsection. In general, fewer method interventions or less affected classes, for a specific maintenance scenario, imply lower code dispersion, entropy, complexity, crosscutting degree, coupling, and higher cohesion. All these concepts are directly related to the intuition that developers will have increased work keeping track of changes that are performed across many source files or any other code unit or segment (Hassan, 2009). SMC metric enforces the measurement process by simultaneously counting different types of affected code segments, expressing by this way not only the number of interventions (e.g., affected methods) but the locality or scattering degree of these interventions as well (e.g., expressed by the number of affected classes). In particular, the measurement of scattering degree magnifies the impact of the relevant interventions with respect to critical characteristics of modifiability such as coupling and cohesion among code segments of logical entities. Thus, SMC metric encloses all previous concepts in an indirect but sufficient way providing an adequate graduation even in the absence of source code. Furthermore, SMC metric ignores the actual size (lines of code) of each elementary method intervention since this code i) is not available in the design stage, and ii) in a long-term perspective it has no significant impact on the final effort assessment. More specifically, it is supported that the actual code and size of each method intervention refers to the business logic of the solution (not its design perspective) and thus, it would be common or similar for all design alternatives under comparison, hence neutral concerning the decision-making. Finally, the SMC metric concentrates only on expansion scenarios (additions) and does not take under consideration alternate scenarios such as editing, debugging, and deleting, supporting that the expansion (new features) scenarios have a dominant role in the progressive evolution of the system and thus to effort assessment, as reported in chapter 3 and analyzed in subsection 4.3.5.

Under the view of architectural design principles, the SMC metric corresponds to the response measure (or the required effort in terms of number of interventions) representing a quantitative assessment of the extend of the required changes of a particular stimulus (major maintenance scenario) to a specific artifact (design alternative) as visualized in Figure 4-2. In other words, SMC metric quantitatively expresses what is the extend, the complexity, and the cost (effort) of an anticipating change scenario (stimulus) during maintenance.

### 4.3.4 Fundamental SMC Effort Metric Derivation

In this subsection, a more systematic derivation approach of SMC metrics is presented in Figure 4-3, which is a further and detailed specialization of the ‘ripple effect’ concept as introduced in (Turver & Munro, 1994). The concepts of ‘ripple effect’ and ‘ripple propagation’ are related to the impact analysis of a particular change or scenario during maintenance. More specifically, the impact analysis evaluates in layers the consequences of a particular change up on a specific design combination to predict the required effort. The more a change causes other changes to be made, in general, the higher the required effort. The outcome of this process is a consequence flow or logical model as depicted in Figure 4-3 which is referred to the Interpreter implementation of the general decision problem (described in Table 1-1, Table 4-3, and presented in Figure 4-1) about the impact of possible maintenance scenarios on the CVP design combination.

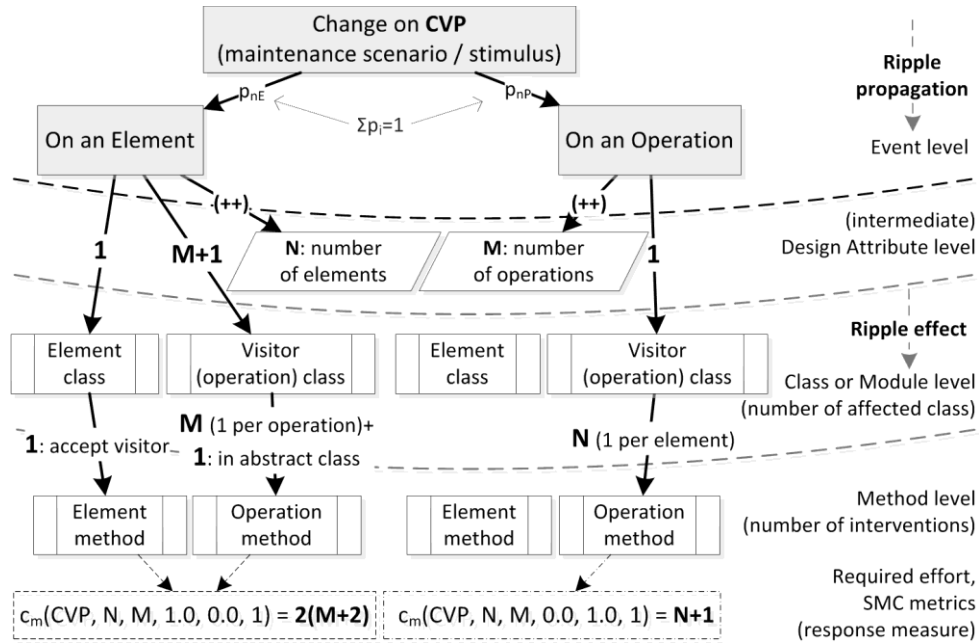


Figure 4-3: Consequence flow (logical model) during impact analysis for changes on the Visitor over Composite design combination (CVP).

More specifically, at the top of Figure 4-3 the major maintenance scenarios are depicted as the Event Layer or Level. In this example, the possible scenarios are referred to the addition either of one element or one operation with their probabilities as their edges' weights ( $p_{nE}$ ,  $p_{nO}$ ).

Below the Event Layer is the intermediate Attribute Level that exposes the design attributes and the way in which these are affected by each scenario type, denoted in their edges' labels. For instance, the scenario of a new element ( $n_E$ ) increases the number of Composition's elements ( $N$ ) by one ( $++$ ). The values of these design attributes contribute to the evaluation of the SMC metric by the following layers.

Under the Attribute Level is the Class or Module Level that depicts the type of classes that effected by each scenario type. The number of classes that are affected by each change type is stated in their edges' labels. For instance, the scenario of a new element ( $n_E$ ) affects one element's class, the abstract Visitor's class, and  $M$  operation's (Visitor) sub-classes, thus totally  $M+2$  effected classes.

Next, the Method Level depicts the type of methods that effected as a consequence of an intervention in each of the class types of previous Class Level. For instance, referring to the scenario of a new element ( $n_E$ ), the single intervention in the element's class causes an intervention on the relative 'accept visitor' method, and each of  $M+1$  interventions in the operation's (Visitor) class causes one method intervention, thus totally  $M+2$  method's interventions.

Hence, the SMC metric for a "new element" scenario on CVP combination is the sum of all sub-effects lined up into the scenario's virtual branch across different layers, formally stated as  $c_m(CVP, N, M, 1.0, 0.0, 1) = 2(M+2)$  according to the notation in Table 4-2. Similarly, the SMC metric for a "new operation" scenario is  $c_m(CVP, N, M, 0.0, 1.0, 1) = N+1$  or  $N$  method interventions in one single (visitor) class. The memo table and the class diagrams in Figure 4-1 give an extra insight on how SMC metrics are interpreted.

A similar consequence flow or logical model for CIBI design combination is presented in Figure 4-4. In this case, the implementation of both elements and operations are placed inside the single design pattern of Composition, thus this model includes fewer type of classes and methods.

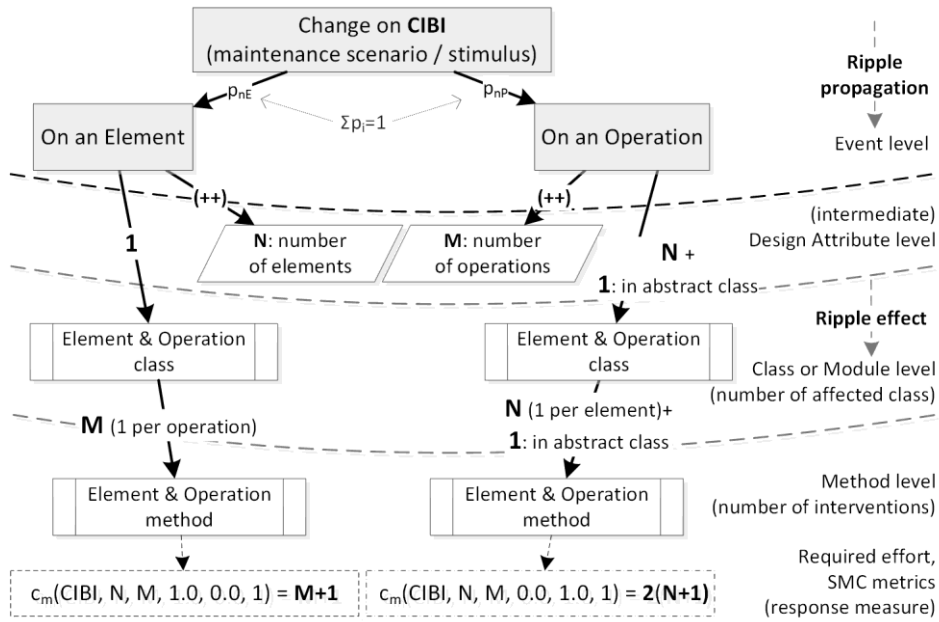


Figure 4-4: Consequence flow (logical model) during impact analysis for changes on the Inheritance-Based Implementation into Composite design combination (CIBI).

Both consequence flow or logical model for CVP and CIBI design combinations provide the pattern for the complete set of the fundamental or SMC metric equations per design combination, as summarized in Table 4-3. From a different perspective, the SMC metrics can be conceived as a set of elementary principles which quantitatively signify how the added code or the related effort is allocated or spread in the context of an evolution pattern that reshapes the code during maintenance, as suggested in (Stopford & Counsell, 2008). Thus, the effect of each single change is quantitatively depicted by the extracted SMC metrics. Conceptually, the whole transformation process is a stratified cause-effect analysis trying to quantify change-effects.

Table 4-3: Equations of Fundamental Effort Metrics of CIBI vs. CVP General Decision Problem

Description of maintenance scenario	Equation of total effort for a single scenario application	Affected design attributes
New element on CVP	$c_m(\text{CVP}, N, M, 1, 0, 1) = 2(M+2)$	N++
New operation on CVP	$c_m(\text{CVP}, N, M, 0, 1, 1) = N+1$	M++
New element on CIBI	$c_m(\text{CIBI}, N, M, 1, 0, 1) = M+1$	N++
New operation on CIBI	$c_m(\text{CIBI}, N, M, 0, 1, 1) = 2(N+1)$	M++

### 4.3.5 Software Expansion Concept

In this subsection, the conceptualization of the proposed theory inspired and supported by several empirical observations is documented. The description emphasizes into the trends and relations among the essential concepts of system's size, required effort, and maintainability degree, concerning software maintenance process.

According to Lehman's first and second laws (Meir M. Lehman et al., 1997), referred as 'Continuing Change' and 'Continuing Growth', a software system has the trend to expand over their lifetime, since it must be continually adapted to maintain user satisfaction. Several studies provide empirical support of these laws (Bakota et al., 2012; Barry et al., 2007; C. R. Cook & Roesch, 1994; H. Gall et al., 1997; Jazayeri, 2002; M. M. Lehman et al., 1998; Yuen, 1988) mostly based on analysis of large repository of historical data. It is noticeable that 66% of changes enhancing an existing feature do so by adding a

new feature as reported in (Paixao et al., 2017). Thus, the system's size expressed in terms of code size has a positive change rate during the maintenance process.

The required maintenance effort is proportional (linearly) related to the size of the code under adjustment as supported in (Araújo et al., 2012; Bengtsson & Bosch, 1999; Bosch & Bengtsson, 2001; Dolado, 2001; Hayes et al., 2004; Hayes & Zhao, 2005; Jabangwe et al., 2015; Jazayeri, 2002; Zhang, 2008). Furthermore, size properties are identified as the most ideal predictors of effort as concluded by Briand et al. (L. C. Briand et al., 2002). Thus, since the system's size has a positive change rate, the required effort during the maintenance process has a positive change rate as well.

Software maintainability can be expressed through the estimation of the required effort during the maintenance process, as concluded in (Riaz et al., 2009a) and suggested in (Heitlager, Kuipers, & Visser, 2007b). Thus, more maintenance effort corresponds to less maintainable software, indicating an inverse relation between required effort and software maintainability degree. Since the system's size and required effort have positive change rates, software maintainability degree has a negative change rate over time as confirmed by the software entropy approach in (Bakota et al., 2012) or by measurements of industrial systems in (Land, 2002). Conclusively, the analysis of the system's size change rate provides a concrete theory about the quantification of the required effort toward maintainability assessment.

#### 4.3.6 Analysis of System's Size Change Rate

In this subsection, the concept of the system's size change rate is decomposed to its underlying factors. In general, the system's size is expanded as new code segments are added, or existing code segments are enhanced. These incoming code segments usually correspond to individual actions or maintenance scenarios (e.g., additions, modifications, debugging) as roughly visualized in large lined up code blocks in Figure 4-5. Each action type affects the logical entities (e.g., elements or operations) of existing software architecture or design pattern combinations. The current state of the affected logical entities is expressed by key design attributes such as number of elements or operations (e.g. [n] in Figure 4-5). According to the introduced approach, during the design stage, the actual size of such action code segments can be approximated by the number of the required interventions. Thus, each individual action and its corresponding code segment can be considered as a set of smaller or elementary code segments, representing the required distinct interventions (e.g., method interventions) as depicted in small stackable code bricks per action in Figure 4-5. The number of those interventions for each action type can be approximated based on the current state of key design attributes. Furthermore, the allocation of the elementary code segments is guided by the design architecture of the system and can be expressed by the number of (different and larger) effected code units such as classes, as roughly visualized in Figure 4-5. This number reflects the locality or the scattering degree of the required interventions for a given action type, capturing major characteristics of modifiability such as cohesion and coupling degree among logical entities. Thus, the more the affected classes for a given action type, the more the required effort for the elementary interventions to be completed. In the context of this study, the number of elementary method interventions and the number of affected classes per individual action or maintenance scenario are expressed through the derived equations of fundamental effort metrics in Table 4-3.

As an action's code segment is entering into the system, the current state of design attributes is affected, usually by increasing their values due to system's innate expansion trend. Consequently, the size of following actions that is based on current values of those design attributes is increasing too. This increasing trend is roughly depicted by the increased size of incoming action code segments in Figure 4-5. In general, the incoming code segments are directly related to the required maintenance effort, which can be considered as incoming energy flow (developers' effort) to a dynamic system of which the



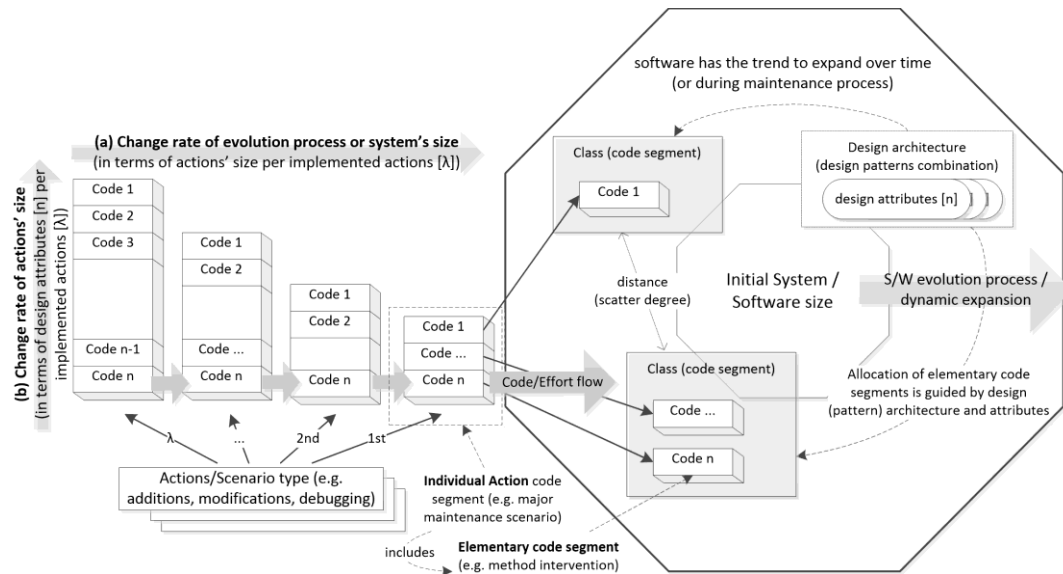


Figure 4-5: Abstract representation of software dynamic expansion during the maintenance process.

change rate could be described through differential equations. This idea is not surprising because in many real-world systems, changes occur, and we want to predict future behavior on the basis of how current values change. The dynamic behavior of such systems can be modeled through differential equations. Hence, the analysis of the system's size change rate is reduced to the study of incoming actions' size change rate, as roughly visualized by the horizontal left-to-right flow (a) in Figure 4-5.

In addition, the individual quantitative design attributes (e.g., number of elements or operations), which are generally increased determining the number of elementary interventions and following actions' size, can be considered as dynamic sub-systems of which the change rates could be described through differential equations as well. Thus, the analysis of each action's size change rate is reduced to the study of individual design attributes change rate as roughly depicted by the vertical bottom-to-up flow (b) in Figure 4-5.

Conclusively, the analysis of system's size change rate is reduced to the analysis of actions' size change rate, which further reduced to the analysis of individual design attributes change rate per implemented action or maintenance scenario.

### 4.3.7 Structural Evolution through Change Rates

In the next subsections, an innovative approach for deriving formal comparison models is presented. The approach further formalizes the software expansion concept and the analysis of system's size change rate presented in previous subsections 4.3.5 and 4.3.6. Through the analysis of those change rates, software maintenance or evolution is holistically approached more as a going concern than as a static evaluation of the code's characteristics.

Since the values of effort assessments serve only for comparison purposes, there is no need for absolute predictions of actual cost in terms of wages, man-hours, fixed-costs, resources, etc. Thus, the evolution's (maintenance) change rate over time is considered as constant and neutral. Hence, instead of actual time, the change rates are expressed in terms of number of applied actions or maintenance scenarios, notated by the Greek letter  $\lambda$  as indicated in Table 4-2 and Figure 4-5.

Furthermore, since the metric equations in Table 4-3 assess the effort impact only for a single scenario application ( $\lambda=1$ ), the introduced approach formulates the system's size change rate through differential equations, assessing the total progressive effort impact for

any number ( $\lambda \in \mathbb{N}^*$ ) of scenario applications. The key concept is in design attributes, such as the number of elements ( $N$ ) and the number of operations ( $M$ ), the values of which are constantly affected during repeated scenario applications according to design patterns logic. Now the primary interest is concentrated on the formulation of the change rates of the system's size and scenario's size, as visualized by the horizontal (a) and vertical (b) change flows in Figure 4-5. This two-level differential analysis overall expresses the change rate of system's size, expressed in terms of design attributes' values per applied actions ( $\lambda$ ). By solving this differential system, equations are derived expressing the total or progressive size, in the same terms.

### 4.3.8 Differential Analysis and Model Derivation

In this subsection, the core of the introduced modeling method is presented in a step-by-step description by using the CIBI vs. CVP general decision problem as an indicative example. In general, the equations of the metrics in Table 4-3 are used as basic elements for the definition of simple first-order differential equations towards extraction of specific equations for precise computations of the total progressive maintenance effort. The analysis includes only expansion scenarios (additions) since they have a dominant role in the progressive evolution of the system and thus to size/effort assessment, as supported in chapter 3.

SMC metric in Table 4-3, as an indirect metric, involves the measurement of other design attributes such as  $N$  and  $M$ . Thus, it implies that the investigation of the maintenance as an evolutionary process requires separate levels of analysis. Hence, the system's expansion is described in two parts: a) one regarding the evolution of the design attributes (such as  $N$  and  $M$ ) expressing the change rate of actions' size, and b) next regarding the evolution of each scenario's size expressing the change rate of the system's size.

**A. Design attribute evolution:** For each maintenance scenario, the related design attributes are updated based on individual probabilities of each scenario type and design pattern structural behavior. Similarly, the change rate of each design attribute is related to the individual probability of each scenario and design pattern structural behavior.

*Design attribute N:* The change rate of the design attribute  $N$  (number of elements) for the new element scenario is related to  $p_{nE}$  probability and design pattern structural behavior, which implies  $N$  increment by one for each scenario application, as indicated in Figure 4-3 and Table 4-3. In addition, for the new operation scenario, the design attribute  $N$  remains unchanged. Thus, the change rate of design attribute  $N$  is equal to the sum factors of each expected scenario probability ( $p_{nE}$  and  $p_{nP}$ ) multiplied by its increment rate (+1 and 0 respectively), returning  $p_{nE} \cdot 1 + p_{nP} \cdot 0$ , or  $p_{nE}$ . Thus, the expected change rate of design attribute  $N$  is expressed by the differential equation (4-1):

$$\frac{dn}{d\lambda} = p_{nE} \cdot 1 + p_{nP} \cdot 0 \rightarrow \int \frac{dn}{d\lambda} d\lambda = \int p_{nE} d\lambda \rightarrow n(\lambda) = \lambda p_{nE} + C \quad (4-1)$$

Initially,  $\lambda=0$ , and  $n(0)$  is equal to the initial value of attribute  $N$ . Thus, the actual value of the design attribute  $N$  during the  $\lambda$ th scenario application is given by the equation (4-2).

$$n(0) = N \rightarrow C = N \rightarrow n(\lambda) = \lambda p_{nE} + N \quad (4-2)$$

*Design attribute M:* Respectively, the expected change rate of the design attribute  $M$  is expressed by the differential equation (4-3):

$$\frac{dm}{d\lambda} = p_{nE} \cdot 0 + p_{nP} \cdot 1 \rightarrow \int \frac{dm}{d\lambda} d\lambda = \int p_{nP} d\lambda \rightarrow m(\lambda) = \lambda p_{nP} + C \quad (4-3)$$

Initially,  $\lambda=0$ , and  $m(0)$  is equal to the initial value of attribute  $M$ . Thus the actual value of the design attribute  $M$  during the  $\lambda$ th scenario application is given by the equation (4-4).

$$m(0) = M \rightarrow C = M \rightarrow m(\lambda) = \lambda p_{nP} + M \quad (4-4)$$

**B. Scenario (action) size evolution:** For each maintenance scenario, the required maintenance effort is related to the scenario’s size and depends on design pattern structural behavior based on the current values of design attributes. Similarly, the change rate of the required maintenance effort is related to the change rate of the scenario’s size, which is related to design pattern structural behavior based on the current values of design attributes for each maintenance scenario and its probability.

From this point and on, the analysis is separately conducted for each design combination under comparison. At this level of analysis, the design attributes values are replaced by the equations (4-2) and (4-4) of the previous step.

*CVP design combination:* The change rate of the scenario’s size for new element scenario is approximated by  $2(m(\lambda)+2)$  or  $m(\lambda)+2$  method interventions at  $m(\lambda)+2$  different classes, based on metric equation  $c_m(CVP,N,M,1,0,1)=2(M+2)$  in Table 4-3. Thus, the expected change rate of the scenario’s size for the new element scenario is expressed by the differential equation (4-5):

$$\frac{dc_{nE}}{d\lambda} = 2(m(\lambda) + 2) \rightarrow \frac{dc_{nE}}{d\lambda} = 2((\lambda p_{nP} + M) + 2) \quad (4-5)$$

The change rate of the scenario’s size for new operation scenario is approximated by  $n(\lambda)+1$  or  $n(\lambda)$  method interventions at one class, based on metric equation  $c_m(CVP,N,M,0,1,1)=N+1$ . Thus, the expected change rate of the scenario’s size for the new operation scenario is expressed by the differential equation (4-6):

$$\frac{dc_{nO}}{d\lambda} = n(\lambda) + 1 \rightarrow \frac{dc_{nO}}{d\lambda} = \lambda p_{nE} + N + 1 \quad (4-6)$$

Thus, the expected change rate of both scenarios’ size is equal to the sum of each scenario’s size change rate multiplied by its probability factor, as expressed by the differential equation (4-7):

$$\begin{aligned} \frac{dc}{d\lambda} &= p_{nE} \frac{dc_{nE}}{d\lambda} + p_{nO} \frac{dc_{nO}}{d\lambda} \rightarrow \frac{dc}{d\lambda} = p_{nE} 2((\lambda p_{nP} + M) + 2) + p_{nO} (\lambda p_{nE} + N + 1) \rightarrow \\ &\int \frac{dc}{d\lambda} d\lambda = \int (3\lambda p_{nE} p_{nP} + 2p_{nE} M + 4p_{nE} + p_{nO} N + p_{nO}) d\lambda \rightarrow \\ c(\lambda)_{CVP} &= \frac{3}{2} \lambda^2 p_{nE} p_{nP} + \lambda p_{nO} N + 2\lambda p_{nE} M + 4\lambda p_{nE} + \lambda p_{nO} + C \end{aligned} \quad (4-7)$$

By definition  $c(0)=0$ , since before the maintenance process, the total required maintenance effort/size is equal to zero. Thus, the total progressive scenarios’ size or required effort, for  $\lambda$  repeated scenario applications based on their individual probabilities, is given by the equation (4-8):

$$c(0)_{CVP} = 0 \rightarrow C = 0 \rightarrow c(\lambda)_{CVP} = \frac{3}{2} \lambda^2 p_{nE} p_{nP} + \lambda p_{nO} N + 2\lambda p_{nE} M + 4\lambda p_{nE} + \lambda p_{nO} \quad (4-8)$$

*CIBI design combination:* The change rate of the scenario’s size for new element scenario is approximated by  $m(\lambda)+1$  or  $m(\lambda)$  method interventions at one class, based on metric equation  $c_m(CIBI,N,M,1,0,1)=M+1$  in Table 4-3. Thus, the expected change rate of the scenario’s size for the new element scenario is expressed by the differential equation (4-9):

$$\frac{dc_{nE}}{d\lambda} = m(\lambda) + 1 \rightarrow \frac{dc_{nE}}{d\lambda} = \lambda p_{nP} + M + 1 \quad (4-9)$$

The change rate of the scenario’s size for new operation scenario is approximated by  $2(n(\lambda)+1)$  or  $n(\lambda)+1$  method interventions at  $n(\lambda)+1$  different classes, based on metric

equation  $c_m(\text{CIBI}, N, M, 0, 1, 1) = 2(N+1)$ . Thus, the expected change rate of the scenario's size for the new operation scenario is expressed by the differential equation (4-10):

$$\frac{dc_{nP}}{d\lambda} = 2n(\lambda) \rightarrow \frac{dc_{nP}}{d\lambda} = 2((\lambda p_{nE} + N) + 1) \quad (4-10)$$

Thus, the expected change rate of both scenarios' size is equal to the sum of each scenario's size change rate multiplied by its probability factor, as expressed by the differential equation (4-11):

$$\begin{aligned} \frac{dc}{d\lambda} &= p_{nE} \frac{dc_{nE}}{d\lambda} + p_{nP} \frac{dc_{nP}}{d\lambda} \rightarrow \frac{dc}{d\lambda} = p_{nE}(\lambda p_{nP} + M + 1) + p_{nP} 2((\lambda p_{nE} + N) + 1) \rightarrow \\ \int \frac{dc}{d\lambda} d\lambda &= \int (3\lambda p_{nE} p_{nP} + p_{nE} M + p_{nE} + 2p_{nP} N + 2p_{nP}) d\lambda \rightarrow \\ c(\lambda)_{\text{CIBI}} &= \frac{3}{2} \lambda^2 p_{nE} p_{nP} + 2\lambda p_{nP} N + \lambda p_{nE} M + \lambda p_{nE} + 2\lambda p_{nP} + C \end{aligned} \quad (4-11)$$

By definition  $c(0)=0$ , since before the maintenance process, the total required maintenance effort/size is equal to zero. Thus, the total progressive scenarios' size or required effort, for  $\lambda$  repeated scenario application based on their individual probabilities, is given by the equation (4-12):

$$c(0)_{\text{CIBI}} = 0 \rightarrow C = 0 \rightarrow c(\lambda)_{\text{CIBI}} = \frac{3}{2} \lambda^2 p_{nE} p_{nP} + 2\lambda p_{nP} N + \lambda p_{nE} M + \lambda p_{nE} + 2\lambda p_{nP} \quad (4-12)$$

Notice that equations (4-7) and (4-11) express a synthetic change rate term or more precisely a probability-weighted term.

**C. Selection decision:** Maintainability assessment between different design pattern combinations can be achieved through the comparison of the estimated total progressive maintenance effort or scenarios' size of each combination for a certain number of scenario applications ( $\lambda$ ). The selection of the most beneficial design combination regarding its maintainability perspective is based on the minimum total progressive amount.

Thus, the selection of the most beneficial design combination is formally stated: Selection =  $\min\{c(\lambda)_d\}$ ,  $\forall d \in \{\text{CVP}, \text{CIBI}\}$ . In addition, the difference of the total required effort estimation for CVP and CIBI is given by the equation (4-13):

$$c(\lambda)_{\text{CVP}-\text{CIBI}} = c(\lambda)_{\text{CVP}} - c(\lambda)_{\text{CIBI}} \rightarrow c(\lambda)_{\text{CVP}-\text{CIBI}} = \lambda(p_{nE} M - p_{nP} N - p_{nP} + 3p_{nE}) \quad (4-13)$$

Because the opposite characteristics of CIBI and CVP are inversely aligned, the equation (4-13) is a first-degree polynomial, neutralizing the second-degree trend of individual equations (4-8) and (4-12). Furthermore, the sign of equation (4-13) depends mainly on  $N$ ,  $M$ , and  $p_{nE}$ ,  $p_{nP}$  values, meaning that the decision is not straightforward, and thus the generated comparison model is significant and useful in respect to all used independent variables.

Finally, the above differential analysis for the CIBI vs. CVP general problem is summarized in Table 4-4, including engaged metrics, change rates, differential expressions, and derived equations for each level of analysis.

Table 4-4: Differential Analysis and Comparison Model Derivation for CVP vs. CIBI General Problem

Level of analysis	Engaged SMC metric, $\lambda=1$	Change rate differential expression	Derived Equation	Eq. 4-
1. Design attribute evolution				
Design attribute N (elements)	new element $\rightarrow N++$	$\frac{dn}{d\lambda} = p_{nE} 1 + p_{nP} 0$	$n(\lambda) = \lambda p_{nE} + N$	1,2
Design attribute M (operations)	new operation $\rightarrow M++$	$\frac{dm}{d\lambda} = p_{nE} 0 + p_{nP} 1$	$m(\lambda) = \lambda p_{nP} + M$	3,4
2. Scenario size evolution				
a) CVP design combination				

Level of analysis	Engaged SMC metric, $\lambda=1$	Change rate differential expression	Derived Equation	Eq. 4-
New Element	$c_m(CVP,N,M,1,0,1)=2M$	$\frac{dc_{nE}}{d\lambda} = 2m(\lambda)$		5
New Operation	$c_m(CVP,N,M,0,1,1)=N+1$	$\frac{dc_{nP}}{d\lambda} = n(\lambda) + 1$		6
Total progressive effort CVP		$\frac{dc}{d\lambda} = p_{nE} \frac{dc_{nE}}{d\lambda} + p_{nP} \frac{dc_{nP}}{d\lambda}$	$c(CVP, N, M, p_{nE}, p_{nP}, \lambda) =$ $\frac{3}{2}\lambda^2 p_{nE} p_{nP} + \lambda p_{nP} N +$ $2\lambda p_{nE} M + 4\lambda p_{nE} + \lambda p_{nP}$	7,8
b) CIBI design combination				
New Element	$c_m(CIBI,N,M,1,0,1)=M+1$	$\frac{dc_{nE}}{d\lambda} = m(\lambda) + 1$		9
New Operation	$c_m(CIBI,N,M,0,1,1)=2N$	$\frac{dc_{nP}}{d\lambda} = 2n(\lambda)$		10
Total progressive effort CIBI		$\frac{dc}{d\lambda} = p_{nE} \frac{dc_{nE}}{d\lambda} + p_{nP} \frac{dc_{nP}}{d\lambda}$	$c(CIBI, N, M, p_{nE}, p_{nP}, \lambda) =$ $\frac{3}{2}\lambda^2 p_{nE} p_{nP} + 2\lambda p_{nP} N +$ $\lambda p_{nE} M + \lambda p_{nE} + 2\lambda p_{nP}$	11, 12

## 4.4 Generalizing and Formalizing Modeling Method

A general formal framework for the introduced modeling method has been developed in which the general notation of all aspects of the differential analysis is strictly defined as sets and arrays. In this case, the basic (SMC) metric equations are defined as first-degree polynomial expressions for all of design attributes' factors and their weights plus one constant factor. In the case of an alternate metric with a different set of independent variables, affected by different weights, or expressed through another type of equation, this framework provides a sufficient general template for any further adaptation. Although such a rigorous framework seems to be complicated, its implementation through software is rather a regular task. An indicative (dynamic) implementation of such a framework in MATLAB® is described in this subsection and provided online in (Karanikolas, Dimitroulakos, & Masselos, n.d.-b) for further research purposes.

### 4.4.1 Modeling Framework

In this sub-section, a more general formal and rigorous framework regarding the differential analysis of the introduced modeling method is provided. The general notation of all the aspects of the differential analysis is strictly defined through sets and matrixes in Table 4-5 in correspondence to CIBI vs. CVP general problem's notation.

Table 4-5: Terminology and Notation of Modeling Framework

Terminology	General Notation	Correspondence to CIBI vs CVP (general problem) Notation
Design (pattern) combination under comparison	$d_i \in D_{ items }$	$d_i \in D = \{CVP, CIBI\}$
Initial Design Attributes (of specific problem)	$l_i \in L_{ items }$	$l_i \in L = \{N, M\}$
Structural (method, class) aspect	$a_i \in A_{ items }$	$a_i \in A = \{a_m, a_c\}$
Maintenance (change) scenario type	$s_i \in S_{ items }$	$s_i \in S = \{n_e, n_p\}$
Individual scenario probability	$p_i \in P_{ items }$	$p_i \in P = \{p_{nE}, p_{nP}\}$
Change rate of Design Attributes per Scenario	$f_{ij} \in F_{ L  \times  S }$	$f_{ij} \in F_{ L  \times  S } =$ $\{\{1,0\}, \{0,1\}\}$
Factor of each Design Attribute in fundamental (single scenario/SMC) metric equation (first degree polynomial expression for each of $ L $ values, plus one constant factors)	$k_{i,j,m,n} \in K_{ D  \times  S  \times  A  \times ( L +1)}$	$k_{i,j,m,n} \in K_{ D  \times  S  \times  A  \times ( L +1)} =$ $\{\{ \{ \{0,1,2\}, \{0,1,2\} \},$ $\{ \{1,0,0\}, \{0,0,1\} \},$ $\{ \{0,1,0\}, \{0,0,1\} \},$ $\{ \{1,0,1\}, \{1,0,1\} \} \}$
Number of implemented maintenance changes (scenarios)	$\lambda$	$\lambda$
Value of specific design attribute during $\lambda$ th maintenance changes (scenarios)	$l_i(\lambda): l_i \in L_{ items }$	$l_i(\lambda): l_i \in \{N, M\}$
Single scenario ( $\lambda=1$ ) effort for specific <b>S, D, L</b> , and all <b>A</b>	$c_m(d_i, l_1, \dots, l_g, s_j, 1) =$ $\sum_{q=1}^{ A } \left( \sum_{g=1}^{ L } (k_{i,j,q,g} \cdot l_g) + k_{i,j,q, L +1} \right)$	$s_1 = n_e \rightarrow p_{nE} = 1 \ \& \ p_{nP} = 0$ $s_2 = n_p \rightarrow p_{nE} = 0 \ \& \ p_{nP} = 1$ $c_m(CVP, N, M, 1, 0, 1),$ $c_m(CVP, N, M, 0, 1, 1),$ $c_m(CIBI, N, M, 1, 0, 1),$ $c_m(CIBI, N, M, 0, 1, 1)$
Total (progressive) effort for specific $\lambda, P, D, L$ , and all <b>S, A</b>	$c_m(d_i, l_1, \dots, l_g, p_1, \dots, p_j, \lambda)$	$c_m(D, N, M, p_{nE}, p_{nP}, \lambda)$

More specifically, the set of design alternatives under comparison is represented by D set. The initial values of each key design attribute are represented by L set. Furthermore, the different structural aspects of the analysis are represented by the auxiliary A set. Each structural aspect reflects a different type of affected code segment, such as method ( $a_m$ ) and class ( $a_c$ ) interventions. Respectively, the different types of major maintenance scenarios are represented by the auxiliary (S) set. The values of the individual probabilities for each scenario type are represented by P set. The change impact of each scenario type on each design attribute is represented by F set. The F set is a matrix of which the dimensions are related to the product of lengths of L and S sets, or else  $|L| \times |S|$ . Thus, the number of F row represents the related number of design attributes of L set, and the number of F column represents the related number of maintenance scenario of S set. For instance, the first value of F set or  $F[1,1]=1$  reflects the increment by one of the 1<sup>st</sup> element or design attribute in L set or  $L[1]=\{N\}$  during the application of the 1<sup>st</sup> element or new element scenario of S set or  $S[1]=\{n_e\}$ . The values of weight-factors that affect each design attribute (L) for each specific structural aspect (A), scenario type (S), and design combination (D) are represented by K set. The K set is a matrix of which the dimensions are related to the product of lengths of D, S, A, and L sets, or else  $|D| \times |S| \times |A| \times |L| + 1$ . For instance, the first element of K set reflects the weight-factor that affects the CVP design combination, during a new element ( $n_e$ ) scenario application, about the structural aspect of method interventions ( $a_m$ ), for the design attribute of composition's elements (N). The values of these weight-factors (K) combined with the initial values of the design attributes (L) define the fundamental (SMC) metric equations ( $c_m$ ) that express the required effort of a specific scenario type (S) and design combination (D) for a single scenario application ( $\lambda=1$ ).

Notice that the fundamental (SMC) metric equations ( $c_m$ ) are defined as first-degree polynomial expressions for each of the design attribute factors (L) plus one constant factor. The constant factor is represented by the extra last value of K set. Thus, in the case where other independent variables, not necessarily design attributes, should be added to the model, they could be inserted in L set. In addition, if some of those attributes (L) are affected by different weight in (SMC) metric equations, this could be inserted into the weight-factors' (K) set. An interesting example of different weights on attributes of SMC metric is presented in subsection 3.9.2 as an extension example. However, even if SMC is expressed through another type of equation, this framework provides a sufficient general description and template for any further adaptation.

The two levels of the differential analysis and integration are formulated by presenting their change rate differential expressions and the derived outcome of each level, as presented in Table 4-6. In the first row, the evolution of design attributes' values (L) is analyzed by stating their change rates per scenario application. Thus, the change rate of each design attribute is expressed by multiplying individual scenario probabilities (P) and their related impact-weight (F) for all scenario's types (S). Consequently, the derived equations  $l_i(\lambda)$  express the expected value of each design attribute after any number ( $\lambda$ ) of scenario applications. In the second row in Table 4-6, the evolution of scenario size is analyzed by stating their change rates per scenario application. Thus, the change rate of scenarios' size is expressed by multiplying individual design attributes (L) and their related weight-factors (K) for all attributes (L), and all structural aspects (A), also probability-weighted through (P), and for all scenario's types (S). Consequently, the derived equations  $c_m(\dots, \lambda)$  express the total progressive or expected value of maintenance effort/size for each design alternative (D) after any number ( $\lambda$ ) of scenario applications. Finally, the selection/decision is derived from the minimum total progressive effort, as typically expressed by the equation (4-14).

Table 4-6: Modeling Framework of Differential Analysis and Comparison Model Generation

Level of analysis	Engaged SMC metric	Change rate differential expression	Derived Outcome
Design attribute evolution			
$\forall l_i \in L_{ items }$		$\frac{dl_i}{d\lambda} = \sum_{j=1}^{ S } p_j \cdot f_{i,j}$	$l_i(\lambda)$
Scenario size evolution			
Total progressive maintenance size/effort $\forall d_i \in D_{ items }$	$c_m(d_i, l_1, \dots, l_g, s_j, 1)$	$\begin{aligned} \frac{dc_i}{d\lambda} &= \sum_{j=1}^{ S } (p_j \cdot c_m(d_i, L, s_j, 1)) = \\ &= \sum_{j=1}^{ S } \left( p_j \sum_{q=1}^{ A } \left( \sum_{g=1}^{ L } (k_{i,j,q,g} \cdot l_g(\lambda)) + k_{i,j,q, L +1} \right) \right) \end{aligned}$	$c_m(d_i, l_1, \dots, l_g, p_1, \dots, p_j, \lambda)$

Although the presented framework of analysis seems to be complicated, its implementation through software (e.g., MATLAB®, MS Mathematics) is rather a regular task. In general, solving a differential equation is not always an easy matter. Still, for the simple first-order form  $y'=f(\lambda)$  used in the proposed modeling method framework, an explicit solution can be easily derived through integration in both sides of each differential equation, as supported by calculus theory in (Stewart, 2015).

$$\min_{\forall d_i \in D} \left\{ \int_{t_1=0}^{\lambda} \left( \sum_{j=1}^{|S|} \left( p_j \sum_{q=1}^{|A|} \left( \sum_{g=1}^{|L|} \left( k_{i,j,q,g} \int_{t_2=0}^{\lambda} \left( \sum_{v=1}^{|S|} p_g \cdot f_{g,v} \right) dt_2 \right) + k_{i,j,q,|L|+1} \right) \right) \right) dt_1 \right\} \quad (4-14)$$

#### 4.4.2 Framework Implementation on General Problems using MATLAB®

In this subsection, an indicative (dynamic) implementation of the framework using MATLAB® is presented in Code A.1 of Appendix A. The code is adapted to CIBI vs. CVP general problem to be meaningful and help the researcher towards further adaptations. In Listing 4-1, the parameters of the CIBI vs. CVP general problem are loaded to the sets and matrixes of the proposed modeling framework. Each level of the differential analysis presented in Table 4-6 has been implemented into separate code segments. By running the script, several intermediate differential expressions are formed and solved. Furthermore, all the formal model's equations are dynamically generated and stored in several expressions and function types supported by MATLAB® environment. These dynamic functions can be further used for massively computations for any combination of the independent variables, producing useful data sheets and graphs. All the presented code has been tested on MATLAB® R2016a version.

Listing 4-1: Loading CIBI vs CVP general problem to MatLab Modeling Framework

```

1. % Data describing (general) comparison problem (CIBI vs CVP) and fundamental (SMC) metric analysis
2. % In this section, different or alternate problems should be described
3. D = {'CVP','CIBI'}; % tags of Design combinations under comparison
4. L_tags = {'N','M'}; % tags of design attributes: N initial elements, M initial operations
5. A = {'Method aspect', 'Class aspect'}; % tags of Structural aspects
6. S = {'nE', 'nP'}; % tags of Types of maintenance scenarios: nE new composition element, nP new operation
7. F = [1 0; 0 1]; % N:+1 and M:+0 for nE, N:+0 and M:+1 for nP (change rates of affected design attributes for each scenario type |S|x|L|)
8. % SMC metric factors on design attributes L, for each design combination D, scenario S, and aspect A are stated in K array
9. K = zeros([size(D,2) size(S,2) size(A,2) size(L_tags,2)+1]); % creates empty matrix with dimensions: |D|x|S|x|A|x|L|+1
10. % method class (structural aspects)
11. % D S N M - N M -
12. K(1,1,:,:) = [0 1 2; 0 1 2]; % on CVP for a nE : totally 0N+1M+0 method + 0N+1M+0 class interventions = 2(M+2)
13. K(1,2,:,:) = [1 0 0; 0 0 1]; % on CVP for a nP : totally 1N+0M+0 method + 0N+0M+1 class interventions = N+1
14. % -----
15. K(2,1,:,:) = [0 1 0; 0 0 1]; % on CIBI for a nE : totally 0N+1M+0 method + 0N+0M+1 class interventions = M+1
16. K(2,2,:,:) = [1 0 1; 1 0 1]; % on CIBI for a nP : totally 1N+0M+0 method + 1N+0M+0 class interventions = 2(N+1)
    
```

### 4.4.3 Graph Generation and Decision-Making for Specific Practical Systems using MATLAB®

In this subsection, the generated expressions and dynamic functions produced by the previous code are used for decision-making support among design alternatives in Code A.2 of Appendix A. More specifically, any possible combination of initial design attribute values and their individual probabilities, describing a specific and practical system implementation (e.g., Compiler, Interpreter/DSL, GUI), can be provided through the parameter sets L and P. Next, the example of plotting code can draw meaningful graphs to support decision-making. Notice that the plotting code has been parameterized based on set matrices, and thus it is common and reusable for any set of parameters.

In Listing 4-2, the design attributes ( $N=40$ ,  $M=10$ ) and scenario's probabilities ( $p_{mE}=p_{nP}=0.5$ ) of the Interpreter implementation which represents an instance of the CIBI vs. CVP general problem, are loaded to the sets and matrixes of the proposed modeling framework. Then, the symbolic functions named 'FM\_cost\_D\_f', dynamically generated by the modeling framework, are used for the estimation of required effort per design alternative and for different values of scenario's applications ( $\lambda$ ). For example, the function call 'FM\_cost\_D\_f{1}(40, 10, 0.5, 0.5, 80)' returns the total required effort for the 1<sup>st</sup> design alternative (i.e., CVP), with specific design attributes (i.e.,  $N=40$ ,  $M=10$ ) and scenario's probabilities (i.e.,  $p_{mE}=p_{nP}=0.5$ ), after the application of a certain ( $\lambda=80$ ) number of scenarios. Respectively, the decision-making code for the tow similar frameworks adapted to the extended general designing problems are presented in Code B.1,2 and C.1,2 of Appendix A.

Listing 4-2: Loading Interpreter characteristics to MatLab Modeling Framework

```

17. % Data (design attributes) derived from specifications of a specific system (instance of general problem)
18. % In this section, the design attributes of a specific system are placed (multiple attributes can be declared as arrays)
19. L = [40 10]; % initial values of design attributes N and M
20. P = [0.5 0.5]; % individual probabilities of each scenario type |S|
21. %===PLOTTING CODE =====
22. It = 5:5:100; % declares the array of the interval of interest
23. merged_parameters_values = {}; % merges L(i) and P(i) values of parameters in a single array of cells
24. for i=[1:size(L_tags,2)]
25.     merged_parameters_values{i} = L(i);
26. end
27. for i=[1:size(P,2)]
28.     merged_parameters_values{size(L_tags,2)+i} = P(i);
29. end
30. % merged_parameters_values are aligned according to FM_cost_D_f symbolic function's declaration
31. % Uses the FM_cost_D_f symbolic function (returning an array of computations for all D) and computes total effort for  $\lambda = [5:5:100]$ 
32. for i=[1:20]
33.     G_lines(i,:) = FM_cost_D_f(merged_parameters_values{:}, i*5);
34. end
35. % adds different lines for each design combination D
36. plot(It, G_lines, 'MarkerSize', 4, 'Marker', 'square');
    
```

### 4.4.4 Formal Model Application in Examples of Practical Specific Problems

After the formal model has been derived, it can be easily used to repeatedly support decision-making for any attribute set of a specific system. Alternatively, each formal model is a parametric solution of the general problem under study. Furthermore, the (indicative) dynamic implementation of the framework in MATLAB®, provided online in (Karanikolas et al., n.d.-b), can be used for generating the formal models and creating meaningful diagrams of the required effort for different  $\lambda$  values. In addition, the derived formal model, as general and reusable, can be integrated into the company's quality policy for future use in similar projects as illustrated in Figure 1-3. As an application example, the derived formal model is applied to the practical examples of an Interpreter implementation (presented in Table 1-1 and Table 4-1), and a Graphic User Interface (GUI) implementation (presented in Table 1-1). The initial values of the design attributes ( $N$ ,  $M$ ,  $p_{nE}$ ,  $p_{nP}$ ) for each practical example are referred in the title of each graph in Figure 4-6.



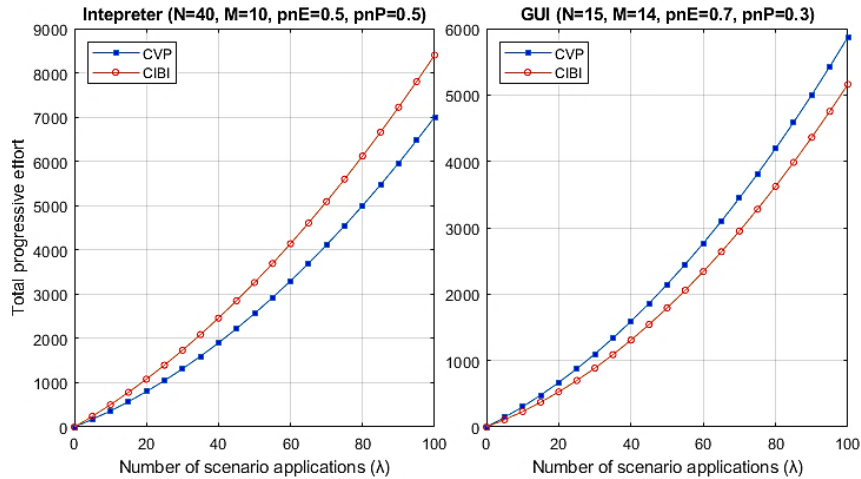


Figure 4-6: Results of the application of the Formal comparison Model on the practical examples of Interpreter, and Graphic User Inter-face (GUI), specific problems as instances of CVP vs. CIBI general problem.

The generated diagrams in Figure 4-6 indicate that CVP design combination is preferred in the case of Interpreter implementation, since it requires less effort during maintenance for any number of scenario applications or  $\lambda$  value. Respectively, CIBI design combination is preferred in the case of GUI implementation, thus proper design combination depends on the specific characteristics or design attributes of a particular instance of the general problem. Consequently, the selection’s outcome is not about that one design alternative is generally superior of some others, but instead, the most beneficial design alternative varies and depends on the model’s parameters (design attributes and scenarios probabilities). Thus, it is very difficult to make such complex decisions based on real-world experience. Furthermore, the long-term gain, between CVP and CIBI required effort, exceeds 20% of optimum (minimum) effort for all implementations, highlighting the beneficial contribution of the formal model to the decision-making process.

Given that the derived formal models are easily reusable in a general family of common problems, preventing significant loses in terms of maintenance effort; the future benefits of the proposed technique outweigh its analysis cost. A more detailed and formal justification about the maximum allowed derivation cost of Formal comparison Models is presented following subsection.

#### 4.4.5 Justification of Formal Model’s Derivation Cost

One critical issue that should be addressed is whether the long-run benefits from the repeated implementation of a formal model to specific instances of a general designing problem justifies or exceeds its initial derivation cost. From a company’s perspective, the formal model’s derivation cost is an investment. The long-run benefits from the implementation of that formal model are the investment’s return. Typically, the evaluation of such investments can be assessed through standard financial methods and measures such as Net Present Value (NPV) and Internal Rate of Return (IRR). However, a simpler formula for finding the relation between initial model’s derivation cost and long-term benefits (return) is suggested in (Bass et al., 2012) and presented in Table 4-7.

Table 4-7: Formula for Finding the Relation Between Initial Model’s Derivation Cost and Long-Term Benefits (Return)

Description of Factor	Notation / Equation
The number of possible design <b>decisions</b> among competing design alternatives for different and specific systems for a general designing problem	Ndes

Description of Factor	Notation / Equation
<ul style="list-style-type: none"> <li>The more general and significant the problem the higher the possible number of design decisions for different instances (systems) of the general designing problem</li> </ul>	
The average <b>lifetime cost</b> of those different and specific systems <ul style="list-style-type: none"> <li>at least 60% of software life-time cost is about <b>maintenance cost</b> until their retirement as suggested by real case evidence</li> </ul>	$Lcost$ $Mcost = 0.6 * Lcost$
The (one-time) derivation cost of the formal models (through the proposed modeling method) for the particular (general) designing problem	$Dcost$
The cost of making a design decision for a particular system by using the derived formal models ( <b>implementation cost</b> ) <ul style="list-style-type: none"> <li>usually is very small since the derived equations of the formal model return the estimated long-term effort per design alternative only through a single computation, thus tends to zero</li> </ul>	$Icost \rightarrow 0$
The <b>possibility</b> of making the <b>right</b> (most maintainable) selection/ <b>decision</b> among design alternatives based on developers' experience or intuition	$Prdes$
The average <b>portion of wasted effort</b> against the optimal maintenance cost (Mcost) in case of a wrong decision-making <ul style="list-style-type: none"> <li>approximated to at least 30% of systems' maintenance cost as suggested by the evidence of the study</li> </ul>	$Pweffort = 0.3$
The <b>average negative impact</b> of making a design decision for a particular system by not using the derived formal models, usually through experience or intuition <ul style="list-style-type: none"> <li>is related to the average amount of wasted effort in case of a wrong decision-making (<b>Experience based cost</b>)</li> <li>depends on the possibility of making a wrong decision (1-Prdes)</li> </ul>	$Ecost =$ $Pweffort * Mcost * (1-Prdes) =$ $Pweffort * 0.6 * Lcost * (1-Prdes) =$ $0.3 * 0.6 * Lcost * (1-Prdes) =$ $0.18 * Lcost * (1-Prdes)$
The <b>relation</b> that justifies a formal method to support decision-making among alternatives for a particular general designing problem <ul style="list-style-type: none"> <li>the average negative impact per decision-making multiplied by the number of possible design decisions should be exceeds the initial (one-time) derivation cost of the formal models plus the implementation cost multiplied by the number of possible design decisions</li> </ul>	$Ndes * Ecost > Dcost + Ndes * Icost \rightarrow$ $Ndes * Ecost > Dcost + Ndes * 0 \rightarrow$ $Ndes * Ecost > Dcost \rightarrow$ $Dcost < Ndes * Ecost \rightarrow$ <b><math>Dcost &lt; Ndes * 0.18 * Lcost * (1-Prdes)</math></b>
Example of the relation (pessimistic assumption) <ul style="list-style-type: none"> <li>experienced developers with portability <math>Prdes = 66\%</math> to take the right decision based on their experience or intuition</li> <li>small software life-cycle cost of <math>Lcost = 20000\\$</math></li> <li>small number of formal model's implementations to specific systems, <math>Ndes=5</math></li> </ul>	$Dcost < Ndes * 0.18 * Lcost * (1-Prdes) \rightarrow$ $Dcost < 5 * 0.18 * 20000\$ * (1-0.66) \rightarrow$ <b><math>Dcost &lt; 6.000\\$</math></b>
Example of the relation (realistic-regular assumption) <ul style="list-style-type: none"> <li>experienced developers with portability <math>Prdes = 66\%</math> to take the right decision based on their experience or intuition</li> <li>software life-cycle cost of <math>Lcost = 100000\\$</math></li> <li>number of formal model's implementations to specific systems, <math>Ndes=20</math></li> </ul>	$Dcost < Ndes * 0.18 * Lcost * (1-Prdes) \rightarrow$ $Dcost < 20 * 0.18 * 100000\$ * (1-0.66) \rightarrow$ <b><math>Dcost &lt; 120.000\\$</math></b>

According to the assumptions and definitions in Table 4-7, the relation that justifies the initial derivation cost of formal models is given by the Equation (4-15). In particular, the (one-time) derivation cost (Dcost) should be less than the product of the number of possible design decisions (Ndes), the constant factor (0.18), the average lifetime cost of the systems on which the decisions are made, and the probability of making a wrong decision based on developers' experience (1-Prdes). Thus, the higher the possibility of using the model (Ndes) or the higher the average lifetime cost of the systems in which the model applied (Lcost) or the lower the experience level of developers (Prdes), the higher the allowed derivation cost of the formal model (Dcost). It is important that the difference between the maximum allowed derivation cost obtained by the formula and the actual derivation cost represents the net present value (NPV) or the extra profit as a result of the use of the formal model.

$$Dcost < Ndes * 0.18 * Lcost * (1 - Prdes) \quad (4-15)$$

Referring to the first example in Table 4-7, even for the most pessimistic assumptions of formula parameters, there is enough justification for analyzing and generating formal models for general and significant design problems through the proposed modeling method. For the more usual and realistic parameters in Table 4-7, there is no doubt about the significant value and competitive advantage provided by the introduced modeling method through the derived formal models.

## 4.5 Validation Evidence

The generated formal model for the comparison of CIBI vs. CVP general problem is validated by many other theoretical and empirical studies in the literature. This proves the validity of the proposed modeling method. More specifically, the argumentation of SMC metrics in Table 4-3 provides sufficient theoretical background about SMC's actual relation to properties like code size, scattering degree, cohesion, coupling, etc., and thus to the required effort or maintainability degree, as discussed in (Aloysius & Arockiam, 2013; Aversano et al., 2009; Canfora et al., 2010; Hassan, 2009; Heitlager et al., 2007b; Karanikolas et al., 2017; Riaz et al., 2009a). Furthermore, the SMC metric is in accordance with other similar metrics such as the Evolution Complexity (Tom Mens & Eden, 2005) and the Computational Complexity (Hills et al., 2011), as discussed in chapter 3. The equations (4-8), (4-12), (4-13), and the application results in Figure 4-6 are also confirmed by the quantitative analysis in chapter 3 in which computations for CIBI vs. CVP comparison have been performed through a custom function. This function is available online for demonstration purposes and further tests (Karanikolas, Dimitroulakos, & Masselos, n.d.-a). Moreover, the permanently second-degree increased rate of equations (4-8) and (4-12) is in accordance with Barry et al. (Barry et al., 2007) empirical validation evidence, and Bakota et al. general prediction model (Bakota et al., 2012).

Furthermore, the SMC metrics focus on expansion scenarios and magnify the impact of the relevant interventions concerning critical characteristics of modifiability such as coupling and cohesion among code segments of logical entities. This reasoning is prior evidence of the method's validity towards its main objective i.e., to compare design alternatives with regards to their modifiability perspective.

Finally, taking into consideration the assumptions and characteristics of SMC metric with regards to the common and neutral to decision-making factors, mentioned in subsection 4.3.3, the following conclusions can be drawn. As the maintenance process evolves, and despite the various stochastic and random factors affecting it, the average long-term effect of these factors would be eventually negligible, and thus the predictions of the required effort are increasingly driven by the standard and recurring structural behavior of the used design patterns. Hence, the proposed approach aims at eliminating transitory and biased factors to enhance mid-to-long-term predictive ability and selection accuracy.

## 4.6 Conclusions

### 4.6.1 General Requirements and Limitations

The proposed modeling method is mainly useful for comparing design alternatives that solve the same general problem using different design approaches. Thus, absolute maintenance cost assessments or effort estimations for individual design implementations are out of the scope of the proposed approach. In any case, the proposed method is suitable only for modeling design alternatives of important families (classes) of problems which also have a dominant impact on the overall maintainability of the system.

Furthermore, the proposed formal method is (by definition) focused on maximizing the potential for being general over different instances of a given general problem. However, formal methods usually suffer from lack of realism of context and precision of measurements, as stressed in (Stol & Fitzgerald, 2018). Ideally, actual observations from field experiments or case studies that maximize the potential for realism of context would be preferable for validation purposes. Nevertheless, in real life, finding identical actual systems with common design attributes, developed in different design variations is almost impossible. Additionally, the number of recorded observations is very limited per case study, using heterogeneous metrics, and unevenly conducted through literature. Thus, they are not statistically meaningful, heavily limiting the generalization of inferences, as pointed in (Langdon et al., 2016; Shepperd & MacDonell, 2012). Moreover, developer-related

aspects, such as experience level and learning rate are also ignored by the method since they are heavily biased (human-related) factors hard to be assessed and measured. Because of all these reasons, there is no easy way to determine the accuracy of the method referred to possible wrong decisions. This is a standard concern with regards to validity since the attempt to validate the method based on a limited number and dissimilar case studies may increase realism of context while sacrificing generalizability which should be the method's primary focus. A possible solution to this issue may be the simulation of system's structural evolution for different design alternatives to produce adequate number of homogenous observations and measurements towards a statistical validation. These issues and potentials are explored in Chapter 6 through an intensive experimentation process.

#### 4.6.2 Extensions and Further Research

The proposed modeling method can be used to support different tools, including aspect-oriented programming tools that generate source code (Völter, 2003), templates or libraries from a higher-level language to evaluate appropriate design pattern combinations. Furthermore, the generated formal models can be used as fitness functions (effort estimators) for optimization problems or design space exploration (seeking for optimal design solution among many alternatives) solved through heuristic algorithms such as genetic algorithms (Clarke et al., 2003). Under this perspective, the proposed modeling method and the generated formal models can be used in by refactoring tools and techniques for UML diagrams and design patterns such as those discussed in (Jahnke & Zündorf, 1997; T. Mens & Tourwe, 2001), as proposed in (T. Mens & Tourwe, 2004).

The provided modeling framework can be used for analyzing alternate general and significant design problems in software engineering. For example, many significant variations or extensions of the general problem of part-whole representations can be considered by attaching other well-known design patterns such as Decorator, Observer, Mediator, Abstract Factory, and Prototype. All these potentials are discussed in Chapter 5.

Furthermore, a promising perspective of the proposed modeling method is towards supporting decision-making among design alternatives even under full or partial uncertainty (e.g., for the whole range or for an interval of scenarios' probabilities). In this case, decision criteria such as minimization of wasted effort in the worst case or maximization of gained effort in the optimal case can be supported by the derived equations through further integration on probabilities' factors (e.g.,  $p_{nE}$ ,  $p_{nP}$ ). Similarly, decision-making can be amplified through the horizon analysis technique. That is, forecasting the realized effort over various maintaining periods or horizons where each period has different scenarios' probabilities. Under this perspective, even the development process can be considered as a preliminary maintenance period. Horizon analysis opens a whole spectrum of different aspects and criteria such as investing, economic, and financing, regarding the evaluation and selection among design alternatives for various sub-periods of software lifecycle. In these ways the proposed framework is extended to even more realistic settings as demonstrated in Chapter 7.

#### 4.6.3 Overall Assessment

Software architecture design includes several decisions with significant impact on the pursued quality attributes. Decisions made during software architecture design also heavily affect maintainability and modifiability of software and the relevant time and effort. Due to software complexity, decisions made by experienced developers lead to suboptimal results. The proposed modeling method generates probabilistic comparison models that estimate maintainability of object-oriented design alternatives through effort predictions in a formal and deterministic way. This approach limits the ambiguity imposed by the stochastic nature of the maintenance process.

The theoretical foundation of the proposed modeling method and the results of relevant works provide strong evidence that the derived formal models provide reliable

estimations of the expected effort. Thus, decisions concerning design alternatives exhibit very limited selection-risk, avoiding significant amounts of wasted maintenance effort. Methods that yield such formal, general, and reusable models can help engineers improve the quality of their decision-making and develop more maintainable software.

## 5 Extended General Design Problems

### 5.1 Chapter Overview

In this chapter, the proposed modeling method, analyzed in chapter 3 and chapter 4, is applied to three different extensions of the CVP vs CIBI general problem, assessing its applicability to even more realistic settings. In particular, the established design patterns of Decorator, Mediator, Observer, Abstract Factory, and Prototype have been engaged and modeled. In addition, the generated formal models have been tested on several specific instances of each general problem. Moreover, the exploration of almost the entire design space of each general problem through samples of one thousand sets of their parameters has been attempted. The results prove that the proposed modeling theory and derived formal models can efficiently support decision-making among design alternatives, leading to considerable reduction of the maintenance effort that lies between 25% and 90% of the optimal required effort.

More specifically, the proposed modeling method is applied to three important general problems in the field of object-oriented design. Each of these problems is an extension of the basic CVP vs CIBI problem, altering its applicability to even more complex and realistic settings. The new general problems incorporate the well-known object-oriented design patterns of Decorator, Mediator, Observer, Abstract Factory, and Prototype, introduced by Gamma et al. (Gamma et al., 1994). These patterns are frequently combined with the Composite and Visitor design patterns to enhance their functionality (e.g., by providing on demand additional functionality on elements, synchronizing communication, and coordination among elements through common interface, instantiating additional families of elements, etc.). Thus, such design patterns are significant and widely used in the field of software engineering. Since many of these design patterns are affected by major maintenance scenarios in complex and conflicting ways, the proposed modeling method derives appropriate formal models that resolve conflicting issues and tradeoffs to support early decision-making among different design alternatives.

The context of this chapter is based on the motivation examples in chapter 1, the significant design problem of part-whole representations in chapter 3, and the modelling method and framework presented in chapter 4. The rest of this chapter is organized as follows. Subsection 5.2 attaches and evaluates the Decorator design pattern. Subsection 5.3 attaches and evaluates the Mediator and Observer design patterns. Subsection 5.4 attaches and evaluates the Abstract Factory and Prototype design patterns. Subsection 5.5 evaluates the initial design problem of part-whole representations. Subsection 5.6 summarizes the contribution evidence of the modeling method. Finally, in subsection 5.7, the validity challenges, limitations, future research issues, and conclusions are presented.

### 5.2 Attaching Decorator Design Pattern

#### 5.2.1 Problem Description

In this subsection, an extension of the CIBI vs. CVP problem is discussed by attaching the Decorator design pattern. A conceptual UML diagram of the new problem and the response measures or SMC metrics per maintenance scenario, are presented in Figure 5-1. More specifically, the Composite (CP) design pattern is often combined with the Decorator (DP) design pattern (Gamma et al., 1994). DP design pattern allows extending the functionality of CP concrete elements, during run-time, against the costly alternative of sub-classing existing CP elements. This situation is very common, especially for GUI implementations where several extra (on demand) functionalities or responsibilities can be attached in a graphical component during run-time. Each Decorator instance can be dynamically linked to a concrete CP element directly or through recursive calls in a chain

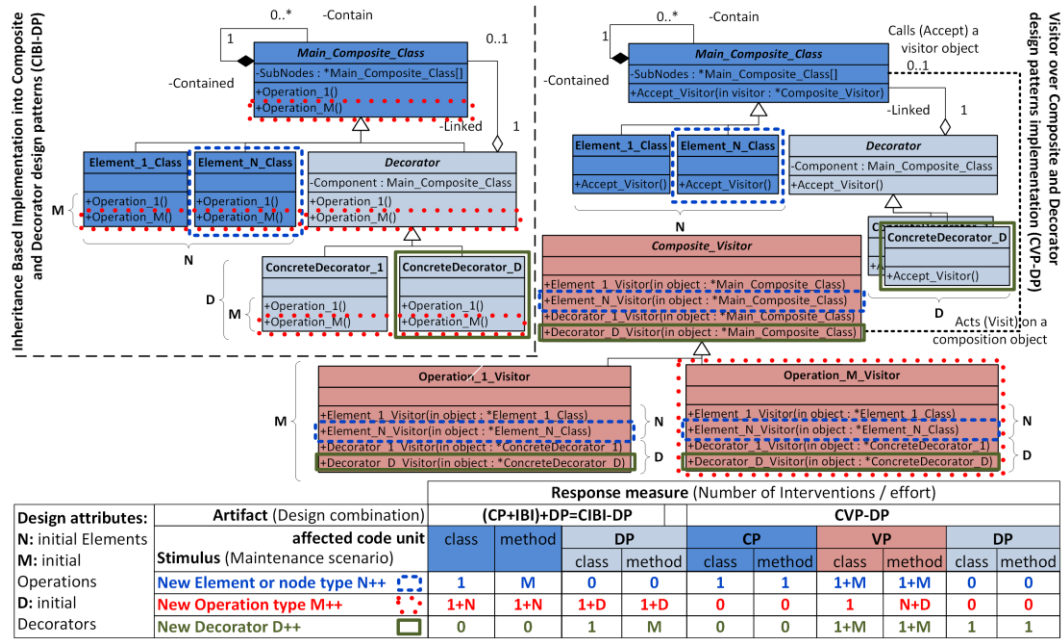


Figure 5-1: Conceptual UML class-diagram of CIBI-DP vs CVP-DP design combinations.

of several nested Decorators. Thus, DP is placed as an abstract sub-class into CP, which is further analyzed to several concrete decorators' sub-classes, implementing the required responsibilities. Hence, from the Decorator pattern perspective, a composite is a concrete component. Respectively, from the Composite's pattern perspective each decorator is part of its hierarchy, similar to a concrete CP element with a common interface. Furthermore, each Decorator usually contains the same operations, which can be also implemented through the Visitor design pattern (as CVP-DP combination versus the straight inheritance-based implementation (CIBI-DP)). Thus, in this case, two alternative design combinations arise (CVP-DP, CIBI-DP).

The selection between these alternatives can be supported by the introduced modeling method in chapters 3 and 4. The key design attributes are enhanced as N, M, and D for the initial number of decorators. Furthermore, major maintenance scenarios and their probabilities are enhanced as  $p_{nE}$ ,  $p_{nD}$ , and  $p_{nD}$  for adding a new decorator. Using the response measures or SMC metrics in Figure 5-1, the new problem can be easily formulated through the provided modeling method framework in subsection 4.4.

### 5.2.2 Derivation of Effort Measurements and Formal Models

A (dynamic) formal model generation for the new problem through the modeling method framework using MATLAB® code is available online in (Karanikolas et al., n.d.-b) and discussed in subsection 4.4. In addition, the modeling framework adapted to the extended with the Decorator design pattern CIBI-DP vs. CVP-DP general designing problem is presented in Code B.1 of Appendix A. In Listing 5-1, the parameters of the CIBI-DP vs. CVP-DP general problem are loaded to the sets and matrixes of the proposed modeling framework.

Listing 5-1: Loading CIBI-DP vs. CVP-DP general problem to MatLab Modeling Framework

```

37. % Data describing (general, extended) comparison problem CIBI_DP vs CVP_DP and fundamental (SMC) metric analysis
38. % In this section, different or alternate problems should be described
39. D = { 'CVP-DP', 'CIBI-DP' }; % tags of Design comb. under comparison
40. L_tags = {'N', 'M', 'D'}; % tags of design attributes: N initial elements, M initial operations, and V initial decorators
41. A = {'Method aspect', 'Class aspect'}; % tags of Structural aspects
42. S = {'nE', 'nP', 'nD'}; % Types of maintenance scenarios: nE new composition element, nP new operation, nDE new decorator element, nDP new decorator operation
43. F = [1 0 0; 0 1 0; 0 0 1]; % N:+1 M:+0 D:+0 for nE, N:+0 and M:+1 D:+0 for nP, N:+0 M:+0 D:+1 for nD (change rates of affected design attributes for each scenario type |S|x|L|)
44. % SMC metric factors on design attributes L, for each design combination D, scenario S, and aspect A are stated in K array
45. K = zeros([size(D,2) size(S,2) size(A,2) size(L_tags,2)+1]); %creates empty matrix dimensions: |D|x|S|x|A|x|L|+1
46. % method class (structural aspects)
47. % D S N M D - N M D -
48. K(1,1,:) = [0 1 0 2; 0 1 0 2]; %CVP-DP for nE: 0N+1M+0D+2 method + 0N+1M+0D+2 class = 2(M+2)
49. K(1,2,:) = [1 0 1 0; 0 0 0 1]; %CVP-DP for nP: 1N+0M+1D+0 method + 0N+0M+1D+1 class = N+D+1
50. K(1,3,:) = [0 1 0 2; 0 1 0 2]; %CVP-DP for nD: 0N+1M+0D+2 method + 0N+1M+0D+2 class = 2(M+2)
51. % -----
52. K(2,1,:) = [0 1 0 0; 0 0 0 1]; %CIBI-DP for nE: 0N+1M+0D+0 method + 0N+0M+0D+1 class = M+1
53. K(2,2,:) = [1 0 1 2; 1 0 1 2]; %CIBI-DP for nP: 1N+0M+1D+2 method + 1N+0M+1D+2 class = 2(N+1)+2(D+1)
54. K(2,3,:) = [0 1 0 0; 0 0 0 1]; %CIBI-DP for nD: 0N+1M+0D+0 method + 0N+0M+0V+1 class = M+1

```

The macro generates one formal model equation for each alternative design combination and produces its results as a single graph for specific design attributes and scenarios' probabilities. More specifically, the generated formal models are returned in the form of the symbolic expression  $cost\_DSAL(D)$  where  $D=\{CVP-DP, CIBI-DP\}$ . Thus, the total required effort for CVP-DP design alternative is given by the equation (5-1).

$$\begin{aligned}
c(\lambda)_{CVP-DP} = & \\
& \frac{3}{2} \cdot p_{nP} \cdot (p_{nD} + p_{nE}) \cdot \lambda^2 + \\
& (p_{nP} + 2 \cdot p_{nD} \cdot (M + 2) + 2 \cdot p_{nE} \cdot (M + 2) + p_{nP} \cdot (D + N))\lambda
\end{aligned} \tag{5-1}$$

Respectively, the total required effort for CIBI-DP design alternative is given by the equation (5-2).

$$\begin{aligned}
c(\lambda)_{CIBI-DP} = & \\
& \frac{3}{2} \cdot p_{nP} \cdot (p_{nD} + p_{nE}) \cdot \lambda^2 + \\
& (p_{nD} + p_{nE} + 4 \cdot p_{nP} + 2 \cdot D \cdot p_{nP} + M \cdot p_{nD} + M \cdot p_{nE} + 2 \cdot N \cdot p_{nP})\lambda
\end{aligned} \tag{5-2}$$

Using these equations, the selection of the most maintainable design combination for any number of future scenario's interventions ( $\lambda$ ) can be supported.

### 5.2.3 Formal Model Application in Examples of Practical Specific Problems

As an application example, consider a GUI system with the following initial design attributes and scenario probabilities  $\{N=15, M=14, D=14\}$  and  $\{p_{nE}=0.1, p_{nP}=0.8, p_{nD}=0.1\}$ . For this specific system, the CVP-DP design combination is preferable since it is the most maintainable, requiring the lowest effort, as indicated by the outcome in Figure 5-2. Respectively, in the case of different probabilities e.g.  $\{p_{nE}=0.4, p_{nP}=0.2, p_{nD}=0.4\}$ , the CIBI-DP design combination is preferable. Once more, it is important that the long-run difference, between the optimum (less effort) and the worst (higher effort) design options, exceeds 20% of optimum effort, which is a considerable amount of effort.



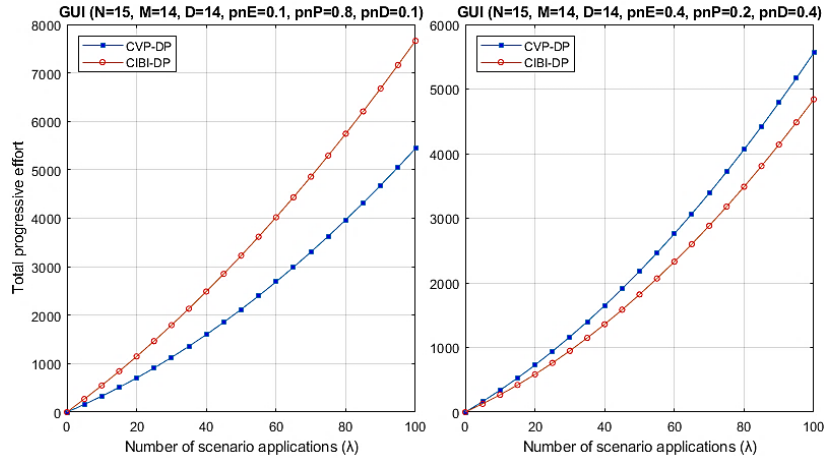


Figure 5-2: Results of the application of the Formal comparison Model on the practical examples of Graphic User Interface (GUI) specific problems as instances of CVP-DP vs. CIBI-DP general problem.

### 5.2.4 Average Rate of Gained or Avoided Wasted Effort

In this subsection, the exploration of almost the entire design space of CIBI-DP vs. CVP-DP general problem is attempted. A sample of one thousand sets of parameters has been randomly selected through a random generator of uniform distributions. The range for each problem’s variable was defined as  $N=[20, \dots, 250]$ ,  $M=[5, \dots, 150]$ ,  $D=[2, \dots, 100]$ ,  $p_{nE}=[0.1, \dots, 0.9]$ ,  $p_{nP}=[0.09, \dots, \text{rest to } 1]$ ,  $p_{nD}=[0.01, \dots, \text{rest to } 1]$ . The derived formal models have been used for the estimation of the required maintenance effort per design alternative (i.e., CIBI-DP, CVP-DP) as well as their difference. Based on these values, the rate of the gained or the avoided wasted effort has been computed for each of the sample’s instances. This rate is equal to the maximum minus the minimum ( $\max(\text{CIBI-DP}, \text{CVP-DP}) - \min(\text{CIBI-DP}, \text{CVP-DP})$ ) divided by the minimum ( $\min(\text{CIBI-DP}, \text{CVP-DP})$ ) of the required effort among all design alternatives. The average of these values for all sample’s instances gives the average gained or the avoided wasted effort of almost the entire design space of CIBI-DP vs. CVP-DP general problem. Figure 5-3 presents boxplots of all the frequency distributions of problem’s parameters for all sample’s instances.

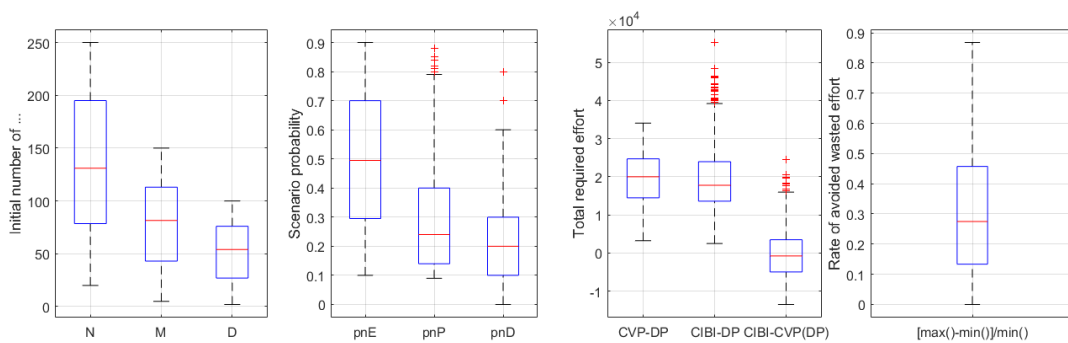


Figure 5-3: Box plots of frequency distributions of sample’s instances, concerning all the parameters of CIBI-DP vs. CVP-DP general problem

The results in Figure 5-3 provide two major inferences. Firstly, the distribution of the difference of effort estimations among design alternatives shows that CIBI-DP design alternative is preferable for approximately 55% of sample’s instances against the CVP-DP design alternative which is preferable for the rest 45%. This highlights how much ambiguous and difficult is the decision-making process of CIBI-DP vs. CVP-DP general problem. Secondly, the mean or the average of the gained or avoided wasted effort is

approximately equal to 30%, while the half of the sample’s instances lie from 15% to 45%. This highlights the overall beneficial contribution of the proposed modeling theory and derived formal models in terms of avoided wasted effort, concerning the CIBI-DP vs. CVP-DP general problem.

### 5.3 Attaching Mediator and Observer Design Patterns

#### 5.3.1 Problem Description

In this subsection, an alternate extension of the CIBI vs. CVP problem is discussed by attaching the Mediator and Observer design patterns. The Composite (CP) design pattern is often combined with the Mediator (MP) design pattern. MP design pattern defines an object (e.g., concrete mediator) that encapsulates or hardcodes the way in which a set of objects (e.g., CP elements) interact. Mediator promotes loose coupling by blocking objects from referring to each other explicitly. Furthermore it allows changing their interaction independently, as analyzed in (Gamma et al., 1994) and presented in Figure 5-4.

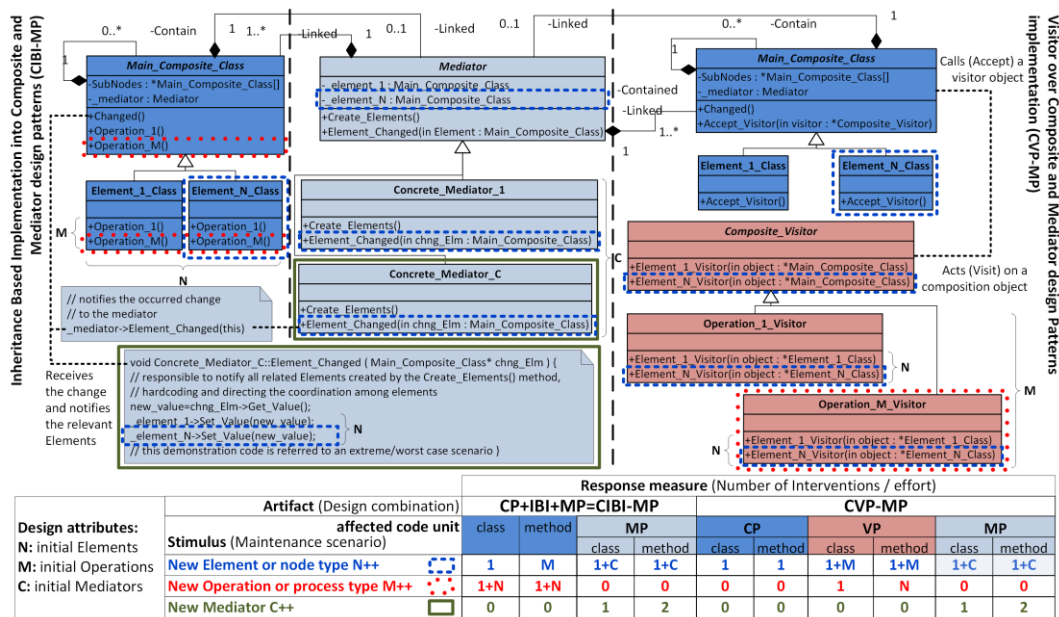


Figure 5-4: Conceptual UML class-diagram of CIBI-MP vs CVP-MP design combinations.

A similar alternative to Mediator pattern is the Observer (OP) design pattern. OP design pattern defines a one-to-many dependency among objects (e.g. CP elements) so that when one object changes state, all its dependents are notified and updated automatically, as analyzed in (Gamma et al., 1994) and presented in Figure 5-5. In OP, the way a set of objects interact is encapsulated or hardcoded in an external, separate code entity. The selection between MP and OP is not only relevant on the low-level object-oriented design. It is still relevant in high-level architectural design, dealing with communication, interfacing, and coordination issues among system’s sub-modules and components including legacy code. At the same time, the interacting objects may be implemented by the Composition design pattern (CP), the operations of which can be implemented through CVP and CIBI design combinations, as discussed in the initial problem. Thus, in this case, four alternative design combinations arise (CVP-MP, CIBI-MP, CVP-OP, CIBI-OP).

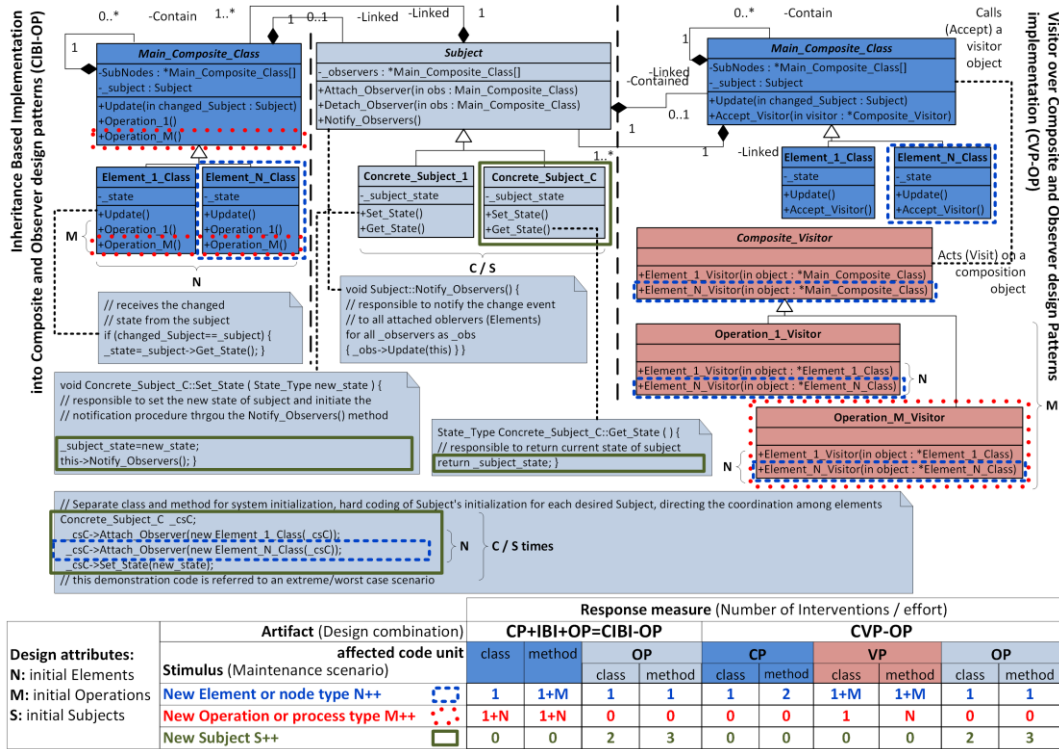


Figure 5-5: Conceptual UML class-diagram of CIBI-OP vs CVP-OP design combinations.

The selection between these alternatives can be supported by the proposed modeling method in chapters 3 and 4. The key design attributes are enhanced as N, M, and C or S for the initial number of mediators or observers. Furthermore, the major maintenance scenarios and their probabilities are enhanced as  $p_{nE}$ ,  $p_{nO}$ , and  $p_{nM}$  or  $p_{nO}$  for adding a new mediator or observer. Using the response measures or SMC metrics in Figure 5-4 and Figure 5-5, the new problem can be easily formulated through the provided modeling method framework in subsection 4.4.

### 5.3.2 Derivation of Effort Measurement and Formal Models

A (dynamic) formal model generation through the modeling method framework using MATLAB® code is provided online (Karanikolas et al., n.d.-b) and discussed in subsection 4.4. In addition, the modeling framework adapted to the extended with the Mediator and Observer design patents CIBI-MP vs. CVP-MP vs. CIBI-OP vs. CVP-OP general designing problem is presented in Code C.1 of Appendix A. In Listing 5-2, the parameters of the current general problem are loaded to the sets and matrixes of the proposed modeling framework.

Listing 5-2: Loading CIBI-MP vs CVP-MP vs CIBI-OP vs CVP-OP general problem to MatLab Modeling Framework

```

55. % Data describing (general, extended) comparison problem CIBI_MP vs CVP_MP vs CIBI_OP vs CVP_OP and fundamental (SMC) metric
    analysis
56. % In this section, different or alternate problems should be described
57. D = { 'CIBI and MP', 'CVP and MP', 'CIBI and OP', 'CVP and OP' }; % tags of Design comb. under comparison
58. L_tags = {'N', 'M', 'C'}; % tags of design attributes: N initial elements, M initial operations, C initial mediators or observers
59. A = {'Method aspect', 'Class aspect'}; % tags of Structural aspects
60. S = {'nE', 'nP', 'nM'}; % Types of maintenance scenarios: nE new composition element, nP new operation, nM new mediator or observer
61. F = [1 0 0; 0 1 0; 0 0 1]; % N:+1 M:+0 C:+0 for nE, N:+0 M:+1 C:+0 for nP, N:+0 M:+0 C:+1 for nM (change rates of affected design
    attributes for each scenario type |S|x|L|)
62. % SMC metric factors on design attributes L, for each design combination D, scenario S, and aspect A are stated in K array
63. K = zeros([size(D,2) size(S,2) size(A,2) size(L_tags,2)+1]); %creates empty matrix dimensions: |D|x|S|x|A|x|L|+1
64. % method class (structural aspects)
65. % D S N M C - N M C -
66. K(1,1,:,:) = [0 1 1 1; 0 0 1 2]; %CIBI_MP for nE: 0N+1M+1C+1 method + 0N+0M+1C+1 class = M+2C+3
67. K(1,2,:,:) = [1 0 0 1; 1 0 0 1]; %CIBI_MP for nP: 1N+0M+0C+1 method + 1N+0M+0C+1 class = 2(N+1)
68. K(1,3,:,:) = [0 0 0 2; 0 0 0 1]; %CIBI_MP for nM: 0N+0M+0C+2 method + 0N+0M+0C+1 class = 3
69. % -----
70. K(2,1,:,:) = [0 1 1 3; 0 1 1 3]; %CIBI_MP for nE: 0N+1M+1C+3 method + 0N+1M+1C+3 class = 2M+2C+6
71. K(2,2,:,:) = [1 0 0 0; 0 0 0 1]; %CIBI_MP for nP: 1N+0M+0C+0 method + 0N+0M+0C+1 class = N+1
72. K(2,3,:,:) = [0 0 0 2; 0 0 0 1]; %CIBI_MP for nM: 0N+0M+0C+2 method + 0N+0M+0C+1 class = 3
73. % -----
74. K(3,1,:,:) = [0 1 0 2; 0 0 0 2]; %CVP_OP for nE : 0N+1M+0C+2 method + 0N+0M+0C+2 class = M+4
75. K(3,2,:,:) = [1 0 0 1; 1 0 0 1]; %CVP_OP for nP : 1N+0M+0C+1 method + 1N+0M+0C+1 class = 2(N+1)
76. K(3,3,:,:) = [0 0 0 3; 0 0 0 2]; %CVP_OP for nM : 0N+0M+0C+3 method + 0N+0M+0C+2 class = 5
77. % -----
78. K(4,1,:,:) = [0 1 0 4; 0 1 0 3]; %CVP_OP for nE : 0N+1M+0C+4 method + 0N+1M+0C+3 class = 2M+7
79. K(4,2,:,:) = [1 0 0 0; 0 0 0 1]; %CVP_OP for nP : 1N+0M+0C+0 method + 0N+0M+0C+1 class = N+1
80. K(4,3,:,:) = [0 0 0 3; 0 0 0 2]; %CVP_OP for nM : 0N+0M+0C+3 method + 0N+0M+0C+2 class = 5
    
```

The macro generates one formal model equation for each alternate design combination and produces its results as a single graph for specific design attributes and scenarios' probabilities. More specifically, the generated formal models are returned in the form of the symbolic expression  $cost\_DSAL(D)$  where  $D=\{CIBI-MP, CVP-MP, CIBI-OP, CVP-OP\}$ . Thus, the total required effort for CIBI-MP design alternative is given by the equation (5-3).

$$\begin{aligned}
 c(\lambda)_{CIBI-MP} = & \\
 & \frac{2 \cdot p_{nM} + 3 \cdot p_{nP}}{2} \cdot p_{nE} \cdot \lambda^2 + \\
 & + (3 \cdot p_{nE} + 3 \cdot p_{nM} + 2 \cdot p_{nP} + 2 \cdot C \cdot p_{nE} + M \cdot p_{nE} + 2 \cdot N \cdot p_{nP})\lambda
 \end{aligned} \tag{5-3}$$

Respectively, the total required effort for CVP-MP design alternative is given by the equation (5-4).

$$\begin{aligned}
 c(\lambda)_{CVP-MP} = & \\
 & \frac{2 \cdot p_{nM} + 3 \cdot p_{nP}}{2} \cdot p_{nE} \cdot \lambda^2 + \\
 & + (3 \cdot p_{nM} + p_{nP} + N \cdot p_{nP} + 2 \cdot p_{nE} \cdot (C + M + 3))\lambda
 \end{aligned} \tag{5-4}$$

The total required effort for CIBI-OP design alternative is given by the equation (5-5).

$$\begin{aligned}
 c(\lambda)_{CIBI-OP} = & \\
 & \frac{3}{2} \cdot p_{nE} \cdot p_{nP} \cdot \lambda^2 + (4 \cdot p_{nE} + 5 \cdot p_{nM} + 2 \cdot p_{nP} + M \cdot p_{nE} + 2 \cdot N \cdot p_{nP})\lambda
 \end{aligned} \tag{5-5}$$

Finally, the total required effort for CVP-OP design alternative is given by the equation (5-6).

$$c(\lambda)_{CVP-OP} = \tag{5-6}$$

$$\frac{3}{2} \cdot p_{nE} \cdot p_{nP} \cdot \lambda^2 + (7 \cdot p_{nE} + 5 \cdot p_{nM} + p_{nP} + 2 \cdot M \cdot p_{nE} + N \cdot p_{nP})\lambda$$

Using these equations, the selection of the most maintainable design combination for any number of future scenario’s interventions ( $\lambda$ ) can be supported.

### 5.3.3 Formal Model Application in Examples of Practical Specific Problems

As an application example, consider a GUI system with the following initial design attributes and scenario probabilities  $\{N=15, M=14, C=10\}$  and  $\{p_{nE}=0.5, p_{nP}=0.2, p_{nM}=0.3\}$ . For this specific case, the CIBI\_OP design combination is preferable since it is the most maintainable, requiring the lowest effort, as indicated by the results in Figure 5-6. Respectively, in the case of different probabilities e.g.  $\{p_{nE}=0.1, p_{nP}=0.2, p_{nM}=0.7\}$ , the CVP-OP design combination is preferable. It is impressive that the long-run difference, between the optimum (less effort) and worst (higher effort) design choices, exceeds almost by 100% that of best section’s effort, which is a huge amount of effort.

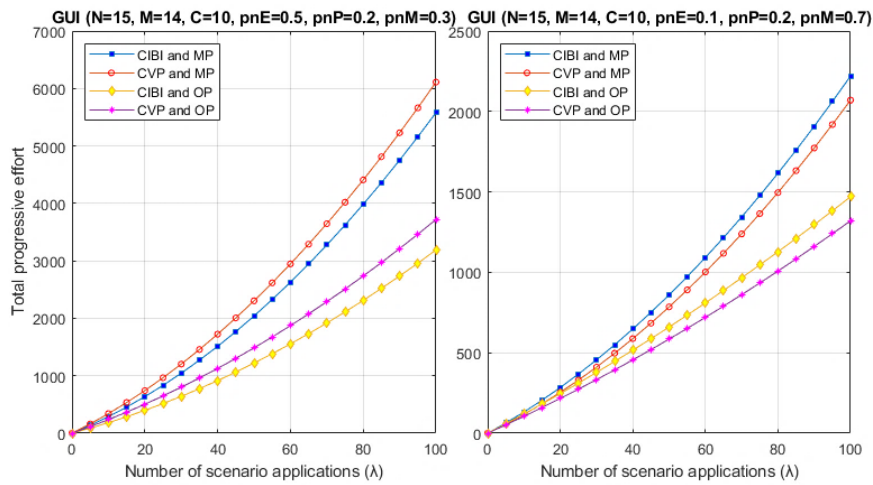


Figure 5-6: Results of the application of the Formal comparison Model on the practical examples of Graphic User Interface (GUI) specific problems as instances of CVP-MP vs. CIBI-MP vs. CVP-OP vs. CIBI-OP general problem.

### 5.3.4 Average Rate of Gained or Avoided Wasted Effort

In this subsection, the exploration of almost the entire design space of CVP-MP vs. CIBI-MP vs. CVP-OP vs. CIBI-OP general problem is attempted. A sample of one thousand sets of parameters has been randomly selected through a random generator of uniform distributions. The range for each problem’s variable was defined as  $N=[20, \dots, 250]$ ,  $M=[5, \dots, 150]$ ,  $C/S=[1, \dots, 100]$ ,  $p_{nE}=[0.1, \dots, 0.9]$ ,  $p_{nP}=[0.09, \dots, \text{rest to } 1]$ ,  $p_{nM}/p_{nO}=[0.01, \dots, \text{rest to } 1]$ . The derived formal models have been used for the estimation of the required maintenance effort per design alternative (i.e., CIBI-MP, CVP-MP, CIBI-OP, CVP-OP). Based on these values, the rate of the gained or the avoided wasted effort has been computed for each of the sample’s instances. This rate is equal to the maximum minus the minimum values divided by the minimum value of the required effort among all design alternatives. The average of these values for all sample’s instances gives the average gained or the avoided wasted effort of almost the entire design space of CVP-MP vs. CIBI-MP vs. CVP-OP vs. CIBI-OP general problem. Figure 5-7 presents boxplots of all the frequency distributions of problem’s parameters for all sample’s instances.

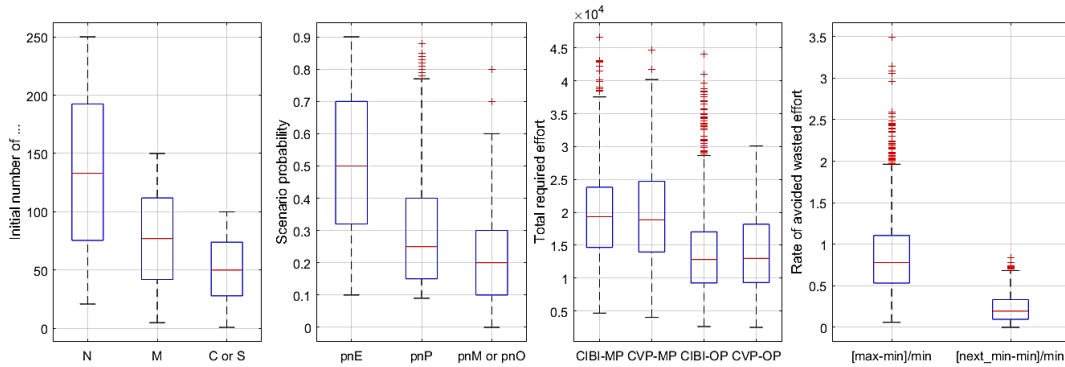


Figure 5-7: Box plots of frequency distributions of sample’s instances, concerning all the parameters of CIBI-DP vs. CVP-DP general problem

The results in Figure 5-7 provide some major inferences. Firstly, the distribution of the effort estimations among design alternatives shows that CIBI-OP design alternative is preferable for approximately 51% of sample’s instances against the CVP-OP design alternative which is preferable for the rest 49%. Furthermore, the rest design alternatives (CIBI-MP, CVP-MP) seems that they are never preferable approaching 0% of sample’s instances. This is a direct indication that the Observer is generally superior to Mediator design pattern in respect to their maintainability perspective. The equations (5-5) and (5-6) provide an explanation since they are not dependent on C or S (initial number of concrete subjects or observers) factor. In the Observer design pattern, all the required adaptations related to existing of new subjects of observers are concentrated and hardcoded in a single initialization method, as indicated in Figure 5-5. It seems that if there is no other particular reason for using Mediator design pattern, it should be avoided as far as concern its maintainability perspective in the context of the specific general problem.

However, the decision-making among the CIBI-OP and CVP-OP design alternatives remains difficult. This highlights how much ambiguous and difficult is the decision-making process of CVP-MP vs. CIBI-MP vs. CVP-OP vs. CIBI-OP general problem. Secondly, the mean or the average of the gained or avoided wasted effort is approximately equal to 88%, while the half of the sample’s instances lie from 60% to 110%. Even when the avoided wasted effort is compared to the second-best solution, the average of the gained or avoided wasted effort is approximately equal to 23%, while the half of the sample’s instances lie from 10% to 30%. This highlights the overall beneficial contribution of the proposed modeling theory and derived formal models in terms of avoided wasted effort, concerning the MP vs. CIBI-MP vs. CVP-OP vs. CIBI-OP general problem.

## 5.4 Attaching Abstract Factory and Prototype Design Patterns

### 5.4.1 Problem Description

In this section, an alternative extension of the CIBI vs. CVP problem is discussed by attaching the Abstract Factory and Prototype design patterns. In many real-life systems, elements of part-whole aggregations, represented by Composite (CP) design pattern, should be implemented in different variants or families to support multiple cases such as look-and-feel standards with different appearances and behavior. The Abstract Factory (AF) design pattern provides an interface for creating families of related or dependent objects (e.g., CP elements) without specifying their concrete classes. AF allows loose coupling by avoiding hardcoding the instantiation of family-specific objects. This makes changing all object instantiations from a different family easy, as analyzed in (Gamma et al., 1994) and presented in Figure 5-8.

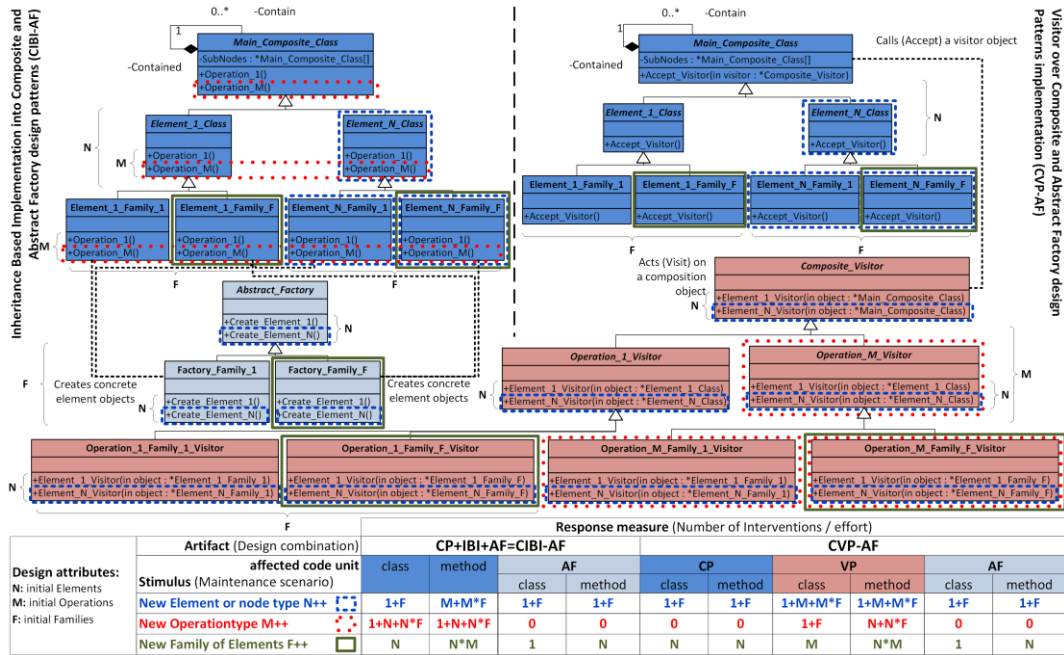


Figure 5-8: Conceptual UML class-diagram of CIBI-AF vs CVP-AF design combinations.

A similar alternative to Abstract Factory pattern is the Prototype (PT) design pattern. PT design pattern specifies the families of objects (e.g. CP elements) to create using prototype instances, and creates new objects by copying or cloning this prototypes, as analyzed in (Gamma et al., 1994) and presented in Figure 5-9. In PT, the instantiation of those family-specific prototype instances is encapsulated or hard-coded in a separate code entity. Thus, in this case, four different alternative design combinations exist (CVP-AF, CIBI-AF, CVP-PT, CIBI-PT).

The selection among which can be supported by the proposed modeling method in chapters 3 and 4. The key design attributes are enhanced as N, M, and F for the initial number of families of objects. Furthermore, the major maintenance scenarios and their probabilities are enhanced as  $p_{nE}$ ,  $p_{nF}$ , and  $p_{nM}$  for adding a new family of objects. It should be noted that usually the probability of a new family of objects is very small compared to other more frequently appeared scenarios during maintenance. However, its impact on system's code is significant, requiring extensive interventions. Thus, it is an important scenario that should be modeled. Using the response measures or SMC metrics in Figure 5-8 and Figure 5-9, the new problem can be formulated through the provided modeling method framework in subsection 4.4.

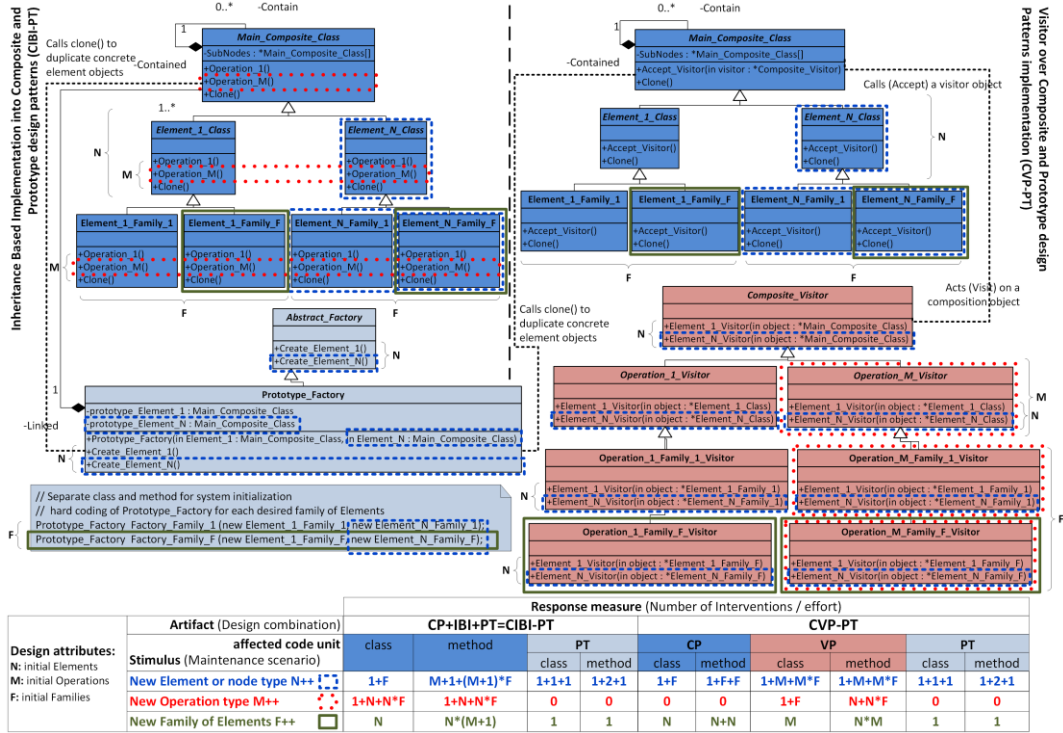


Figure 5-9: Conceptual UML class-diagram of CIBI-PT vs CVP-PT design combinations, including basic design attributes (N,M,F) and analysis of the affected code units per major maintenance scenario.

#### 5.4.2 Derivation of Effort Measurement and Formal Models

The generated formal models are returned in the form of the symbolic expression cost<sub>DSAL</sub>(D) where D={CVP-AF, CIBI-AF, CVP-PT, CIBI-PT}. Thus, the total required effort for CVP-AF design alternative is given by the equation (5-7).

$$c(\lambda)_{CVP-AF} = \frac{4}{3} \cdot p_{nE} \cdot p_{nF} \cdot p_{nP} \cdot \lambda^3 + \frac{1}{2} \left( \begin{matrix} p_{nE} \left( \frac{11 \cdot p_{nF}}{5} + \frac{11 \cdot p_{nP}}{10} + 2 \cdot F \cdot p_{nE} + 2 \cdot M \cdot p_{nF} \right) + \\ p_{nP} \left( \frac{p_{nE}}{10} + p_{nF} + F \cdot p_{nE} + N \cdot p_{nF} \right) + \\ p_{nF} \left( \frac{6 \cdot p_{nE}}{5} + p_{nP} + M \cdot p_{nE} + N \cdot p_{nP} \right) \end{matrix} \right) \lambda^2 + \left( \begin{matrix} p_{nP} \left( F + \frac{N}{10} + F \cdot N + 1 \right) + \\ p_{nF} \left( M + \frac{6 \cdot N}{5} + M \cdot N + 1 \right) + \\ p_{nE} \left( \frac{11 \cdot F}{5} + \frac{11 \cdot M}{10} + 2 \cdot F \cdot M + \frac{33}{10} \right) \end{matrix} \right) \lambda \quad (5-7)$$

Respectively, the total required effort for CIBI-AF design alternative is given by the equation (5-8).

$$c(\lambda)_{CIBI-AF} = \frac{4}{3} \cdot p_{nE} \cdot p_{nF} \cdot p_{nP} \cdot \lambda^3 + \frac{1}{2} \left( \begin{matrix} p_{nE} \left( \frac{21 \cdot p_{nF}}{10} + \frac{p_{nP}}{10} + F \cdot p_{nE} + M \cdot p_{nF} \right) + \\ p_{nP} \left( \frac{11 \cdot p_{nE}}{10} + 2 \cdot F \cdot p_{nE} + 2 \cdot N \cdot p_{nF} \right) + \\ p_{nF} \left( \frac{11 \cdot p_{nE}}{10} + M \cdot p_{nE} + N \cdot p_{nP} \right) \end{matrix} \right) \lambda^2 + \left( \begin{matrix} p_{nE} \left( \frac{21 \cdot F}{10} + \frac{M}{10} + F \cdot M + \frac{21}{10} \right) + \\ p_{nP} \left( \frac{11 \cdot N}{10} + 2 \cdot F \cdot N + \frac{11}{10} \right) + \\ p_{nF} \left( \frac{11 \cdot N}{10} + M \cdot N + 1 \right) \end{matrix} \right) \lambda \quad (5-8)$$

The total required effort for CVP-PT design alternative is given by the equation (5-9).



$$\begin{aligned}
 c(\lambda)_{CVP-PT} = & \\
 & \frac{4}{3} \cdot p_{nE} \cdot p_{nF} \cdot p_{nP} \cdot \lambda^3 + \\
 & \frac{1}{2} \left( \begin{aligned} & p_{nE} \left( \frac{21 \cdot p_{nF}}{10} + \frac{11 \cdot p_{nP}}{10} + 2 \cdot F \cdot p_{nP} + 2 \cdot M \cdot p_{nF} \right) + \\ & p_{nP} \left( \frac{p_{nE}}{10} + p_{nF} + F \cdot p_{nE} + N \cdot p_{nF} \right) + \\ & p_{nF} \left( \frac{21 \cdot p_{nE}}{10} + p_{nP} + M \cdot p_{nE} + N \cdot p_{nP} \right) \end{aligned} \right) \lambda^2 + \left( \begin{aligned} & p_{nP} \left( F + \frac{N}{10} + F \cdot N + 1 \right) + \\ & p_{nF} \left( M + \frac{21 \cdot N}{10} + M \cdot N + 2 \right) + \\ & p_{nE} \left( \frac{21 \cdot F}{10} + \frac{11 \cdot M}{10} + 2 \cdot F \cdot M + \frac{32}{5} \right) \end{aligned} \right) \lambda
 \end{aligned} \quad (5-9)$$

Finally, the total required effort for CIBI-PT design alternative is given by the equation (5-10).

$$\begin{aligned}
 c(\lambda)_{CIBI-PT} = & \\
 & \frac{4}{3} \cdot p_{nE} \cdot p_{nF} \cdot p_{nP} \cdot \lambda^3 + \\
 & \frac{1}{2} \left( \begin{aligned} & p_{nE} \left( 2 \cdot p_{nF} + \frac{p_{nP}}{10} + F \cdot p_{nP} + M \cdot p_{nF} \right) + \\ & p_{nP} \left( \frac{11 \cdot p_{nE}}{10} + 2 \cdot F \cdot p_{nE} + 2 \cdot N \cdot p_{nF} \right) + \\ & p_{nF} \left( 2 \cdot p_{nE} + M \cdot p_{nE} + N \cdot p_{nP} \right) \end{aligned} \right) \lambda^2 + \left( \begin{aligned} & p_{nE} \left( 2 \cdot F + \frac{M}{10} + F \cdot M + \frac{26}{5} \right) + \\ & p_{nP} \left( \frac{11 \cdot N}{10} + 2 \cdot F \cdot N + \frac{11}{10} \right) + \\ & p_{nF} \left( 2 \cdot N + M \cdot N + 2 \right) \end{aligned} \right) \lambda
 \end{aligned} \quad (5-10)$$

Using these equations, the selection of the most maintainable design combination for any number of future scenario’s interventions ( $\lambda$ ) can be supported. Notice that the derived equations for the latest general problem are quite complex. This is due to the existence of families of objects that requires extensive interventions to be made for any possible scenario type as indicated by the response measures or SMC metrics in Figure 5-8 and Figure 5-9.

### 5.4.3 Formal Model Application in Examples of Practical Specific Problems

As an application example, consider a GUI with the following initial design attributes and scenario probabilities  $\{N=55, M=20, F=2\}$  and  $\{p_{nE}=0.88, p_{nP}=0.1, p_{nF}=0.02\}$ . For this case, the CIBI-AF design combination is preferable since it is the most maintainable, requiring the lowest effort, as indicated by the results in Figure 5-10. In case of different probabilities e.g.  $\{p_{nE}=0.48, p_{nP}=0.5, p_{nF}=0.02\}$ , the CVP-AF and CVP-PT design combinations are almost equally preferred. The long-run amount of gained or avoided/wasted effort is once more significant.

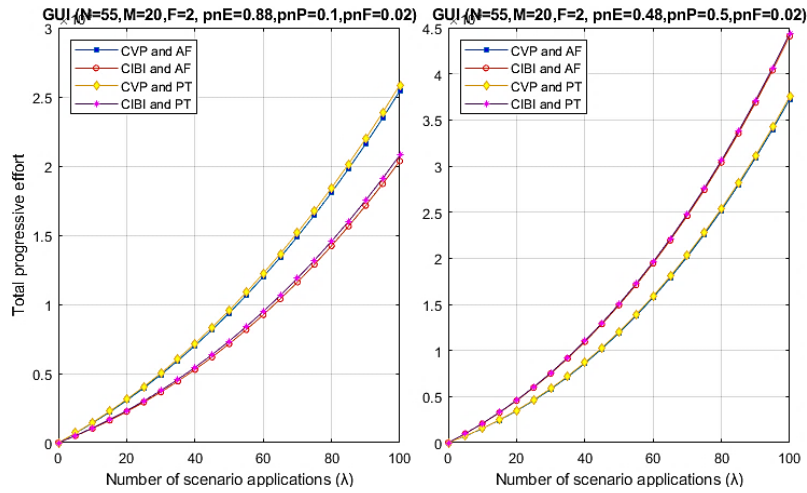


Figure 5-10: Results of the application of Formal comparison Model on the practical examples of Graphic User Interface (GUI) specific problems as instances of CVP-AF vs. CIBI-AF vs. CVP-PT vs. CIBI-PT general problem.

#### 5.4.4 Average Rate of Gained or Avoided Wasted Effort

In this subsection, the exploration of almost the entire design space of CVP-AF vs. CIBI-AF vs. CVP-PT vs. CIBI-PT general problem is attempted. A sample of one thousand sets of parameters has been randomly selected through a random generator of uniform distributions. The range for each problem’s variable was defined as  $N=[20, \dots, 250]$ ,  $M=[5, \dots, 150]$ ,  $F=[1, \dots, 10]$ ,  $p_{nF}=[0.01, \dots, 0.10]$ ,  $p_{nE}=[0.01, \dots, \text{rest to } 1]$ ,  $p_{nP}=[0.01, \dots, \text{rest to } 1]$ . The derived formal models have been used for the estimation of the required maintenance effort per design alternative (i.e., CVP-AF, CIBI-AF, CVP-PT, CIBI-PT). Based on these values, the rate of the gained or the avoided wasted effort has been computed for each of the sample’s instances. This rate is equal to the maximum minus the minimum values divided by the minimum value of the required effort among all design alternatives. The average of these values for all sample’s instances gives the average gained or the avoided wasted effort of almost the entire design space of CVP-AF vs. CIBI-AF vs. CVP-PT vs. CIBI-PT general problem. Figure 5-11 presents boxplots of all the frequency distributions of problem’s parameters for all sample’s instances.

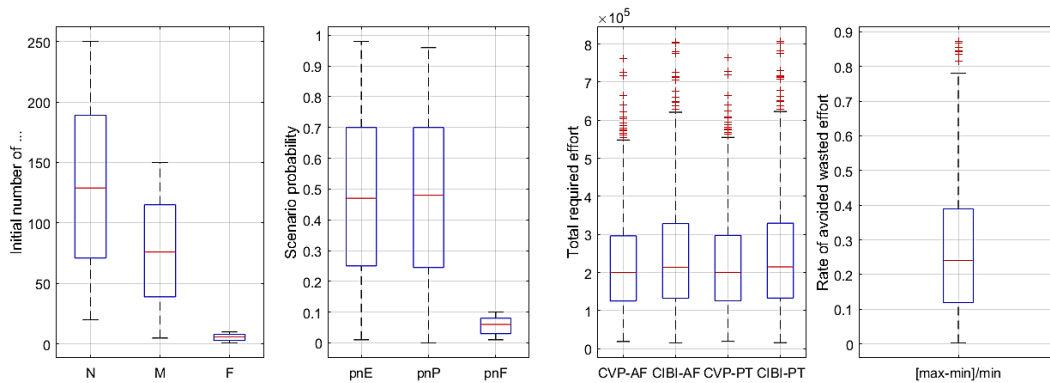


Figure 5-11: Box plots of frequency distributions of sample’s instances, concerning all the parameters of CVP-AF vs. CIBI-AF vs. CVP-PT vs. CIBI-PT general problem

The results in Figure 5-11 provide some major inferences. Firstly, the distribution of the effort estimations among design alternatives shows that CVP-AF design alternative is preferable for approximately 63% of sample’s instances against the CIBI-AF design alternative which is preferable for the rest 37%. Furthermore, the rest design alternatives (CVP-PT, CIBI-PT) seems that they are never preferable approaching 0% of sample’s instances. This is a direct indication that the Abstract Factory is generally superior to Prototype design pattern in respect to their maintainability perspective. It seems that if there is no other particular reason for using Prototype design pattern, it should be avoided as far as concern its maintainability perspective in the context of the specific general problem.

However, the decision-making among the CVP-AF and CIBI-AF design alternatives remains difficult. This highlights how much ambiguous and difficult is the decision-making process of CVP-AF vs. CIBI-AF vs. CVP-PT vs. CIBI-PT general problem. Secondly, the mean or the average of the gained or avoided wasted effort is approximately equal to 27%, while the half of the sample’s instances lie from 12% to 39%. This highlights the overall beneficial contribution of the proposed modeling theory and derived formal models in terms of avoided wasted effort, concerning the CVP-AF vs. CIBI-AF vs. CVP-PT vs. CIBI-PT general problem.

### 5.5 Average Rate of Gained or Avoided Wasted Effort of CVP vs CIBI

In this subsection, the exploration of almost the entire design space of CIBI vs. CVP general problem, analyzed in chapters 3 and 4, is attempted. A sample of one thousand sets of parameters has been randomly selected through a random generator of uniform

distributions. The range for each problem’s variable was defined as  $N=[20, \dots, 250]$ ,  $M=[5, \dots, 150]$ ,  $p_{nE}=[0.1, \dots, 0.9]$ ,  $p_{nP}=[0.1, \dots, \text{rest to } 1]$ . The derived formal models as generated by the framework in subsection 4.4.3 (equations (4-8) and (4-12)) have been used for the estimation of the required maintenance effort per design alternative (i.e., CIBI, CVP) as well as their difference. Based on these values, the rate of the gained or the avoided wasted effort has been computed for each of the sample’s instances. This rate is equal to the maximum minus the minimum ( $\max(\text{CIBI}, \text{CVP}) - \min(\text{CIBI}, \text{CVP})$ ) divided by the minimum ( $\min(\text{CIBI}, \text{CVP})$ ) of the required effort among all design alternatives. The average of these values for all sample’s instances gives the average gained or the avoided wasted effort of almost the entire design space of CIBI vs. CVP general problem. Figure 5-12 presents boxplots of all the frequency distributions of problem’s parameters for all sample’s instances.

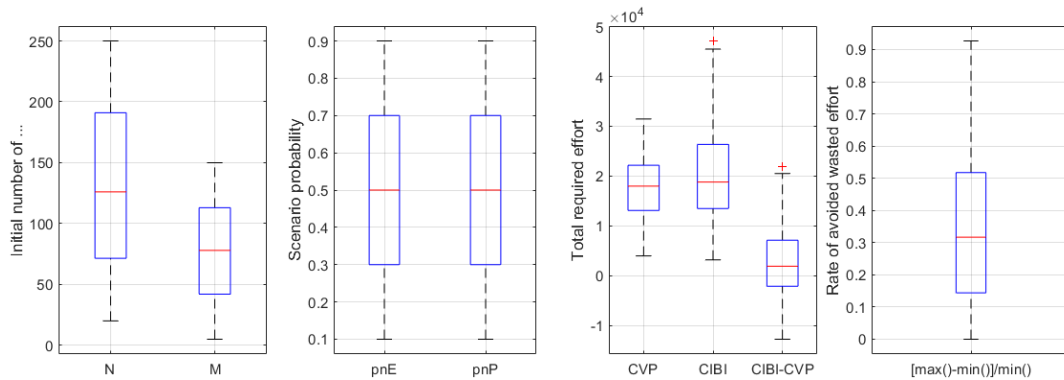


Figure 5-12: Box plots of frequency distributions of sample’s instances, concerning all the parameters of CIBI vs. CVP general problem

The results in Figure 5-12 provide two major inferences. Firstly, the distribution of the difference of effort estimations among design alternatives shows that CVP design alternative is preferable for approximately 62% of sample’s instances against the CIBI design alternative which is preferable for the rest 38%. This highlights how much ambiguous and difficult is the decision-making process of CIBI vs. CVP general problem. Secondly, the mean or the average of the gained or avoided wasted effort is approximately equal to 34%, while the half of the sample’s instances lie from 15% to 52%. This highlights the overall beneficial contribution of the proposed modeling theory and derived formal models in terms of avoided wasted effort, concerning the CIBI vs. CVP general problem.

## 5.6 Summarizing the Contribution of Modeling Method

In this subsection, the beneficial contribution of the proposed modeling method, as analyzed in previous subsection for different general and significant design problems, is summarized in Table 5-1.

Table 5-1: Overall results for 1000 instances of design attributes and scenario’s probabilities per General Problem

General Problem	Design Attribute / Scenario	Design Variation (Pattern)	Preferable <sup>1</sup>	Average Benefit <sup>2</sup>	Benefit Interval 50% <sup>3</sup>
Recursive hierarchy of part-whole representation	Element Operation	CP+IBI=CIBI	38%	34%	15% - 52%
		CP+VP=CVP	62%		
Recursive hierarchy of part-whole representation with extra Responsibilities	Element Operation Responsibility	CIBI+DP	55%	30%	15% - 45%
		CVP+DP	45%		
Recursive hierarchy of part-whole representation with independent interaction	Element Operation Mediator or Observer	CVP+MP	≈0%	88%	60% - 110%
		CIBI+MP	≈0%		
		CVP+OP	49%		
		CIBI+OP	51%		
Recursive hierarchy of part-whole representation with families of objects	Element Operation	CVP+AF	63%	27%	12% - 39%
		CIBI+AF	37%		
		CVP+PT	≈0%		

General Problem	Design Attribute / Scenario	Design Variation (Pattern)	Preferable <sup>1</sup>	Average Benefit <sup>2</sup>	Interval 50% <sup>3</sup>
	Abstract factory or Prototype object	CIBI+PI <sup>1</sup>	≈0%		

<sup>1</sup> Refers to the portion of sample's instances for which the design alternative is preferable (requires the minimum effort)

<sup>2</sup> Refers to the average of the rates of gained or avoided effort for the entire sample

<sup>3</sup> Refers to the interval of the rates of gained or avoided effort for the half instances of each sample

The results in Table 5-1 suggest that the overall benefit, or else, the avoided wasted effort from the use of the derived formal models corresponds to a significant portion of the optimal maintenance cost which on average lies between 25% and 90% concerning almost the entire design space of each general problem under study.

Based on the proposed modeling method and by using the programable MATLAB framework (Appendix A) as an initial template the cost of analyzing, deriving, and testing the formal comparison models for the design alternatives of each general problem presented in this study ranged approximately between 3 (for the basic CVP vs CIBI problem in Chapter 4) and 6 working man hours (for the most complicate problems presented in this chapter) for a typical software engineer. These derivation costs are quite reasonable compared to the actual benefits of using the derived formal models. Given that the derived formal models are easily reusable in a general family of common problems, preventing significant loses in terms of maintenance effort; the future benefits of the proposed technique significantly outweigh its reasonable analysis cost. A more detailed and formal justification about the maximum allowed derivation cost of formal comparison models is presented in (Bass et al., 2012) and further specialized subsection 4.4.5.

## 5.7 Conclusions

### 5.7.1 General Requirements and Limitations

In general, the analysis of the alternate general problems presented in this chapter are subject to the same requirements and limitations of the introduced modeling method as described in subsections 3.6 and 4.6. Furthermore, the actual design structure for a specific system may deviate from the typical class-diagrams based on which the proposed formal models have been derived. This is a possible threat to validity regarding the reliability of the decisions made based on effort predictions of those formal models.

Moreover, it is important that the conducted analysis must be specific enough to capture all the principal components of each general problem, while at the same time, general enough bypassing minor functionalities and technical details related to code implementation to support early decisions and avoid unnecessary complexity. Consequently, selecting and analyzing the proper (major) maintenance scenarios and design attributes for a given general design problem is a critical and creative task that may negatively affect the reliability of the derived formal models.

### 5.7.2 Extensions and Further Research

All the potentials and research perspectives, as referred in subsections 3.6 and 4.6, are still valid for the alternate general problems presented in this chapter since they are all subject to the same principles imposed by the introduced modeling theory. In addition, an interesting perspective could be the mathematical analysis of the derived formal models for each variation of the general problem. A such analysis could reveal possible similarities, differences, conflicts, requirements, limitations, and patterns regarding the evolution and structural behavior of the used design patterns, thus providing further insight about the maintenance perspective of each design alternative under comparison, or even about the nature the general problem itself.

### 5.7.3 Overall Assessment

Evaluation results using extended problems such as Observer vs. Mediator indicate that the proposed method can be also applied during the high-level architecture design, to

handle communication, interfacing, and coordination issues among sub-modules and components of new or even legacy code. These examples prove the applicability of the proposed modeling method in a wide spectrum of common and difficult software architectural design problems.

Furthermore, the results of the indicative examples suggest that the overall benefit, or else, the avoided wasted effort from the use of the derived formal models corresponds to a significant portion of the optimal maintenance cost which on average lies between 25% and 90% concerning almost the entire design space of the general problems under study. These results highlight the beneficial contribution of the proposed modeling method and derived formal models to the early decision-making among design alternatives in terms of avoided wasted effort during software maintenance.

Finally, the deterministic nature of the derived formal models combined with their computational efficiency through software, allows the exploration of the entire design space for a given general and significant design problem in the field of software engineering.

## 6 Simulation of Software Evolution

### 6.1 Chapter Overview

In this chapter, the proposed modeling method and derived formal models for the CVP vs. CIBI general problem, analyzed in chapter 3 and 4, are statistically evaluated based on massive and homogenous measurement observations which have been generated by a well calibrated and highly stochastic simulation model that imitates the variability and underlying activities of actual maintenance process. The proposed modeling theory is strongly related to structural behavior of well-known and established design patterns (Gamma et al., 1994) during maintenance. This modeling method derives formal comparison models among design alternatives for general, significant, and frequently tackled design problems while provides reusable formal models sensitive to several design parameters making them easily end repeatedly applied to a wide spectrum of specific instances of each general problem. More specifically, the derived formal modes try to predict the impact for each design alternative selection in terms of required maintenance effort in a deterministic way limiting the ambiguity imposed by the stochastic nature of actual maintenance process. The design alternative that requires the lowest predicted maintenance effort is the most beneficial, and thus preferable. The main intent of such comparison models is to provide reliable decisions through proportionally equivalent effort estimations for comparison purposes, thus away from the need of accurate effort estimations in terms of absolute values as discussed in chapter chapter 3. However, it is substantially difficult to estimate the risk taken during such decisions, or differently, the possibility of an incorrect selection of a less maintainable design alternative for a given instance of the general problem. Even if the absolute values of effort predictions returned by formal models for each design alternative are not of primary interest, their difference defines the most beneficial design alternative, and thus the outcome of the decision which is of primary interest. To validate the reliability of the decisions that are based on effort predictions of formal models, these predictions should be statistically compared to actual effort measurements of long maintenance periods of real-world systems. Yet, there is a lack of evidence regarding the effectiveness of the prediction techniques and models of software maintainability (Riaz et al., 2009a; Shepperd & MacDonell, 2012). In addition, there is a confirmed need for further validation of maintainability prediction models (Riaz et al., 2009a), primarily through statistical techniques.

As discussed in chapter 3 and 4, the derived formal models are mainly focused on maximizing the potential for being general over different instances of a given general problem. However, formal methods usually suffer from lack of realism of context and precision of measurements, as discussed in (Stol & Fitzgerald, 2018). Ideally, actual measurements and observations from case studies that maximize the potential for realism of context would be preferable for validation purposes. Nevertheless, in real life, finding identical actual systems with uniform design attributes, developed in different design variations is almost impossible. Additionally, the number of recorded observations is very limited per case study, using heterogeneous metrics, and unevenly conducted through literature. Thus, they are not statistically meaningful, heavily limiting the generalization of inferences, as pointed in (Langdon et al., 2016; Shepperd & MacDonell, 2012). Moreover, developer-related aspects, such as experience level and learning rate are also ignored by these methods since they are heavily biased, as human-related, factors hard to be assessed and measured. Because of all these reasons, there is no easy way to determine the reliability of the method referred to possible incorrect design decisions in terms of maintainability. This is a standard concern with regards to validity since the attempt to validate the formal models based on a limited number and dissimilar case studies may increase realism of context while sacrificing generalizability which should be the models' primary goal.

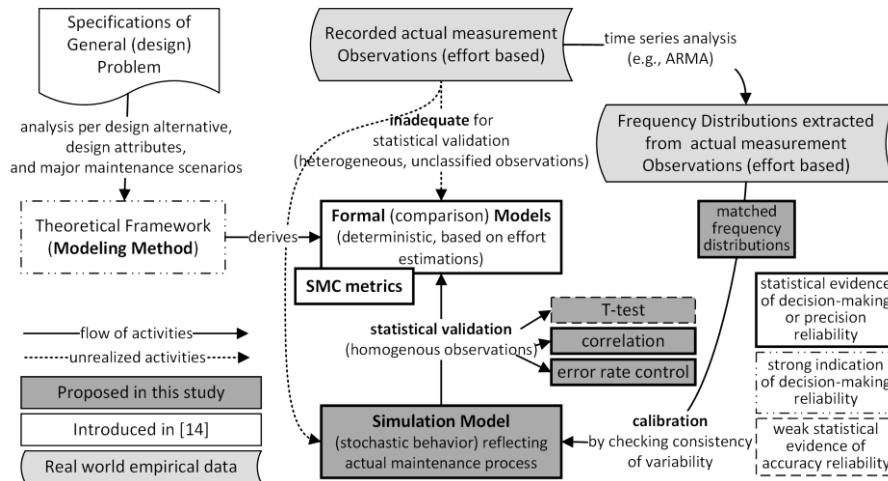


Figure 6-1: Representation of study’s goals, contribution, and limitations

This chapter introduces the simulation of system’s structural evolution for different design alternatives to generate adequate volume of homogenous observations and measurements towards a statistical validation of the decision reliability of the derived formal models in chapter 3 and 4. More specifically, the validation process concerns the significant and general decision problem between the design combinations of Visitor design pattern and Composition design pattern for data structures, both serving recursive hierarchies of part-whole aggregations. The derived formal comparison models have been validated under several statistical techniques to evaluate the proposed modeling method in chapter 3 and 4. A sample of one thousand possible system’s instances with specific design attributes and scenarios’ probabilities has been randomly selected. A simulation model that replicates the underlying activities of actual (real-world) maintenance process, providing sufficient, unbiased, classified, and homogenous validation data is introduced. The simulation model has been designed and developed in the forms of MATLAB© functional model and object-oriented entity model, engaging all problem’s parameters, and providing additional switches for controlling the simulation settings and environment. Several intermediate variations of the model based on the multi-resolution modeling technique has been tested to reach the desired stochastic behavior and realistic outcomes. However, the validation of the simulation mode’s consistency in terms of actual effort predictions is subject to the same restrictions as in the case of formal models (i.e., heterogenous real-world observations). Taking advantage from the fact that accurate effort estimations are not critical for the decision-making, the consistency of the simulation model has been verified by matching model’s variability with frequency distributions of real-world (effort based) observations as illustrated in Figure 1-6. Thus, bypassing the obstacle of a strictly validation against actual effort estimations, while connecting the simulation model with the real world and ensuring its decision-making reliability or precision. Concerning the consistency criterion, the simulation model has been calibrated based on empirical evidence (frequency distributions) of relevant studies from the field of time series analysis (Raja et al., 2009; Shariat Yazdi et al., 2016). In principle, the simulation model imitates the stochastic nature of the actual maintenance process by incorporating developers’ stochastic characteristics such as experience and learning rate as well as other random factors like uncertainty of scenarios’ probabilities, alternate maintenance scenarios, non-repeated application patterns, the actual code size of interventions, code aging issues, etc.

Several intermediate results computed by the simulation model have been compared against formal models’ deterministic predictions under the hypothesis testing of non-significant difference. The results demonstrate a high coefficient of correlation (near to 0.96) providing sufficient statistical evidence of formal model’s decision-making reliability.

Furthermore, the conducted hypothesis tests provide statistical evidence of formal models' long-term accuracy in terms of absolute effort predictions which, however, is a weak inference due to the lack of a strictly validation of the simulation model against actual (real world) estimations. Most importantly, the results showed that the formal models provide reliable decisions among design alternatives with an overall long-term error-rate about 8% with only 2% of it being critical in terms of significant wasted effort. Hence, the statistical validation of the formal models' decision-making ability is a strong indication that the introduced modeling method in chapter 3 and 4 trustworthy describes the software evolution during maintenance process, deriving reliable formal models of limited decision-risk.

The context of this chapter is based on the significant design problem of part-whole representations in chapter 3, and the modelling method and framework presented in chapter 4. The rest of this chapter is organized as follows. Subsection 6.2 presents the theoretical background of the modeling method, the general problem, and its formal models under evaluation. Subsection 6.3 refers existing evaluation evidence and further validity concerns under exploration. Subsection 6.4 analyzes the statistical validation approach and the introduced simulation model. Subsection 6.5 lists the result and inference of the experimentation process. Finally, in subsection 6.6, the validity challenges, limitations, future research issues, and conclusions are presented.

## 6.2 Background

For the sake of completeness and cohesion, in this subsection, the theoretical background of the general problem, the characteristics of the used effort metric, and the notation of the derived formal models under validation are presented. The context of this subsection is in accordance and directly related to the context in chapters 1, 3, and 4.

### 6.2.1 Example of Practical General Problem

An example of a significant and general problem is referred to the recursive implementation of various types of operations upon part-whole aggregations of different types of elements which encountered in a wide range of critical systems such as compilers, interpreters, GUIs, CADs, high-level synthesis, Domain Specific Languages, Intermediate Representations, and hierarchical frameworks. Several design alternatives to address a such general problem have been reported, usually, as a combination of well-known design patterns. Visitor and Composite are examples of established design patterns which combined can provide implementations of part-whole aggregations. The Inheritance Based Implementation into Composition (CIBI) and Visitor upon Composition (CVP) are the most prevailing design combinations capable to address this general problem, as presented in Figure 6-2. However, these design alternatives have opposite characteristics regarding their maintainability perspective.

More specifically, the main intent of the Composite design Pattern (CP) is to compose objects into tree structures to represent part-whole hierarchies (Gamma et al., 1994). CP is the basis of both design combinations and presented on both sides in Figure 6-2. The number of distinct node or element types, which can be represented by CP, is equal to the number of leaf classes, denoted as  $N$  in Figure 6-2 and Table 6-1. In a Composite structure (CP), the Inheritance Based Implementation (IBI) can be used to implement operations uniformly, as presented on the left side combination (CIBI) in Figure 6-2. All distinct operations, denoted as  $M$  in Figure 6-2 and Table 6-1, are declared as virtual methods in the abstract root class of the hierarchy. The implementation of every distinct operation (method) is placed in each distinct object (leaf) class of the hierarchy. This pattern combination makes adding new types of nodes (elements) easier (Gamma et al., 1994) thanks to the concentration (locality) of the related interventions in a single class. Alternatively, Visitor design pattern (VP) can be used over CP as presented on the right side combination (CVP) in Figure 6-2, and further analyzed in (Alexandrescu, 2001;



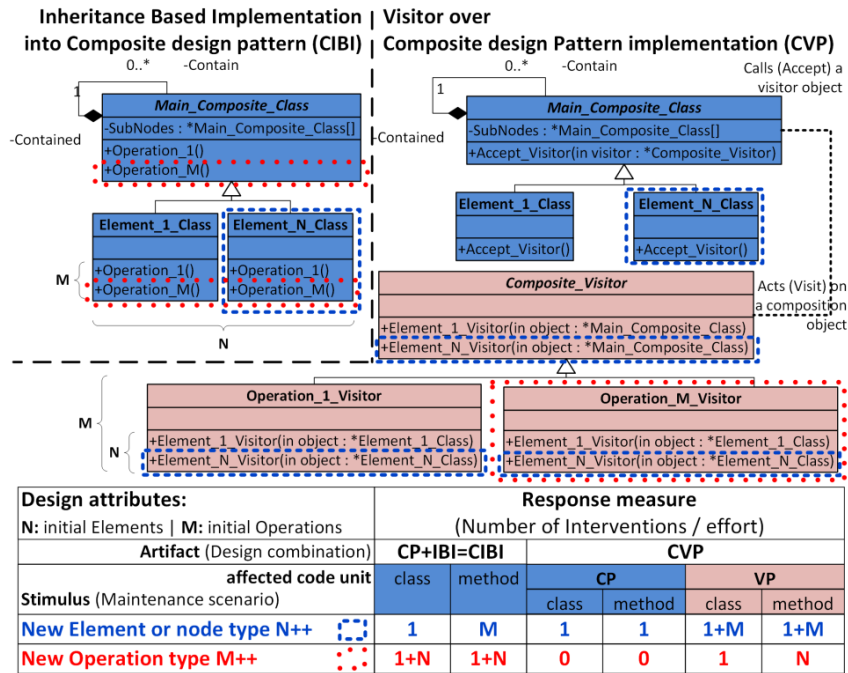


Figure 6-2: Conceptual UML classes diagram of CIBI and CVP design combinations, including basic design attributes (N,M) and analysis of the affected code units per major maintenance scenario.

Gamma et al., 1994; B. C. d. S. Oliveira et al., 2008; Palsberg & Jay, 1998; Visser, 2001). For every distinct type of CP node (leaf class), a new virtual method is declared in an abstract root class called Visitor. In addition, for every distinct operation, a new subclass is created which includes all the implementations of the methods of distinct node types for this specific operation. In contrast with IBI, VP makes adding new operations easier (Gamma et al., 1994) thanks to the concentration (locality) of the related interventions in a single visitor class.

According to the rule “design for change” (David Lorge Parnas, 1994), in order to facilitate maintainability or changeability, changes that are likely to occur over the software’s “lifetime” should be categorized. Since actual changes cannot be precisely predicted, the categorization is about classes of resembling changes. In principle, this design rule implies that logical entities that are most likely to change are “confined” to a small or grouped code entities so that if those entities do change, only a small amount of code would be affected. Towards this direction, several architectural tactics such as splitting or rearranging responsibilities target on increasing cohesion and reducing coupling among the model’s logical entities, and thus improving maintainability (Arbuckle, 2011). Referring to the general problem of part-whole aggregations, events like adding or updating or debugging a new or existing type of element constitute a major maintenance scenario. Similar events referred to a new or existing type of operation is another major scenario, as presented in Figure 6-2 and Table 6-1.

Table 6-1: Design Characteristics and Model’s Notation of the General Problem of Part-Whole Representations

Description	Notation
Design characteristics of CVP vs CIBI general problem	
Composite (design pattern) represents recursive part-whole aggregations or structures of elements	CP
Visitor (design pattern) links operations to different type of elements of a Composition	VP
Inheritance based implementation incorporates operations inside Composition’s elements	IBI
Design combination of Visitor (operations) over Composite structure (element)	CVP

Description	Notation
Design combination of direct attachment of operations inside Composite structure (element)	CIBI
Design attributes/parameters the values of which distinguishes a specific instance of the general problem	
Number of initial Elements of Composition	N
Number of initial Operations (or Processes) acting on Elements	M
probability of Adding/updating/debugging a new/existing Element	$p_{nE} = 1 - p_{nP}$
probability of Adding/updating/debugging a new/existing Operation (or Process)	$p_{nP} = 1 - p_{nE}$
Model's parameters and outcome	
Number of maintenance scenarios' applications during software evolution/maintenance	$\lambda$
Partial effort prediction per applied scenario, expressed by SMC metric in terms of number of required interventions in different code entities (e.g., classes, methods)	
New Element ( $p_{nE}=1.0, p_{nP}=0.0$ ) on CVP for $\lambda=1$	$c_m(CVP, N, M, 1.0, 0.0, 1)$
New Operation ( $p_{nE}=0.0, p_{nP}=1.0$ ) on CVP for $\lambda=1$	$c_m(CVP, N, M, 0.0, 1.0, 1)$
New Element ( $p_{nE}=1.0, p_{nP}=0.0$ ) on CIBI for $\lambda=1$	$c_m(CIBI, N, M, 1.0, 0.0, 1)$
New Operation ( $p_{nE}=0.0, p_{nP}=1.0$ ) on CIBI for $\lambda=1$	$c_m(CIBI, N, M, 0.0, 1.0, 1)$
Total effort prediction by derived Formal Model	
CVP design combination (solution / alternative)	$c_m(CVP, N, M, p_{nE}, 1 - p_{nE}, \lambda)$
CIBI design combination (solution / alternative)	$c_m(CIBI, N, M, p_{nE}, 1 - p_{nE}, \lambda)$
Total computed effort prediction per design alternative by Simulation Model	$sc_m(CVP, N, M, p_{nE}, 1 - p_{nE}, \lambda)$ $sc_m(CIBI, N, M, p_{nE}, 1 - p_{nE}, \lambda)$
Error Rate of incorrect decisions (of a single sample instance) among design alternatives in terms of maintainability	Er
Average Error Rate of incorrect decisions (of all sample instances)	avg Er
Critical Error rate of incorrect decisions of high impact (of a single sample instance)	cEr
Average Critical Error rate of incorrect decisions of high impact (of all sample instances)	avg cEr

In this general problem, the number of initial elements and operations are conceived as basic design attributes which define a specific system as an instance of the general design problem. Such design attributes are usually referred to the problem's logical entities (i.e., elements and operations) which are represented by design patterns' components such as methods, classes, or modules. The numbers of distinct element types and distinct operation types, denoted as N and M in Figure 6-2 and Table 6-1, are conceived as key design attributes referred to the problem's logical entities. Through these attributes, the number of required method and class interventions can be quantitatively expressed in the event of major maintenance scenarios as summarized in memo table in Figure 6-2 and analyzed in chapter 3 and 4.

Furthermore, since it is impossible to do everything equally easy to change, it is important to estimate the probability of each class of changes or maintenance scenario. During maintenance, several of the initial design attributes (i.e., N and M) are updated according to the behavior of the engaged design patterns based on the individual probabilities (i.e.,  $p_{nE}$  and  $p_{nP}$ ) of major maintenance scenarios, as presented in Table 6-1. Scenarios' probabilities are assessments according to the scope of each specific problem's instance.

The evolution of software during its maintenance is strongly related and mostly determined by the behavior of the engaged design patterns in future changes or stimulus or major maintenance scenarios. The number of the applied scenarios during maintenance process are denoted as  $\lambda$  in Table 6-1. More specifically, the factor  $\lambda$  represents the total number of maintenance scenarios that have been occurred and applied (by developers) on a design combination (of design patterns) during maintenance. Considering the software maintenance as an evolution process, this factor is an alternate expression of time perception.

### 6.2.2 Characteristics of SMC Effort Metric

The effect or the response of each single change or maintenance scenario is quantitatively expressed by the Structural Maintenance Cost (SMC) metric, introduced in chapter 3. For each general problem, a set of SMC metrics can be derived as presented in Figure 6-2 and Table 6-1. In general, for a given problem with x design alternatives and y major maintenance scenarios, a set of x\*y distinct SMC metrics should be derived to fully analyze the problem as discussed in chapter 3. The outcome of SMC metric (required effort) for each design alternative (i.e., CVP and CIBI) and type of major scenario (i.e., new element type and new operation type) depends on the value of the initial design attributes (i.e., N

and M) and it is referred as  $c_m()$  in Table 6-1 and Table 6-2. SMC metric is used as fundamental effort measurement of which the principal characteristics are briefly discussed.

Table 6-2: Equations of Fundamental Effort Metrics of CIBI vs. CVP General Decision Problem

Description of maintenance scenario	Equation for a single scenario application	Affected design attributes	Formal / Simulation Model
New element on CVP	$c_m(\text{CVP}, N, M, 1, 0, 1) = 2(M+2)$	N++	FM & SM
Edit/debug element on CVP	2M		SM
Delete <sup>i</sup> element on CVP	1+M	N--	SM
New operation on CVP	$c_m(\text{CVP}, N, M, 0, 1, 1) = N+1$	M++	FM & SM
Edit/debug operation on CVP	N+1		SM
Delete operation on CVP	1	M--	SM
New element on CIBI	$c_m(\text{CIBI}, N, M, 1, 0, 1) = M+1$	N++	FM & SM
Edit/debug element on CIBI	M+1		SM
Delete element on CIBI	1	N--	SM
New operation on CIBI	$c_m(\text{CIBI}, N, M, 0, 1, 1) = 2(N+1)$	M++	FM & SM
Edit/debug operation on CIBI	2N		SM
Delete <sup>ii</sup> operation on CIBI	1+N	M--	SM

Equations are derived based on the approach discussed in chapter 3

<sup>i</sup> since deleting elements in CVP is rather an efficient task, the number of required interventions is reduced from 2M to 1+M

<sup>ii</sup> since deleting operations in CIBI is rather an efficient task, the number of required interventions is reduced from 2N to 1+N

The equations of SMC metrics in Table 6-2 can be derived through the derivation approach presented in subsection 4.3.4. Conceptually, this approach is a stratified cause-effect analysis trying to quantify change-effects of major maintenance scenarios to specific design alternatives of a general problem. In subsection 4.3.4, the SMC metrics for the scenarios of new element and new operation are presented. A more complete logical model for the CVP design alternative, including the alternate maintenance scenarios of modifying and deleting existing elements and operations is presented in

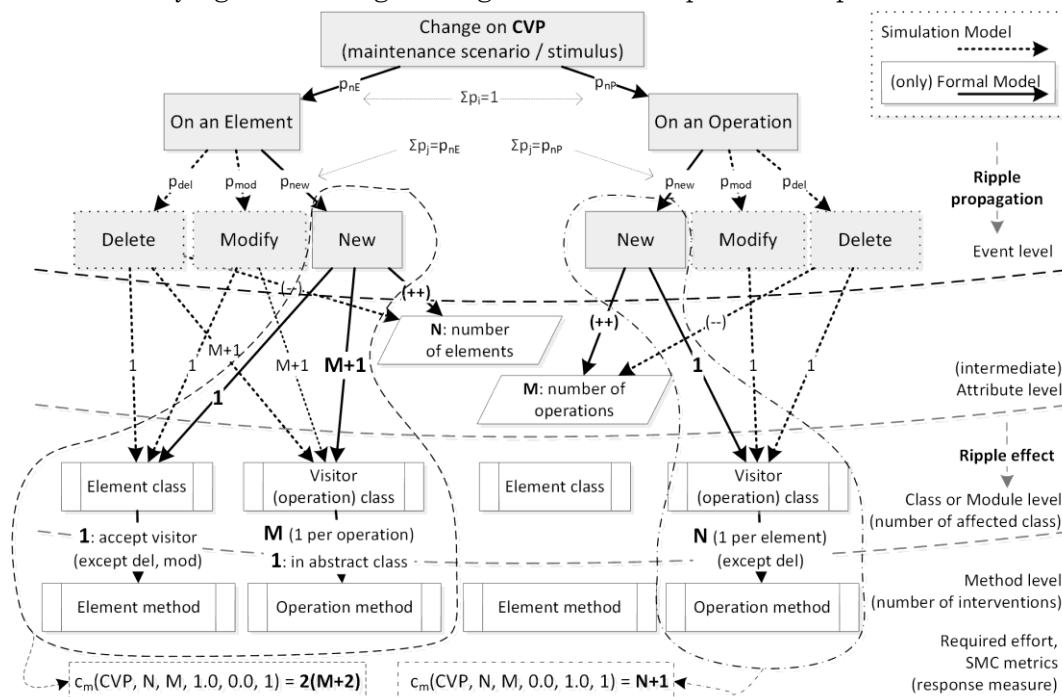


Figure 6-3. Notice that for ‘deleting’ scenarios the corresponding design attributes are decreased accordingly. Respectively, for ‘modifying’ or ‘editing’ scenarios the corresponding design attributes are left unchanged.

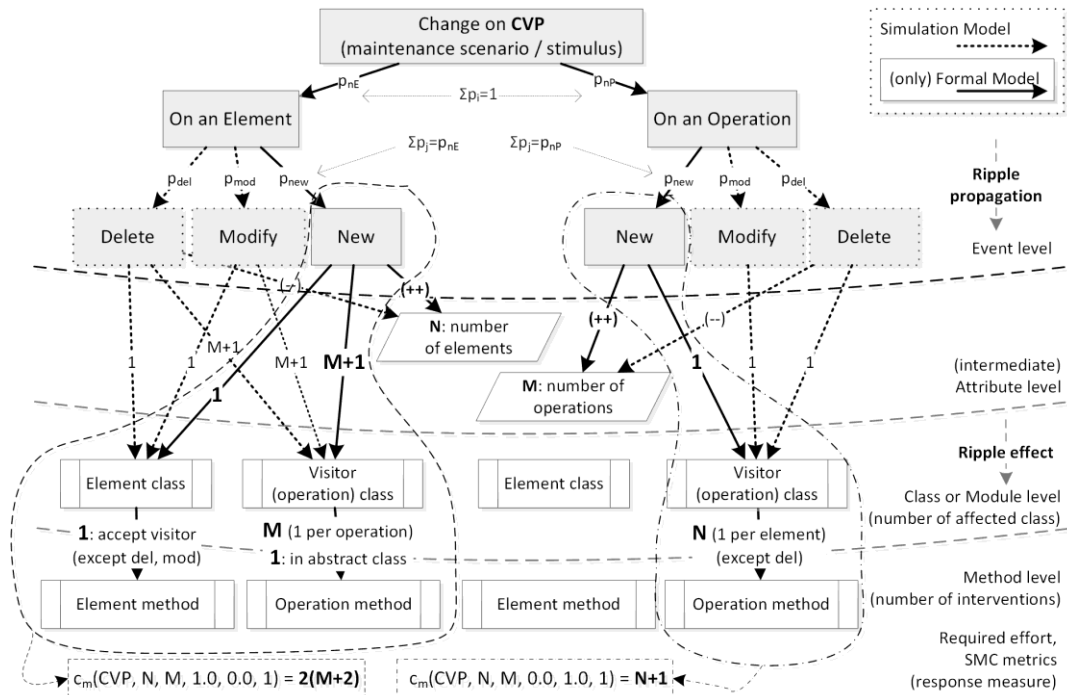


Figure 6-3: Consequence flow (logical model) during impact analysis for changes on CVP design combination, including alternate scenarios.

Respectively, a more complete logical model for the CIBI design alternative, including the alternate maintenance scenarios of modifying and deleting existing elements and operations, is presented in Figure 6-4.

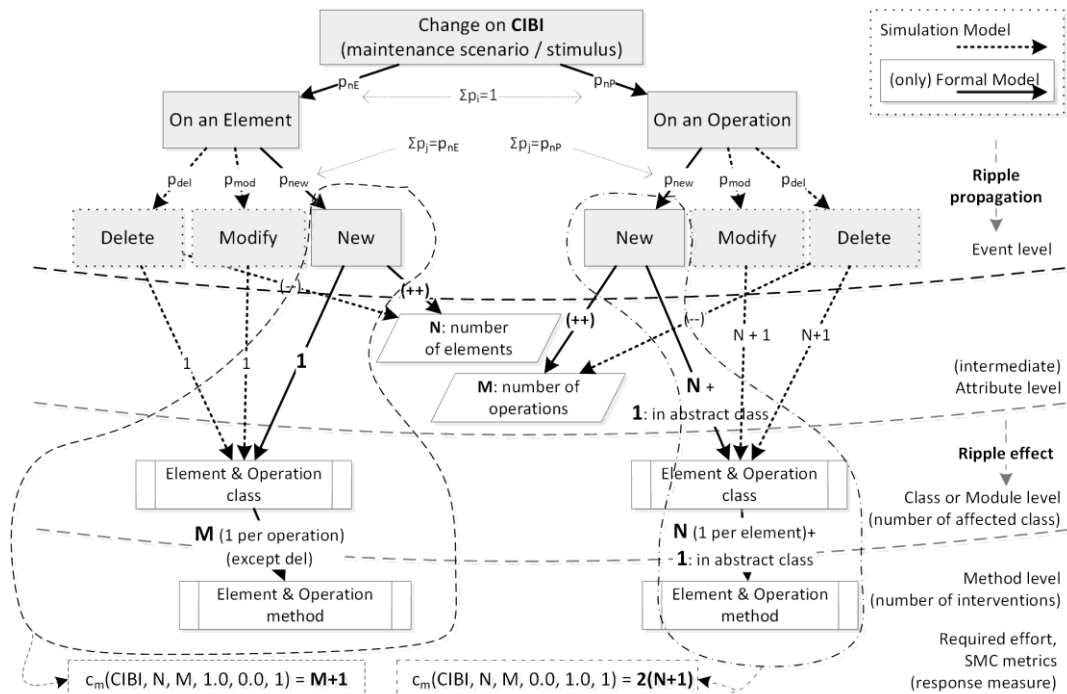


Figure 6-4: Consequence flow (logical model) during impact analysis for changes on CIBI design combination, including alternate scenarios.

In general, fewer method interventions or less affected classes, for a specific maintenance scenario, imply higher cohesion, and lower code entropy, dispersion,

complexity, coupling, and crosscutting degree. Thus, developers will have decreased work keeping track of changes that are performed across fewer files or any other segment, entity, or code unit (Hassan, 2009). SMC metric expands its measurement capacity by simultaneously counting different types of affected code segments. More specifically, SMC metric expresses both, the number of interventions (e.g., affected methods), and the locality or scattering degree of these interventions as well (e.g., expressed by the number of affected classes). Thus, SMC metric provides an adequate graduation of effort assessments even in the absence of source code, as suggested in chapters 3 and 4.

To demonstrate how SMC metric captures the effect of maintenance scenarios as they applied on each design alternative of CIBIvsCVP general problem, a typical code example is presented in Figure 6-5 according to the analysis in chapters 3 and 4. The design attributes N and M represent the current number of the logical entities (i.e., Elements and Operations) represented by the used design patterns in the form of classes and methods. For example, for an Interpreter implementation, the attribute N may represent the number of distinct types of the parse-tree nodes derived from a custom BNF grammar (e.g., terminal – nonterminal symbols, identifiers, etc.) while the attribute M may represent the number of distinct types of the operations over nodes (e.g., type checking, code generation, execution, etc.). Assume that during maintenance there is a need for adding a new element type (e.g., a parse-tree node) to satisfy user requirements. This task requires several interventions or maintenance activities to be made by the developer into different code entities depending on the used design patterns. Initially the corresponding design attribute is updated (i.e.,  $N++$ ). Referring to CIBI design combination in Figure 6-5, M method interventions concentrated in a single class should be made, thus totally  $c_m(\text{CIBI}, N, M, 1, 0, 1) = 1 + M$  class and method interventions, as reported in Table 6-2. Respectively, referring to CVP design combination, M method interventions widespread through M different classes, plus 2 separate method interventions each in a single class should be made, thus totally  $c_m(\text{CVP}, N, M, 1, 0, 1) = 2(M + 2)$  class and method interventions. In principle, a maintenance scenario is considered as major when its effect among design alternatives is significantly different (i.e.,  $1 + M \neq 2(M + 2)$ ). For example, a minor scenario related to a particular operation for a particular element would affect the same number of entities or code units (e.g., a single method) in both design alternatives, thus it would be neutral concerning the comparison and decision-making reliability. Respectively, in the event of adding a new operation (i.e.,  $M++$ ), proper SMC metrics can be derived as illustrated in Figure 6-5. This practical example gives insight on how SMC metric is derived for a given general problem. Keep in mind that as the maintenance scenarios are continually applied, the design attributes are affected and gradually shifted. Thus, for each scenario application, the SMC metric is computed based on different values of design attributes.

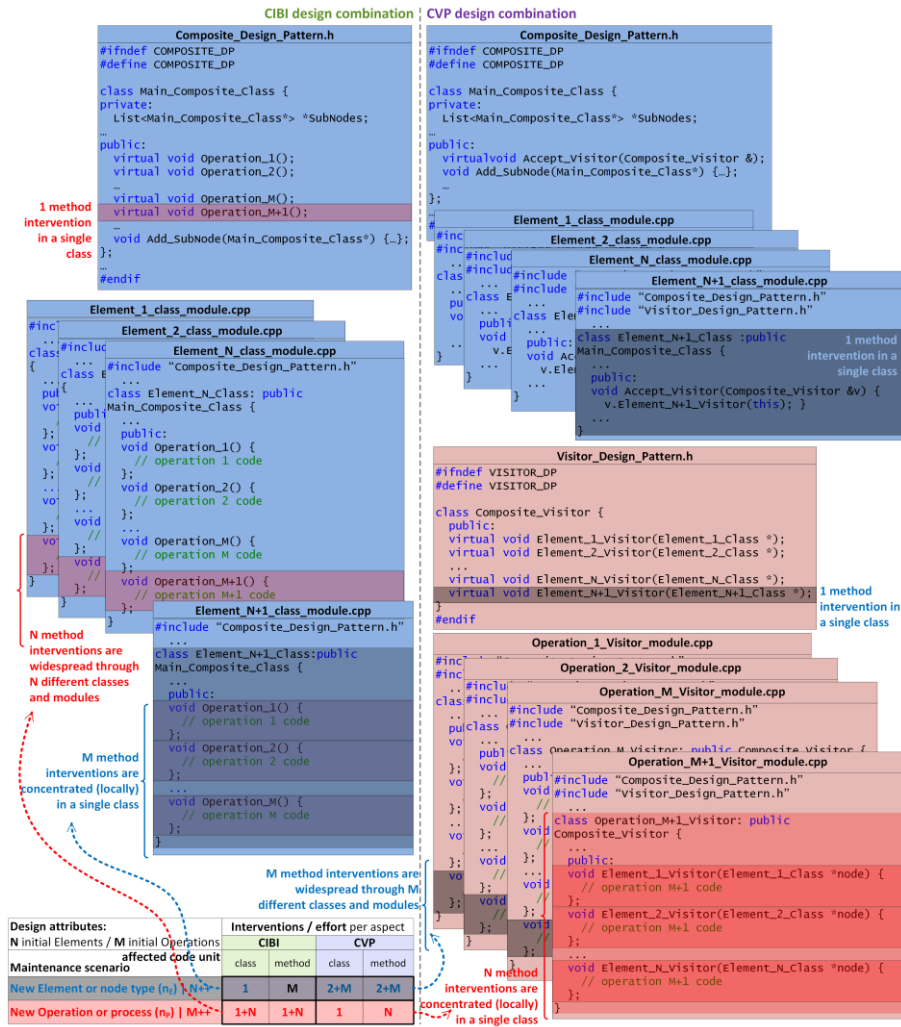


Figure 6-5: Typical code example after the application of one new element and one new operation scenarios for CIBI and CVP design alternatives

Overall, there several theoretical and empirical evidences about the actual relation of SMC metric with properties like code size, scattering degree, etc., and thus to required effort or maintainability degree, as supported in (Aloysius & Arockiam, 2013; Aversano et al., 2009; Canfora et al., 2010; Hassan, 2009; Heitlager et al., 2007b; Karanikolas et al., 2017; Ostberg & Wagner, 2014; Riaz et al., 2009a). In addition, the SMC metric is in accordance with other similar metrics such as Evolution Complexity (Tom Mens & Eden, 2005) and Computational Complexity (Hills et al., 2011), as shown in chapters 3 and 4.

### 6.2.3 Software Expansion Concept and Formal Models Derivation

While SMC metric provides effort assessments for a single applied maintenance scenario ( $\lambda=1$ ), the modeling method introduced in chapters 3 and 4 derives formal models for each general problem and design alternative that provide (total) effort assessments for any number and type of applied scenarios ( $\lambda \in [0, \dots, +\infty)$ ). This theory is based on the expansion trend of software size over their lifetime, since it must be continually adapted to maintain user satisfaction, as suggested in (Meir M. Lehman et al., 1997) and supported by empirical evidences from large repository of historical data (Bakota et al., 2012; Barry et al., 2007; C. R. Cook & Roesch, 1994; H. Gall et al., 1997; Jazayeri, 2002; M. M. Lehman et al., 1998; Yuen, 1988). The underlying concept is that the required maintenance effort is proportional to the size of the code under adjustment as supported in (Araújo et al., 2012;

Bengtsson & Bosch, 1999; Bosch & Bengtsson, 2001; Dolado, 2001; Hayes et al., 2004; Hayes & Zhao, 2005; Jabangwe et al., 2015; Jazayeri, 2002; Zhang, 2008). Furthermore, size properties are identified as the most applicable predictors of effort as concluded in (L. C. Briand et al., 2002). Thus, software maintainability can be expressed through the estimation of the required effort during the maintenance process, as concluded in (Riaz et al., 2009a) and suggested in (Heitlager et al., 2007b). The outcome of the derived formal models for each design alternative (i.e., CVP and CIBI) depends on a) the initial design attributes (i.e.,  $N$  and  $M$ ), b) the scenarios' probabilities (i.e.,  $p_{nE}$  and  $p_{nP}=1-p_{nE}$ ), c) the number of the applied maintenance scenarios ( $\lambda$ ) and it is referred as  $c_m(\text{CVP/CIBI}, N, M, p_{nE}, 1-p_{nE}, \lambda)$  in Table 6-1.

Referring to the general problem of CVP vs CIBI, the derived formal models for CVP and CIBI design alternatives are expressed by the equations (6-1) and (6-2) respectively, as analyzed in chapters 3 and 4. The derivation process is based on a strictly mathematical and quantitative analysis through which the fundamental SMC metrics in Table 6-2 are used to gradually compute the total progressive effort per design alternative in respect to all problem's parameters (i.e., design attributes, scenarios' probabilities, and number of applied scenarios). The method integrates the continually affected design attributes and effort levels through repeated cycles of applied maintenance scenarios (based on their probabilities) in a single function per design alternative.

$$c_m(N, M, p_{nE}, p_{nP}, \lambda)_{CVP} = \frac{3}{2} \lambda^2 p_{nE} p_{nP} + \lambda p_{nP} N + 2 \lambda p_{nE} M + \lambda p_{nP} \quad (6-1)$$

$$c_m(N, M, p_{nE}, p_{nP}, \lambda)_{CIBI} = \frac{3}{2} \lambda^2 p_{nE} p_{nP} + 2 \lambda p_{nP} N + \lambda p_{nE} M + \lambda p_{nE} \quad (6-2)$$

The equations (6-1) and (6-2) are in accordance with empirical validation evidence in (Barry et al., 2007), and the entropy-based prediction model in (Bakota et al., 2012). The selection of the most beneficial design alternative in terms of maintainability is formally stated as the  $\min \{c_m(d, N, M, p_{nE}, p_{nP}, \lambda)\}$ ,  $\forall d \in \{CVP, CIBI\}$ . The difference of the total required effort estimation for CVP and CIBI is given by the equation (6-3).

$$c_m(N, M, p_{nE}, p_{nP}, \lambda)_{CVP} - c_m(N, M, p_{nE}, p_{nP}, \lambda)_{CIBI} = \lambda(p_{nE} M - p_{nP} N + p_{nP} - p_{nE}) \quad (6-3)$$

#### 6.2.4 Formal Models Application in Specific Instances of the General Problem

After the formal models have been derived, it can be easily used to repeatedly support decision-making for any attribute set of a specific instance of the general problem domain. As an application example, the derived formal models (equations (6-1) and (6-2)) are applied to the practical instance of an Interpreter implementation where  $N=40$ ,  $M=10$ ,  $p_{nE}=p_{nP}=0.5$ , and  $\lambda=[1, \dots, 200]$  as discussed in chapters 3 and 4 and subsection 6.2.2. Concentrating on formal outcomes, the diagram in Figure 6-6 indicates that CVP design alternative is preferred, since it requires approximately 15% less effort during maintenance than CIBI.

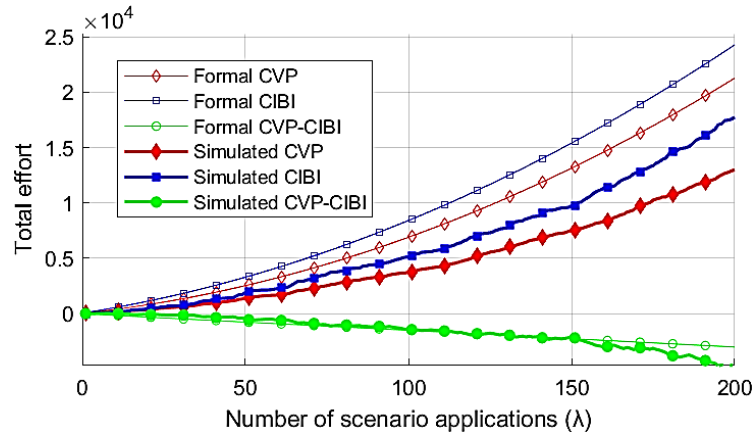


Figure 6-6: Diagram of formal & simulated comparison models applied on the practical example of Interpreter ( $N:40, M:10, p_{nE}:0.5, e_{xp}:1$ ) as an instance of CVP vs. CIBI general problem

Figure 6-6 also presents efforts estimations based on simulated results which discussed in next subsections.

### 6.3 Method Evaluation and Validity Concerns

The modeling method and the derived formal modes have been evaluated on the general and significant problem of CVP vs CIBI in chapters 3 and 4. Although extensive argumentation and validation evidence are provided in chapters 3 and 4, there are still several critical issues under research.

#### 6.3.1 Effort Measurement Validity Concerns

Referring to the used measurement approach, the SMC metric ignores the actual size (lines of code) of each elementary method intervention since the code is not available in the design stage. It is assumed that in a long-term perspective, actual (or business logic) code would be common for both design alternatives under comparison, and thus it has no significant impact on the final effort assessment, hence neutral concerning the decision-making.

#### 6.3.2 Modeling Method Validity Concerns

Referring to the modeling approach, the introduced modeling method concentrates the analysis only in expansion maintenance scenarios (i.e., adding new element or operation) as reported in Table 6-1 and Table 6-2. It is assumed that addition scenarios have a dominant impact on system evolution and maintenance, invoking the innate expansion trend of software to incorporate additional functionality, and arguing that 66% of changes enhancing an existing feature do so by adding a new feature as reported in (Paixao et al., 2017). However, focusing only on addition scenarios seems to deviate from actual (real-world) maintenance circumstances. Moreover, the introduced method ignores code aging issues (David Lorge Parnas, 1994) and developers' aspect, such as experience and learning rate. As a result, the derived formal models are deterministic, despite the heavy uncertain and stochastic nature of actual (real-world) maintenance process.

Under these circumstances, there is no easy way to determine the decision-making reliability of the derived formal models, or their error rate referred to possible incorrect decisions. The latest concerns, as possible threats to validity, are further analyzed and tested in this study through several statistical techniques in the context of an attentive experimentation process.



## 6.4 Method Validation Through Simulation

In this subsection, an experiment is presented to validate the formal models of CIVI vs. CVP general problem under several statistical techniques. The experiment has been designed and conducted according to guidelines provided in (Wohlin et al., 2012). In addition, the guidelines for reporting dynamic simulation studies in software engineering as proposed in (de Fraça & Travassos, 2016) are followed.

Computer simulations are recommended as a more proper (knowledge-seeking (Stol et al., 2016)) research strategy that aligns with the validation goals of this study, approaching better realism of context than formal theories at the cost of lower generalizability. More specifically, the use of simulation models provides several favorable conditions such as limiting bias (mostly human) factors, ensuring common comparison terms, and wide amount of homogenous validation data. Furthermore, the artificial nature of simulation models offers sufficient control over several stochastic factors toward a better understanding of possible causal relationships as recommended in (Hannay & Jørgensen, 2008).

### 6.4.1 Scoping and Planning

The scoping of the conducted experiment is determined by stating its goal framework according to the template proposed in (V. R. Basili & Rombach, 1988) as follows:

- Analyze the formal models of the CIVI vs. CVP general design problem,
- for the purpose of comparison and evaluation (statistical validation)
- with respect to their reliability (reduced estimating error) in supporting correct selection of design alternatives (lesser maintenance effort),
- from the point of view of software engineers,
- in the context of an in silico simulated maintenance process that replicates the stochastic nature and underlying activities of actual maintenance process by evaluating effort assessments of several randomly generated maintenance scenarios for each design alternative of the problem.

### 6.4.2 Hypothesis Formulation

The hypothesis is appropriately stated, in the pursuing of statistical evidence that the two models return almost similar effort assessments for any possible set of their independent variables. That implies paired T-tests, meaning that both treatments (formal and simulation models) are applied in each experiment scenario.

- Null Hypothesis ( $H_0$ ):  $\mu_d=0$ , where  $d = c_m - sc_m$ , or the mean ( $\mu_d$ ) of the differences ( $d = c_m - sc_m$ ) between formal model ( $c_m$ ) and simulation model ( $sc_m$ ) effort assessments for each experiment scenario is not significantly different from zero
- Alternative Hypothesis ( $H_1$ ):  $\mu_d \neq 0$  the mean of the differences between models' effort assessments is significantly different from zero

Although usually it is perfected to refute the null hypothesis, in this case, reliable formal model's assessments are indicated by not rejecting the null hypothesis. According to statistical theory in (Berenson, Levine, & Timothy, 2012), when the null hypothesis is not rejected, it is not implied that it is accepted, but mostly that it is still believable based on the available sampling data. Therefore, proper selection of test's confidence level and sample size to maximize the power of the test to detect that  $H_0$  is false, is required.

### 6.4.3 Variables and Treatments Selection

The formal models (equations (6-1) and (6-2)) and the simulation model are the treatments of the experiment of which the outcomes or dependent variables are under statistical assessment. All the engaged variables and their characteristics are classified per treatment and type in Table 6-3. More specifically, the formal model is affected by five

independent variables returning deterministic assessments through its dependent variable  $c_m()$ . The simulation model is affected by ten independent variables plus four dummy (switch) variables that define the engagement of several stochastic factors, returning fluctuating assessments through the  $s_{cm}()$  dependent variable. Hence, the simulation model introduces random behavior expressed through the following stochastic variables: uncertainty factor effecting scenarios' probabilities ( $f_{BM}$ ,  $u_F$ ), alternate maintenance scenario application ( $a_{lt}$ ), maintenance scenarios actual size ( $s_{izing}$ ,  $s_{size}$ ), developers experience level ( $e_{xp}$ ), developers' learning rate type ( $r$ ), and code aging issues ( $a_{ging}$ ,  $a_{ge}$ ). The intelligible experiment goal is to explore whether the derived formal models and related modeling method introduced in chapters 3 and 4 take into consideration all these random factors in an indirect manner.

Table 6-3: Experiment Variables per Treatment for Formal and Simulation Models on CIBI vs. CVP General Problem

Description or attribute or behavior or outcome	C <sup>1</sup>	Notation	Scale	Range	Type	Options	Distribution (during simulation)	F S M M
Object: System or Design combination under maintenance								
Design combination	R	D	Nominal	{CVP, CIBI}	Indep	Constant		x x
Initial Elements	S	N	Ratio	[1, ..., +∞)	Indep	Initial	Variation	x x
Initial Operations	S	M	Ratio	[1, ..., +∞)	Indep	Initial	Variation	x x
New element probability	S	$p_{nE}$	Ratio	[0.0, ..., 1.0]	Indep	Constant		x x
Uncertainty Factor of Brownian Motion	R	$f_{BM}$	Ratio	[0.0, ..., 0.5, ..., 1.0]	Indep	Constant		x
Overall Uncertainty Factor effecting probability $p_{nE}$	I	$u_F$	Ratio	( $-f_{BM} \times 0.3$ , ..., $f_{BM} \times 0.3$ )	Indep	Random	$f_{BM} \times N(0, \sqrt{\lambda}, 0, 3) / (10 \times \sqrt{\lambda})$	x
Number of scenarios	R	$\lambda$	Ratio	[1, ..., +∞)	Indep	Increased	Linearly	x x
Scenarios' actual Size	I	$s_{size}$	Ratio	[1.0] (0.01, ..., 2.0)	Indep	Constant Random	Fixed=1.0 $N(1, 0.33, 0, 3)$	x
Scenarios' actual Size type	R	$s_{izing}$	Nominal	[Constant, Random]	Switch	Constant		x
Alternate scenarios type	R	$a_{lt}$	Nominal	[Only expansion, All]	Switch	Constant		x
Code Aging or expansion or entropy type	R	$a_{ging}$	Nominal	[Constant, Increased]	Switch	Constant		x
Age factor	I	$a_{ge}$	Interval	[1.0] [1.0, ..., 2.0]	Indep	Constant Increased	Fixed=1.0 Linearly	x
Theoretical Subject (perceived as object): Developer(s) / Company								
Experience level or comprehension degree or quality of resources	I,S	$e_{xp}$	Interval	[1.0] [0.1, ..., 2.0] (0.1, ..., 2.0)	Indep	Constant Sample Random	Fixed=1.0 Sample value ( $e_{xp}$ ) $N(1.5, 0.33, -0.5, 3) + 1$	x
Learning Rate type	R	$r$	Nominal	[Constant, Sample, Random]	Switch	Constant		x
Total progressive outcome: size of affected code's entities, or maintenance effort prediction / assessment								
Formal Model	-	$c_m()$	Ratio	[0, ..., +∞)	Depend		Deterministic	x
Simulation Model	I	$s_{cm}()$	Ratio	[0, ..., +∞)	Depend		Variation	x

<sup>1</sup> Controlled by R: researcher in the lab, S: sampling-random selection, I: internally by Simulation Model according to switches' state.

Normal distributions are referred to +/- 3σ limits.

Distribution notation is referred as  $N(\text{mean}, \text{std deviation}, \text{skewness}, \text{kurtosis})$  where  $N(0,1,0,3)$  is referred to normal distribution with  $\mu=0, \sigma=1$ .

#### 6.4.4 Selection of Sample (Subjects and Objects)

*Sample selection:* an experiment scenario encloses the object that represents the initial system's attributes, and one quasi-subject that represents developer(s) characteristics as indicated in Figure 6-7. Developers are referred as quasi-objects since their offer in the maintenance process is simulated by the simulation model. Thus, an experiment scenario is defined by totally 14 independent variables. Two of those variables (Design,  $\lambda$ ) are controlled by the researcher, five of those ( $f_{BM}$ ,  $s_{izing}$ ,  $a_{lt}$ ,  $a_{ging}$ ,  $r$ ) are switches also controlled by the researcher, and three of those ( $s_{size}$ ,  $a_{ge}$ ,  $u_F$ ) are internally controlled by the simulation model, as classified in Table 6-3 and Figure 6-7. The rest four independent variables ( $N$ ,  $M$ ,  $p_{nE}$ ,  $e_{xp}$ ) define the design attributes, scenarios' probabilities, and developer(s) experience level of a specific system's instance of the general problem. In particular, the first three variables ( $N$ ,  $M$ ,  $p_{nE}$ ) define an object or system's attributes, and the rest one ( $e_{xp}$ ) defines the theoretical subject or developer(s) characteristics. The possible combinations of these variables define the spectrum or a listing of items that make up the population from which the sample of experiment scenarios is selected. Notice that  $p_{nE}$  indirectly defines  $p_{nP}=1-p_{nE}$ . In addition, occasionally,  $e_{xp}$  variable is also controlled internally by the simulation model, as indicated in Figure 6-7.

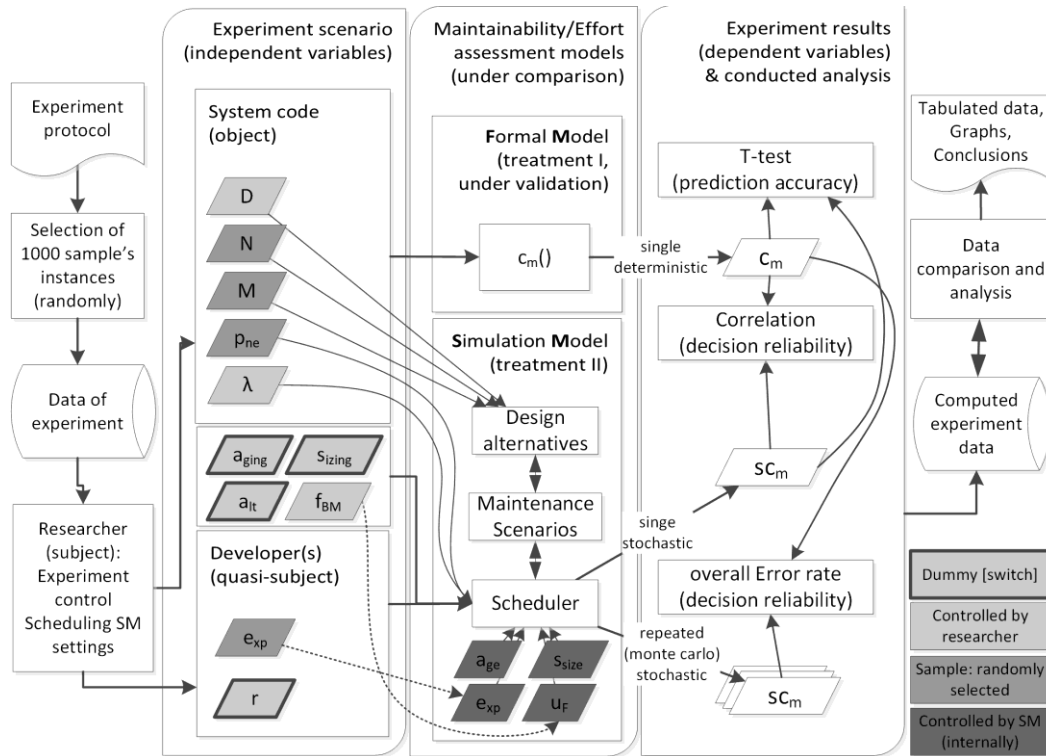


Figure 6-7: Experimental setup visualization

The sample of system's properties and developer characteristics, represented by  $N$ ,  $M$ ,  $p_{nE}$ ,  $e_{xp}$  variables, is selected through absolute random sampling. In order the sample be representative of actual object-subjects, the ranges of the independent variables are limited into  $N=[20, \dots, 200]$ ,  $M=[5, \dots, 150]$ ,  $p_{nE}=[0.05, \dots, 0.95]$ , and  $e_{xp}=[0.2, \dots, 2.0]$ . Given that the range of each variable covers the majority of any possible (actual) object-subject, the randomly generated sample adequately represents the whole population. The entire set of the 1000 sample's instances is provided in Appendix B.

*Confidence level and sample's size determination:* several preliminary trials paired t-tests on predictions of simulation models, and formal models' computations for various parameters showed that there is no need for increasing the confidence level ( $1-\alpha=0.95$ ). Furthermore, considering the limitations related to required process-time and the need for a small sampling error less than 1.3% of the maximum effort's range, the optimal sample size is selected to  $n=1000$ . Setting sampling error less than 1.3% of the maximum effort's range permits the detection of very small differences, thus reducing  $\beta$  risk of Type-II errors and increasing the power ( $1-\beta$ ) of a statistical test to detect that  $H_0$  is false.

## 6.4.5 Conceptual Analysis of Validation Process

### 6.4.5.1 Selected Research Strategy

In this subsection, the selected research strategy properly adapted to the context of this study is conceptually presented and discussed. Scientifically speaking, researchers try to predict the physical or general systems' behavior through theories, prediction models, and methods that captures the cause effect relationships among independent and depended variables or factors of interest. Thus, the reality aspect be approached by a theoretical aspect. Two basic (modeling) approaches prevail on this try as illustrated in Figure 6-8.

*Formal/analytical/empirical study (path A):* In this case, a system or a phenomenon is observed and some possible or intuitive concepts regarding system's behavior are highlighted. Based on this theoretical concepts, a (prediction) model is proposed, usually through a special modeling methodology or theory. Next, several (mostly sampling) data -

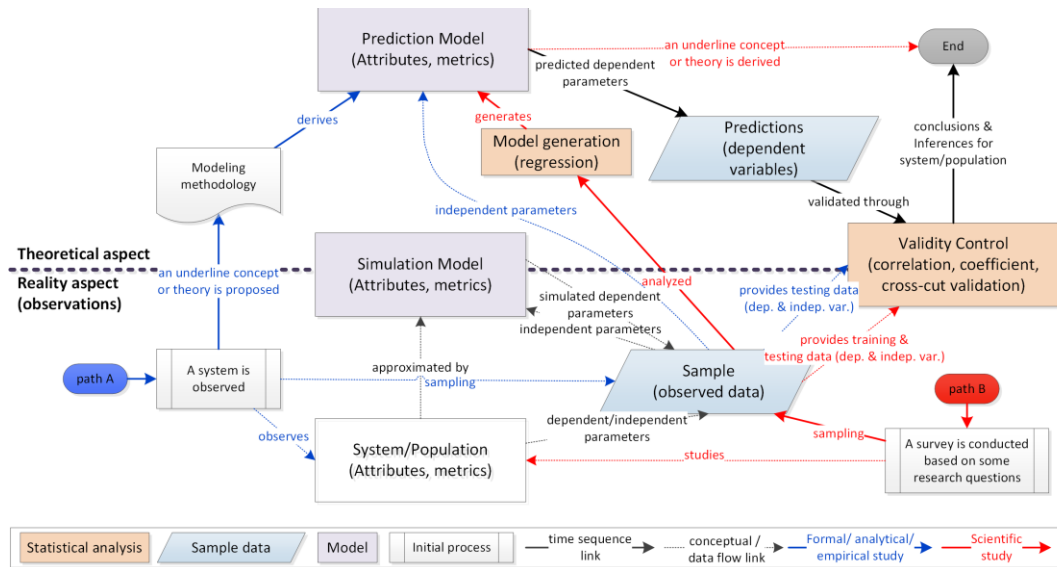


Figure 6-8: Visualization of research strategy, focusing on reality and theoretical aspects.

observations about the system's behavior are collected through surveys, case studies, experiments, simulations, sampling processes, etc. These data are usually classified in independent and dependent variables where the latest are those that can predicted by the proposed theory and prediction model. Finally, the observed and predicted data are compared each other under several statistical methods in order the prediction model be validated. Such validation processes could be t-test, z-test, correlation coefficient, etc. The successful validation of the proposed model by empirical observations provides sufficient statistical evidence that the relevant theory or methodology adequately describes, represents, or reflects the actual system or the phenomenon under study.

*Scientific study (path B):* In this case, several (mostly sampling) data - observations about the system's behavior are directly collected through surveys, case studies, experiments, simulations, sampling processes, etc. Usually, the data collection and selection process are guided by some fundamental research questions about the system or the phenomenon under study, however, no particular theory or model that describes the behavior or cause-effect relationships of the phenomenon is yet available. Next, all the observed data are analyzed under several statistical techniques such as regression analysis or by using computer aided techniques such as neural networks and machine learning algorithms. Through the previous analysis, prediction models are generated regarding the prediction of the dependent variables based on the (significant) independent variables. Finally, the observed and predicted data are compared each other under several statistical methods in order the prediction model be validated. Such validation process could be the ANOVA test, determination coefficient, cross-cut validation, etc. In addition, an underlying concept, or a theory about the behavior or cause-effect relationships of the system or the phenomenon under study can be derived through the interpretation of the prediction model and its parameters.

Which of the two approaches are the most proper for a given problem (system or phenomenon) under study is a difficult question and the answer depends on the specific features of each problem. In general, the derivation of a prediction model through statistical techniques (path B) requires a significant amount (sample size) and range of historical or survey data. Moreover, in the case of software maintainability assessment, there also some other obvious concerns and limitations. For example, the conducted surveys, case studies and sampling process during software maintenance are extremely costly in time and resources and thus the number of the observations are very limited per study,

heterogeneous, usually unclassified, and unevenly conducted through literature. In addition, the historical data in literature, regarding various quantitative measurements and evidence during maintenance process, are significantly different each other and unequally distributed and thus, are not comparable and statistically meaningful. Furthermore, the sampling process during software maintenance is heavily affected by human activities and many other stochastic and bias factors, arising out of heterogeneity among different developers' teams, programming environments-tools, and system's types. Thus, it is rather difficult if not impossible, a decent set of observations (sample) be extracted for statistical analysis and validation purposes, especially in the field of software maintenance. Although, simulation models running in software could be an alternative toward this direction, a modeling methodology (through path A) that could provide formal and valid prediction models without the need of statistical analysis and validation process, would be very helpful. Especially for the case of comparison of design alternatives based on their maintainability perspective where homogenous observations are not available. However, a formal model without a strictly statistical validation against real-world observations may be subject to several accuracy or reliability issues.

The proposed theory, modeling method, and derived formal models under statistical validation through simulations are in accordance with the Formal/analytical/empirical study as represented by the path A in Figure 6-8. More specifically, given that software maintenance process can be approached from the perspective of software evolution, the engaged design patterns per design alternative provide an insight regarding the followed evolution pattern of the system for major maintenance scenarios (classes of resembling activities). Furthermore, the design attributes of the logical entities of the design problem under study as they represented by code entities of the engaged design patterns (per design alternative) gives an extra insight regarding the cause-effect relationships and underlying evolution theory of the addressed design problem. The introduced modeling theory, the SMC metric, and the derived formal comparison models (chapters 3 and 4) describes the underlying concept of software evolution during maintenance based on the architectural analysis of the engaged design patterns. Due to the absence of adequate volumes of homogeneous observations, a multi-variable simulation model that replicates the underlying activities and variability of actual maintenance process providing homogenous observations is introduced in this chapter. Finally, the simulated observations are statistically compared with the outcome of formal modes to evaluate the reliability of initially proposed modeling theory and method.

#### **6.4.5.2 Theoretical and Observational Aspects**

In this subsection, the conceptual analysis of the validation process focusing on the contradistinction among the theoretical and observational aspects is discussed. This study introduces a (high level) theory regarding the early comparison of design alternatives based on their maintainability perspective. More specifically, a modeling method that generates comparison formal models based on change rate analysis of software design attributes through differential equations is proposed (chapters 3 and 4) as visualized in Figure 6-9. The modeling method uses the fundamental Structural Maintenance Cost (SMC) metric, which captures the expansion behavior of the design combinations. The introduced theory suggests that the modeling method describes the progressive software evolution during the maintenance process. Hence, the generated formal models predict the required effort during the maintenance process expressed in terms of numbers of method and class interventions as measured by the SMC metric. Thus, the predicted maintenance effort guides the selection among implementation alternatives based on their maintainability perspective since less maintenance effort corresponds to a better maintainability degree.

The generated formal models predict the required effort based on a small set of independent variables such as design attributes and scenario probabilities. Although persuasive argumentation about the modeling method and formal models' logic is

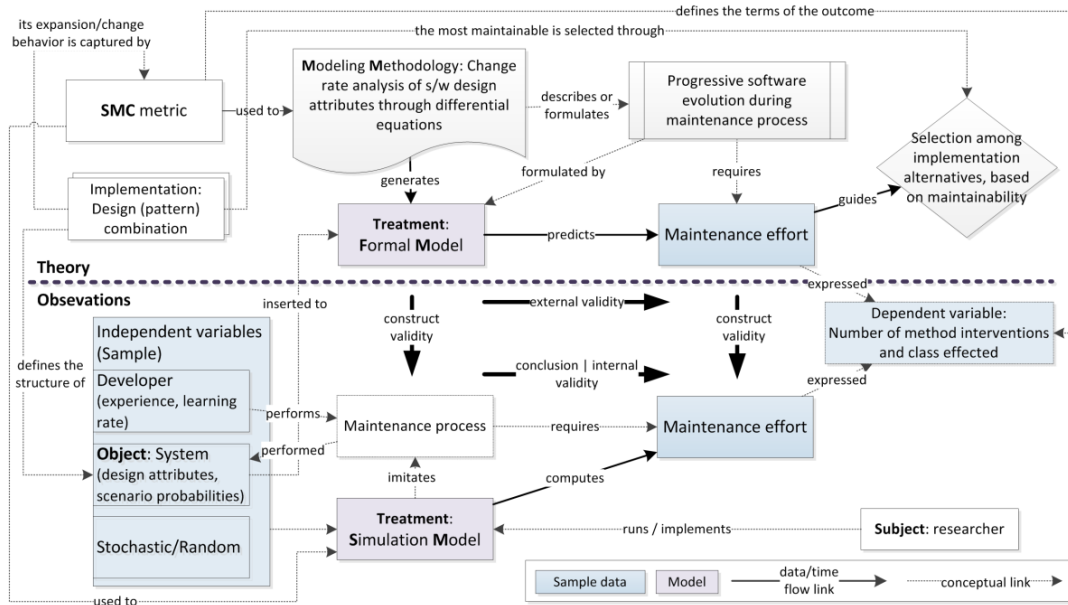


Figure 6-9: Visualization of the experiment context, focusing on contradistinction among theoretical and observational aspects

provided, the effect of the uncovered or ignored random and stochastic factors in effort predictions is a significant concern that should be further investigated. Thus, the statistical validation of the formal models' prediction ability is required through observations and measurements about the maintenance process of actual systems.

The statistical validation of the formal models' effort prediction ability provides sufficient evidence (or strong indication) that i) the introduced theory is also valid and, thus the proposed modeling method adequately describes the progressive software evolution during the maintenance process, and ii) the predicted effort is a reliable (not necessarily accurate) magnitude toward the comparison and selection among design alternatives based on their maintainability perspective.

In the absence of adequate sampling data, a simulation model that imitates the maintenance process of a system while taking into consideration many other random and stochastic factors as independent variables is proposed. The simulation model uses the SMC metric and computes the maintenance effort for a large number of systems or problem's instances (sample). Hence, the simulation model computes the required effort during the maintenance process, also expressed in terms of numbers of method and class interventions as measured by the SMC metric. Notice that the simulation model encapsulates random and stochastic behavior, and thus it returns different computations for the same values of its independent variables.

Because simulation in computers is an automated activity that takes place in a controlled environment, the validation process in the current study is documented and conducted under the sight of experimentation in software engineering, as described in (Wohlin et al., 2012). Thus, theoretically, the developers (subjects) perform maintenance (treatment) over a sample of problem's instances or systems (objects), which are defined or distinguished by attributes (independent variables), requiring a specific amount of effort (dependent variable). However, the maintenance process is imitated by the simulation model. Hence, in practice, the researcher (subject) performs simulations through the simulation model (treatment) over a sample of problem's instances or systems (objects), which are defined or distinguished by attributes (independent variables), requiring a specific amount of effort (dependent variable).

The sample of the possible practical systems is selected randomly with a constant distribution considering meaningful intervals for all independent variables. Dummy variables (categorical / switches) can be selected systematically to explore the contribution of each stochastic variable to the simulation model and its outcome. Considering the formal model as an alternative treatment, the validation process targets to the statistical comparison between formal models' effort predictions and simulation model's effort computations, for the selected sample of systems and various switch sets and iterations. More specifically, the statistical validation targets to the inference that the formal models' predictions are not significantly different from the simulation models' computations or that the average of the differences between paired formal model and simulation model predictions is not significantly different from zero. That inference generalizes the observation evidence from the sample scope to population scope. In other words, this outcome provides statistical evidence about i) the formal model's effort prediction ability for any actual possible system, and ii) the modeling method's formulation ability to (statistically) describe the progressive software evolution during software maintenance. However, due to the lack of as strictly validation of simulation model against real-world observations, this inference may be of low importance as discussed in chapter 1 and subsection 6.1.

Finally, because the selection among design alternatives is based on the minimum required effort, the statistical comparison between formal model's and simulation model's effort predictions is conducted based on the differences of effort predictions for any implementation alternative under comparison. Also, because the sign of the difference between the predictions of design alternatives indicates an opposite selection (incorrect decision or error), an additional error rate assessment is conducted regarding the reliability of the decisions. The introduced modeling method and the derived formal models (chapters 3 and 4) must support the decision/selection of the most beneficial design combination among alternatives based on their maintainability perspective. The decision is based on fundamental design (pattern) attributes as well as on probability assessments about primary maintenance scenarios. Hence, the decision is (by default) a probabilistic assessment, and thus, its accuracy should be determined by a certain confidence level. The pre-mentioned error rate assessment returns an accurate confidence level about formal models' ability not only to provide valid effort assessments, but correct design decisions as well. After all, proper selection or limited decision-risk among design alternatives is the main goal of the suggested theory in chapters 3 and 4.

Conclusively, the context of the described experiment is an off-line project, which is conducted by the researcher through computer-aided simulations (treatment). This treatment incorporates professionals' (developers-subjects) behavior and individual system's design attributes (objects) as stochastic and random independent variables, for general purposes, covering almost all the range of possible values of design attributes (systems-objects). Thus, the generalization of the conclusions about a specific formal model and design alternatives is valid for any reasonable set of the problem's instances (objects of the population) as distinguished by their design attributes.

#### 6.4.6 Experiment Design

The overall experimental setup is visualized in Figure 6-7. In brief, both treatments (Formal and Simulation Models) are simultaneously applied for each set of experiment scenario's variables (system's attributes and developers' characteristics), returning effort assessments through their dependent variables ( $c_m$ ,  $sc_m$ ). Furthermore, the tests are massively conducted for several combinations of the dummy (switches) variables that control the stochastic behavior of the simulation model. This allows the further exploration of the contribution of each stochastic variable to the validation process. Overall, the experiment results are arranged at seven distinct simulation states in which each stochastic variable is gradually engaged, as presented in Table 6-4. Each state returns data for both

design combinations, different  $\lambda$  values, and all instances of the random object-subject sample. Lastly, the modular analysis of the experimental data is described next.

*Internal convergence:* The convergence control confirms that the simulation model provides targeted computations ( $sc_m$ ) in a limited interval for a specific system (object) given the fact that simulations are internally affected by several random and stochastic factors. This preliminary control is repeatedly conducted up to 100 times for all the selected simulation states in Table 6-4. In principle, this control process approximates the Monte Carlo Simulation (Rubinstein & Kroese, 2016), since it substitutes several variables’ ranges by random values based on specific probability distributions for any factor ( $u_F, s_{size}, e_{xp}$ ) in Table 6-3 that has inherent uncertainty.

Table 6-4: Combinations (States) of Simulation Model’s Independent Variables and Switches

Simulation Model’s Variables Description of gradually engaged stochastic factor	Independent		Switches / dummy independent					Independent (Sample)	
	Design alternative $\lambda$		Uncertainty Factor $f_{BM}$	Alternate scenarios $alt$	Scenario size $S_{izing}$	Code aging $aging$	Developers learning rate $r$	System attributes $N, M, p_{ne}$	Developers (experience) $e_{xp}$
1. Variable scenario sequences	CIBI & CVP	1, ..., 200	Low: 0.0	Only new	Constant	Constant	Constant	Randomly selected sample size: 1000 instances of the general problem (Number of repeated simulations per sample instance: 100)	x 1000 x 100
2. Shifting scenarios probabilities			<b>Mid: 0.5</b>	Only new	Constant	Constant	Constant		
3. Alternate maintenance scenarios			Mid: 0.5	<b>All</b>	Constant	Constant	Constant		
4. Variable interventions’ size			Mid: 0.5	All	<b>Normal</b>	Constant	Constant		
5. Code aging & learning rate			Mid: 0.5	All	Normal	<b>Increased</b>	<b>Sample value</b>		
6. Variable developers’ experience			Mid: 0.5	All	Normal	Increased	<b>Normal Skewed</b>		
7. Highly shifting scenarios probabilities			<b>High: 1.0</b>	All	Normal	Increased	Normal Skewed		
Total outcomes $2.8 \times 10^8$	2 x 200			x 7					

*External correlation:* Referred to the coefficient of correlation between formal model’s effort predictions ( $c_m$ ) and simulation model’s computations ( $sc_m$ ) for all the selected simulation states in Table 6-4. This type of control is conducted based on one time (single) simulation per sample instance since repeated simulations could conveniently manipulate the statistical significance. Thus, any statistical inference is subject to the stochastic nature of the simulation model.

*Hypothesis Testing:* Two-sided, paired t-test among formal model’s ( $c_m$ ) and simulation model’s ( $sc_m$ ) assessments is preferred since the population variation  $\sigma^2$  is typically unknown, as supported in (L. Briand, Emam, & Morasca, 1996; Montgomery, 2012). In general, the parameters involved in a parametric test should be normally distributed. However, for large sample sizes, as in this case, either of the parametric or the nonparametric tests work adequately, and thus the assumption of the t-test is met even for non-normal measurements. Furthermore, several tests showed that parametric methods, such as the t-test, are fairly robust to deviations from the preconditions (interval scale) as long as the deviations are not too large, as discussed in (L. Briand et al., 1996). To address this concern, the quantitative parameters (Formal and Simulate effort assessments) of the tests are investigated under the assumption of normal distribution. Again, this test is conducted based on one time (single) simulation per sample instance as in external correlation test.

*Error rate assessment:* Probabilistic models could have some precision issues for some marginal cases. Thus, decisions based on formal model’s and Simulated results may conclude to opposite selections among design alternatives for the same set of input variables, indicating an incorrect decision or “error” due to formal model’s precision issues and simulation model’s stochastic nature. Hence, the error rate is another sophisticated



measure of the model’s reliability degree, focusing on the decision-risk taken by the designers.

In the context of this study, the average error rate ( $E_r$ ) is computed through up to 100 repeated simulations for different  $\lambda$  values and for all the simulation states in Table 6-4. During each repetition, the intermediate simulated results ( $sc_m$ ) are compared to formal model’s deterministic results ( $c_m$ ) while occurring errors are counted. Repeated executions are suggested in the context of several studies about testing randomized software for simulation purposes (Guderlei, Mayer, Schneckenburger, & Fleischer, 2007). In addition, conducting repeated executions is an approximation of the Monte Carlo Simulation (Rubinstein & Kroese, 2016), a technique that furnishes the decision-maker with a range of possible outcomes and the probabilities they will occur for any possible choice of action. Finally, through further analysis of individual errors, critical errors with high negative impact in terms of wasted effort and high probability to occur are spotted.

#### 6.4.7 Instrumentation

The main instrument of the experiment was a custom simulation model, which has been designed based on several related works as proposed in (Barros, Werner, & Travassos, 2004; Kelsen, 2004; Müller & Pfahl, 2008; Stopford & Counsell, 2008).

*General Description:* The simulation model has been developed in two variations with identical behavior and outcomes. Given the initial formal models and the directions provided in this subsection, the development cost of the simulation model is expected to be moderate with the most efforts being placed on its calibration as discussed in subsection 6.4.9. The first variation was through an object-oriented language where its logical entities such as the scheduler, types of maintenance scenarios, design patterns, and virtual subject (developer) are represented by separate classes, thus approaching an entity or modular model (Zeigler, Mittal, & Traore, 2018). The design combinations under assessment are declared through run-time instances of these classes, like a custom internal Domain Specific Language (DSL). This variation allows the deployment of the components of a general problem through an (typical) object-oriented programming language while provides potentials for adapting the model to different design problems. The second variation was in the form of MATLAB® dynamic functions and scripts which is a more versatile implementation in terms of data manipulation and graph generation. However, this functional model requires a strictly mathematical background and programming style. In both variations, the total effort for each design alternative is computed through repeated executions (or applications) of several maintenance scenarios based on their individual probabilities. It is important that the proposed simulation model generates scenario sequences and streams of (stochastic) factors’ values which are simultaneously used in all design alternatives. Furthermore, it replicates the same pattern of structural evolution under the same parameters as in the formal modes. Thus, the simulation model adapts its behavior according to its parameters to provide classified observations with regards to specific design parameters and scenarios probabilities. Thus, these are the perfect comparison conditions that only a controlled simulation can provide.

##### 6.4.7.1 Description of Parameters and Stochastic Factors

*Effort/size measurements:* The simulation model’s scheduler is responsible for recording the virtually provided effort per scenario application. More specifically, the simulation model computes the maintenance effort for each design combination and maintenance scenario through the linear equation (6-4) which also uses the formal SMC metric ( $c_m$ ) in Table 6-2. As a result, even if several stochastic factors are engaged, the simulation model returns effort assessments ( $sc_m$ ) also expressed in terms of numbers of method and class interventions.

$$sc_m(\text{CIBI/CVP}, N, M, p_{nE}, p_{nP}, 1) = \frac{a_{ge} \cdot s_{size}}{e_{xp}} c_m(\text{CIBI/CVP}, N, M, p_{nE}, p_{nP}, 1) \quad (6-4)$$

Conceptually, the equation (6-4) can be perceived as the disaggregation of the actual required effort in meaningful components or factors. More precisely, the size factor ( $s_{size}$ ) represents the actual action's size, which is proportional to the formal SMC metric since greater actions demand more effort. The experience factor ( $e_{xp}$ ) reflects the developers' efficiency, which is inversely proportional to the required effort since higher experience degree implies less effort. The age factor ( $a_{ge}$ ) can be considered as a friction coefficient that contradicts developers' efficiency, and thus it slows down the maintenance process by requiring more effort. Consequently, the age factor ( $a_{ge}$ ) is proportional to the required effort. The linear relationship between sub-factors such as size and effort is supported by many studies (Araújo et al., 2012; Bengtsson & Bosch, 1999; Bosch & Bengtsson, 2001; Dolado, 2001; Hayes et al., 2004; Hayes & Zhao, 2005), including the area of early Function Point Analysis (Giuliano Antoniol, Lokan, Caldiera, & Fiutem, 1999; B. Boehm et al., 1995; Barry W. Boehm et al., 2000; Caldiera, Antoniol, Fiutem, & Lokan, 1998; Meli, 1997; Musilek, Pedrycz, Nan Sun, & Succi, 2002). The principal component of SMC metric  $c_m()$  makes the simulated measurement of equation (4) representative for a particular software architecture of interest pertaining to a specific evolution scenario. That because SMC metric captures the evolution pattern of a particular scenario per design alternative as affected by the current values of design attributes (described in subsection 6.2.2). Design decisions for a different design problem would have different scenarios, design attributes, and set of SMC metrics per maintenance scenario and design alternative, thus adapting the (simulated) maintenance effort for alternative design decisions as well. Furthermore, the use of SMC metrics as the core of effort measurements by the simulation model is required in order the comparison of two treatments' outcomes be fair.

*Sequence of Maintenance Scenarios:* The scheduler of the simulation model affects the sequence or the order of the applied (arriving) maintenance scenarios. The Modeling Theory under validation considers that the generated scenarios' sequence is unique flowing a steady repetition pattern based on individual scenarios' probabilities. This approach ignores possible random instances of scenarios' sequences, and thus, it has been selected in chapter 3 as an approximation due to its computational formality. In practice, during the maintenance of real-world systems, there are many possible combinations (sequences) of scenarios that can occur for a given number of scenarios' applications ( $\lambda$ ) and specific probabilities. However, the frequency of all these sequences is normally distributed around the most common case, which is the unique repeated sequence adopted by the formal models. More specifically, considering the possible maintenance scenario types as separate events (in this case  $n=2$  events) with probability  $p_1$  and  $1-p_1$ , the frequency distribution of all possible sequences of  $\lambda$  event occurrences is represented by the binomial distribution. Practically, the simulation model generates scenario sequences based on a random generator, which is affected by the individual scenario's probabilities. The stochastic generation of scenarios (set of tasks) is a standard approach in the domain of simulations (e.g., as Events or Requirements (Kelsen, 2004; Stopford & Counsell, 2008)).

*Alternate Maintenance Scenarios:* The scheduler of the simulation model can apply alternate maintenance scenarios through the switch  $a_{it}$ : [Only expansion, All]. In 'Only expansion' mode, only the expansion scenarios are engaged in conformity with formal model and modeling method assumptions. In 'All' mode, the alternate maintenance scenarios for modifications and deletions are engaged in accordance with the additional SMC metric equations provided in Table 6-2.

*Actual Size of Maintenance Scenarios:* The scheduler of the simulation model can influence the size of each maintenance scenario through the switch  $s_{sizing}$ : [Constant, Random]. The scenario's size factor ( $s_{size}$ ) is a value related to the actual size of a single

method intervention, e.g., the real added code for a new method, including its business logic code. In ‘Constant’ mode, the factor  $s_{size}=1$ , thus it does not affect standard effort assessment as assumed by the Modeling Theory under validation. In ‘Random’ mode, the factor  $s_{size}$  receives randomly generated values in the range (0.01, ..., 2.0) based on a normal distribution pattern. Considering the random factor  $s_{size}$  as the average value of all individual methods’ actual sizes for a scenario, it could be safely assumed that  $s_{size}$  values proximally follow a normal distribution pattern regardless of the (real-world) distribution of the methods’ actual sizes (in accordance with the Central Limit Theorem properties). Consequently, the SMC metric  $c_m$ , initially expressed as the number of method interventions per scenario, multiplied by the stochastic factor  $s_{size}$  (in actual size per method units), becomes a measure that statistically expresses the actual expected size of the code affected by a particular maintenance scenario (in actual size per scenario units). Typically, this approach aggregates a Micro stochastic sequence (individual methods’ sizes) into macro behavior using law of large numbers expressed in a simpler stochastic form (average value of all methods’ sizes), as suggested in (Zeigler et al., 2018).

*Code Aging:* The scheduler of the simulation model takes under account software aging issues through the switch  $a_{ging}$ : [Constant, Increased]. In ‘Constant’ mode, the factor  $a_{age}=1$ , thus it does not affect standard effort assessment. In ‘Increased’ mode, the age factor is gradually (linearly) increased in the range of [1.0, ..., 2.0] for each maintenance scenario. In general, software code and its quality tend to be fading mostly due to its increasing size and complexity, outdated technical and dissimilarity issues, long-term compatibility, and comprehension issues, etc. All these concerns about software aging are inevitable as substantiated in (David Lorge Parnas, 1994). In principle, as the system’s code becomes older, more effort required by the developers for adding or modifying a fixed amount of code.

Furthermore, code aging or decay is one of the reasons that partially explains the increment trend of required effort per fixed number of activities suggested by software entropy concept (Bakota et al., 2012). As the software entropy concept implies, code has the innate trend to decay or loss its structural cohesion over (maintenance) time. However, looser structural cohesion implies higher coupling among code entities (e.g., modules, classes, methods), lower maintainability degree, and thus higher effort during maintenance. In terms of entropy, looser structural cohesion implies code of lower order (higher disorder), and higher entropy as further discussed in subsection 6.4.7.2.

*Developers Experience and Learning Rate:* The scheduler of the simulation model incorporates developers’ experience and their learning rate in different ways through the switch  $r$ : [Constant, Sample, Random]. In ‘Constant’ mode, the factor  $e_{xp}=1$  for all scenario applications, thus it has no effect on standard effort assessment. In the ‘Sample’ mode, the factor  $e_{xp}$  is equal to the sample’s parameter (independent and randomly pre-selected variable) and it remains constant for all scenario applications. Although developers experience is evolved during actual maintenance, this unusual situation is intentionally included for further analysis purposes. In ‘Random’ mode, factor  $e_{xp}$  is set to a random value in the range (0.1, ..., 2.0) for each scenario application, following a left-skewed standard normal distribution. Usually, several developers with variant experience levels could be engaged simultaneously as a team or/and in different periods during actual (real-world) maintenance of a system. Furthermore, in general, companies and developers tend to increase their experience and efficiency, e.g., by hiring specialists, through training, etc. Moreover, the developers’ teams increase their efficiency and their cooperation degree during the course of the projects. In addition, as technologies mature, developers become more familiar or expert. All these reasons arising from real-world circumstances cause a left shifting of the distribution curve of the developers’ experience factor, as confirmed by IT’s community in (Woolf, 2016).

*Interpretability of factors' values:* The stochastic variables (i.e.,  $s_{size}$ ,  $e_{xp}$ ) and the shifting factor age of the simulation model are defined in a specific range of values as reported in Table 6-3. These factors act as a weight on SMC metric in equation (6-4), and thus they have been normalized around the neutral value of unit. For example, the factor of actual scenario size ( $s_{size}$ ) lies between 0.01 (implying a very small scenario in terms of size) and 2.0 (implying a very large scenario in terms of size), where a value of  $s_{size}=1.0$  implies a scenario with average size. When this factor takes random values from the normal distribution of mean  $\mu=1$  and standard deviation  $\sigma=0.33$ ,  $N(1, 0.33, 0, 3)$ , then around 68% of its possible values lies between 0.67 and 1.33, 68%+27% lies between 0.34 and 1.66, 95%+4% lies between 0.01 and 1.99, and the rest  $\approx 1\%$  lies below 0.01 and above 1.99, thus representing possible extremely small or large scenarios). Notice, that even with low probability, values far above the range of 2.0 are possibly to occur. Respectively, the factor of developers' experience ( $e_{xp}$ ) lies between 0.01 (implying a novice developer or team) and 2.0 (implying an expert developer or team), where a value of  $e_{xp}=1.0$  implies a typical competent developer or team. Again, values above the range of 2.0 are possibly to occur. Finally, the increasing factor of code aging ( $a_{ge}$ ) lies between 1.0 (implying a fresh system without aging issues in its code) and 2.0 (implying an old system with aged code). These ranges and the distributions of the stochastic factors have been selected after intensive calibration efforts to be realistic as possible as discussed in subsection 6.4.9.

*Random behavior:* It is important that the introduced simulation model encapsulates random behavior through several probabilistic components or variables which called stochastic (Müller & Pfahl, 2008). If a simulation for the same experiment scenario is repeatedly executed, the results will be different because of the internal randomness introduced by the stochastic variables ( $u_F$ ,  $a_{ge}$ ,  $s_{size}$ ,  $e_{xp}$ ) and random scenario sequences as it would be in a field case study. Especially the uncertainty factor  $f_{BM}$  allows researcher to define the overall uncertainty level of the  $u_F$  internal factor which randomly shifts scenarios' probabilities during simulation. That because the initially assessed scenarios probabilities may be gradually shifted during actual maintenance process by a random and uncertain way. More precisely, the scenarios probabilities are randomly shifted by an overall uncertainty factor  $f_{BM} \times u_F$ , where  $u_F$  factor returns normally distributed random values of zero mean  $\mu=0$ , and standard deviation  $\sigma=\sqrt{\lambda}$ , according to the stochastic Brownian Motion or Wiener process (Bhattacharya & Waymire, 2009; Durrett, 2010). As the maintenance process evolves or  $\lambda$  factor increases, the standard deviation  $\sigma=\sqrt{\lambda}$  of the stochastic values returned by  $u_F$  factor increases too. The  $u_F$  factor represents a stochastic process with stationary independent increments and occur frequently in pure and applied mathematics as well as in quantitative and evolutionary analysis of real-world systems. Furthermore, it is a fundamental process in terms of which more sophisticated stochastic processes can be described.

All the used frequency distributions are realistic assessments mostly derived from the statistical theory and empirical evidence. Furthermore, the simulation model tries to deal with highly unlikely but extreme and important events that may occur without any historical precedent. Normally, simulations from normal distributions allows unbounded bad or good outcomes. Nevertheless, without increasing extreme outcomes' probabilities through fat-tails curves, we may greatly underestimate their likelihood and thus, exhibit high exposure to tail-risk. To deal with these issues, several types of normal frequency distributions, including fat-tails variations, have been tested in a try to explore the effect of highly unlikely but extreme and important outcomes as discussed in subsection 6.4.9. Finally, all variables' values are randomly generated based on each specific distribution type since preliminary tests showed that sophisticated randomized sampling techniques such as Latin Hypercube sampling (Ye, 1998) do not provide any significant improvement.

### 6.4.7.2 Connecting Code Aging with Software Entropy Concept

In this subsection, the connection of code aging (David Lorge Parnas, 1994) with the software entropy concept (Bakota et al., 2012) is documented. One of the main effects of code’s aging is the gradual decrement of its structural cohesion. However, looser structural cohesion implies higher coupling among code entities (e.g., modules, classes, methods), lower maintainability degree, and thus higher effort during maintenance.

Given a set of software requirements there are several possible implementations. Each implementation has its own structure, cohesion degree, coupling degree, etc. However, each structure has different maintainability degree. Structures with high cohesion and low coupling degree are more maintainable but require higher skills and initial development effort (e.g., by using proper design pattern combinations) than structures of lower cohesion and higher coupling degree that may initially requires less development effort but are less maintainable in the future. Given a structure of high cohesion, any future maintenance activities should follow the design principles of the initially selected structure. This is for the interest of developers since this approach requires less maintenance effort and sustains the structural cohesion and maintainability degree of the code. However, in practice, there several reasons that may tempt or even force developers to deviate from this approach. Such reasons may be insufficient code documentation, lack of skills and comprehension regarding the arrangement and operation of the used design patterns, the pressure imposed by strict deadlines, minor or trial functionalities that are carelessly or temporary attached to the code bypassing its formal structure, limited access to relevant source code, etc. Because of all these reasons, the initial structural cohesion and maintainability degree of the code tends to be loosened during its evolution or maintenance process as illustrated in Figure 6-10.

Conceptually, the cohesion degree of a structure reflects its order degree, and thus higher structural cohesion implies higher code order. Since code of high order (structural cohesion) is a state that requires increased skills, control, and design effort to be reached and sustained, it is less likely to spontaneously occur, while as the time pass, states of lower code order (disorder) are more likely to occur. In terms of code’s entropy, code of high order reflects states of low possibility and low entropy while code of low order (disorder) reflects states of high possibility and high entropy as depicted in Figure 6-10. According to the second law of thermodynamics, the entropy of isolated systems left to spontaneous evolution cannot decrease with time, as they always arrive at a state of thermodynamic

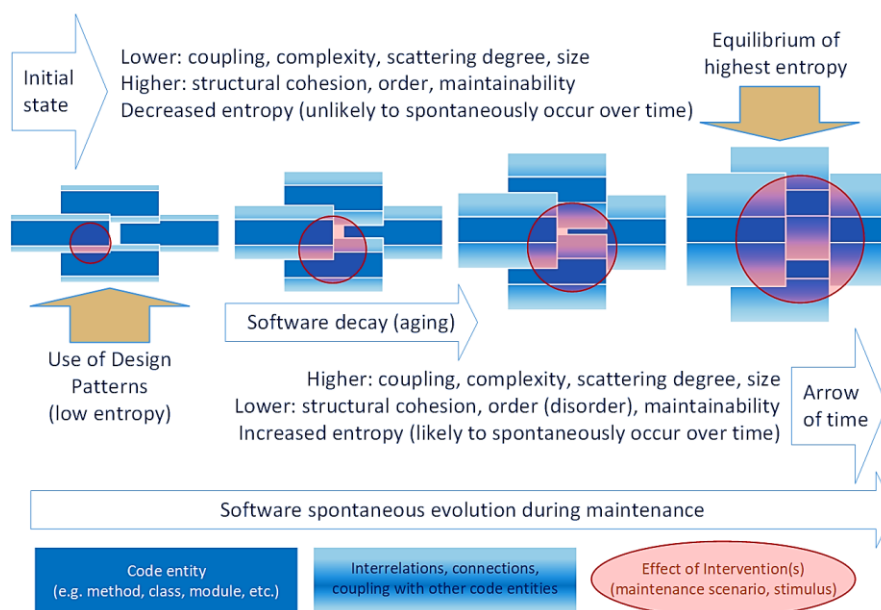


Figure 6-10: Software entropy concept in relation to code’s aging.

equilibrium, where the entropy is highest. Respectively, the entropy of isolated software systems left to spontaneous evolution cannot decrease with time, as they always arrive at a state of equilibrium, where the entropy is highest (high probability and low code order). Here, the phrase of “isolated software systems left to spontaneous evolution” refers to the maintenance (evolution) of code in a spontaneous manner, thus concentrating on (spontaneous) interventions that satisfies new functionalities and requirements and excluding (targeted) interventions that reforms and improve the structure and cohesion degree of the code. Conclusively, software entropy concept implies that the code has the innate trend to decay (age) or loss its structural cohesion over (maintenance) time given that no reforming activities of its structure take place.

#### 6.4.7.3 MATLAB Functional Representation of Simulation Model

An indicative and abstract representation of the introduced simulation model is provided in Figure 6-11. The provided pseudocode is a simplification of the model’s functional implementation in MATLAB environment. Several details, subroutines, intermediate variables, initializations, and marginal conditions have been omitted or integrated to keep the representation condense and focused on its basic functionality. The focus is on the principal function `SM_Cost_CVP_CIBI()` which performs a single (one time) simulation of the maintenance process for the general problem of recursive hierarchies of part-whole aggregations (CVP vs CIBI). The characteristics ( $N$ ,  $M$ ,  $p_{nE}$ ,  $p_{nP}$ ,  $e_{xp}$ ) of each specific (sample) instance of the general problem are declared as separate parameters (avoiding tables for simplicity). In addition, several other parameters ( $f_{BM}$ ,  $\lambda$ ) and switches ( $a_{lt}$ ,  $s_{sizing}$ ,  $r$ ,  $a_{ging}$ ) related to the control of model’s stochastic behavior are declared. Furthermore, extra parameters to control the characteristics of the used frequency distributions for developers’ experience and scenarios’ size have been added. The function enables different levels of stochastic behavior according to the values of these parameters, thus enabling the user to run all the simulation states in Table 6-4. The effort estimations for both design alternatives (CVP, CIBI) and for the entire sequence of applied scenarios ( $\lambda$ ) are the function’s output. The model generates sequences of several auxiliary values such as sequences of age, size, experience, design attributes, simulated cost, and stochastic factors to keep track their progress as different types of maintenance scenarios are repeatedly applied. These sequences of values are gradually and commonly applied in both design alternatives to ensure fair comparison conditions. The evolution policy of the model is defined by the discrete types of scenarios, implying a discrete-event or event-driven model. The SMC metric (Table 6-2) for different types of scenarios is also integrated to facilitate a homogenous measurement process. However, SMC metric could be coded in a different function for better abstraction. The used identifiers of the parameters are in accordance with the notation used in Table 6-3 and Figure 6-7. Finally, this function can be repeatedly executed (by other functions) to perform massive simulations: a) on different sample instances of the general problem for generalization and statistical validation purposes, b) on the same instance (Monte Carlo approach) to investigate frequency patterns, variability, error rate, and convergence, or even c) combinedly (Monte Carlo approach to all sample instances) to investigate overall variability and error rate control. Respectively, the formal models as expressed by the equations (6-1) and (6-2) can be directly coded in parametric functions to support comparison purposes.

FUNCTION: SM_Cost_CVP_CIBI		sample	controled	extra	stochastic
<b>IN:</b>	N, M, P <sub>nE</sub> , P <sub>nP</sub> , P <sub>nnew</sub> , P <sub>edit</sub> , P <sub>delete</sub> , E <sub>xp</sub> , λ, f <sub>BM</sub> , a <sub>it</sub> , a <sub>ging</sub> , S <sub>izing</sub> , Size_mean, Size_std, Size_skew, Size_kurt, r, Exp_mean, Exp_std, Exp_skew, Exp_kurt				
<b>OUT:</b>	SCm_seq(1:2, 1:λ+1) % Simulated Cost (effort) estimations for {CVP, CIBI} and λ applied scenarios				
<b>AUX:</b>	DAs_seq(1:2, 1:λ+1) % sequence of current values of Design Attributes N,M per scenario application (λ) SF_seq(1:λ) % sequence of values of Stochastic Factor per scenario application (λ)				
BM_seq= Brownian_motion(λ); % sequence of a random walk with λ steps, μ=0, σ=sqrt(λ) DAs_seq(1,1)= N; DAs_seq(2,1)= M; % initial design attribute values (N and M) for λ=1 IF (a <sub>ging</sub> ='Constant') THEN Age_seq(1:λ)=1; ELSE Age_seq=1.0+[1:λ]/λ; END IF; IF (S <sub>izing</sub> ='Constant') THEN Size_seq(1:λ)=1; ELSE Size_seq(1:λ)= RANDOM N(Size_mean, Size_std, Size_skew, Size_kurt, λ, 1)); END IF; IF (r='Constant') THEN Exp_seq(1:λ)=1; ELSE IF (r='Random') THEN Exp_seq(1:λ)= RANDOM N(Exp_mean, Exp_std, Exp_skew, Exp_kurt, λ, 1))+0.1; ELSE Exp_seq(1:λ)= E <sub>xp</sub> ; END IF;					
<b>REPEAT FOR t=2 UNTIL λ+1</b> % applying λ scenarios and SMC metric (commonly for CVP, CIBI) DAs_seq(1,t)= DAs_seq(1,t-1); DAs_seq(2,t)= DAs_seq(2,t-1); %copy design attributes to next step SCm_seq(1,t)= SCm_seq(1,t-1); SCm_seq(2,t)= SCm_seq(2,t-1); %copy simulated cost (effort) to next step U <sub>r</sub> = f <sub>BM</sub> * BM_seq(t) / (10*sqrt(t)); % overall Uncertainty Factor affecting scenarios probabilities SF_seq(t-1) = (Age_seq(t-1)*(Size_seq(t-1))/Exp_seq(t-1)); % Stochastic Factor affecting SMC <b>RANDOMLY SELECT SCENARIO_TYPE BASED ON PROBABILITIES P<sub>nE</sub>, P<sub>nP</sub> AS AFFECTED BY U<sub>r</sub></b> <b>RANDOMLY SELECT SCENARIO_ACTIVITY BASED ON a<sub>it</sub> AND PROBABILITIES P<sub>nnew</sub>, P<sub>edit</sub>, P<sub>delete</sub> AS AFFECTED BY U<sub>r</sub></b>					
<b>IN CASE OF SCENARIO_TYPE IS 'Element'</b>					
<b>IN CASE OF SCENARIO_ACTIVITY IS 'Addition'</b>					
DAs_seq(1,t) += 1; % N++ Increasing Design Attribute of Element type					
SCm_seq(1,t) += SF_seq(t-1)*(2*DAs_seq(2,t)+4); % SMC for CVP=2*(M+2)					
SCm_seq(2,t) += SF_seq(t-1)*(DAs_seq(2,t)+1); % SMC for CIBI=M+1					
<b>IN CASE OF SCENARIO_ACTIVITY IS 'Modification'</b>					
SCm_seq(1,t) += SF_seq(t-1)*(2*DAs_seq(2,t)); % SMC for CVP=2*M					
SCm_seq(2,t) += SF_seq(t-1)*(DAs_seq(2,t)+1); % SMC for CIBI=M+1					
<b>IN CASE OF SCENARIO_ACTIVITY IS 'Deletion'</b>					
DAs_seq(1,t) -= 1; % N-- Decreasing Design Attribute of Element type					
SCm_seq(1,t) += SF_seq(t-1)*(DAs_seq(2,t)+1); % SMC for CVP=1+M					
SCm_seq(2,t) += SF_seq(t-1)*1; % SMC for CIBI=1					
<b>IN CASE OF SCENARIO_TYPE IS 'Operation'</b>					
<b>IN CASE OF SCENARIO_ACTIVITY IS 'Addition'</b>					
DAs_seq(2,t) += 1; % M++ Increasing Design Attribute of Operation type					
SCm_seq(1,t) += SF_seq(t-1)*(DAs_seq(1,t)+1); % SMC for CVP=N+1					
SCm_seq(2,t) += SF_seq(t-1)*(2*DAs_seq(1,t)+2); % SMC for CIBI=2*(N+1)					
<b>IN CASE OF SCENARIO_ACTIVITY IS 'Modification'</b>					
SCm_seq(1,t) += SF_seq(t-1)*(DAs_seq(1,t)+1); % SMC for CVP=N+1					
SCm_seq(2,t) += SF_seq(t-1)*(2*DAs_seq(1,t)); % SMC for CIBI=2*N					
<b>IN CASE OF SCENARIO_ACTIVITY IS 'Deletion'</b>					
DAs_seq(2,t) -= 1; % M-- Decreasing Design Attribute of Operation type					
SCm_seq(1,t) += SF_seq(t-1)*1; % SMC for CVP=1					
SCm_seq(2,t) += SF_seq(t-1)*(DAs_seq(1,t)+1); % SMC for CIBI=1+N					

Figure 6-11: Abstract (indicative) representation of the Simulation Model implementation as functional model.

#### 6.4.7.4 DSL Modular Representation of Simulation Model

An indicative and modular representation of the model in the form of class diagram of a typical object-oriented language is provided in Figure 6-12. Furthermore, a run-time representation of model's objects is depicted in Figure 6-13. Several details, auxiliary methods, intermediate variables, and marginal conditions have been omitted to keep the representation condense and focused on its basic functionality. During run-time, each design alternative of the problem (e.g., CVP) is conceptually represented by a hierarchy or tree of associated objects each of them representing a specific design pattern or artifact (e.g., Composite, Visitor). Referring to the class diagram, the engaged design patterns (e.g., Composite\_VP) are represented in a hierarchy of sub-classes under the class "Artifact" where each of them can contain other artifacts. This structure is a composite implementation that allows its extension with other artifacts that can be combined or attached in various possible ways during run-time. Each artifact has its own design attributes (e.g., N, M) and some common properties like its age and the required effort. For example, during run-time, the CIBI design alternative is represented by the single (root) design pattern "Composite\_IBI", while CVP by the root pattern "Composite\_VP" and the attached "Visitor" pattern separately. The maintenance scenarios or stimuluses are applied on each type of artifact in different ways based on SMC metrics (i.e., Table 6-2). This

implementation is an instance of the general design problem of recursive hierarchies of part-whole aggregations, which ironically is the subject of this study (i.e., CIBI vs CVP). This is a nice opportunity to demonstrate how this design decision can be supported by the formal models. In this case,  $N=3$  (distinct types of design patterns or artifacts) and  $M=6$  (distinct types of scenarios or stimulus). Given that each future addition of an artifact (e.g., Decorator pattern) requires approximately three new types of stimulus (e.g., Decorator Addition, Modification, and Deletion) in an analogy 1:3, it could be safely assumed that the probability of a new element against a new operation is  $p_{nE}=0.25$ :  $p_{nP}=0.75$ . In less than a minute, the equation (6-3) indicates that  $c_m(CVP) < c_m(CIBI)$  and thus, CVP is the most beneficial design alternative in terms of maintainability.

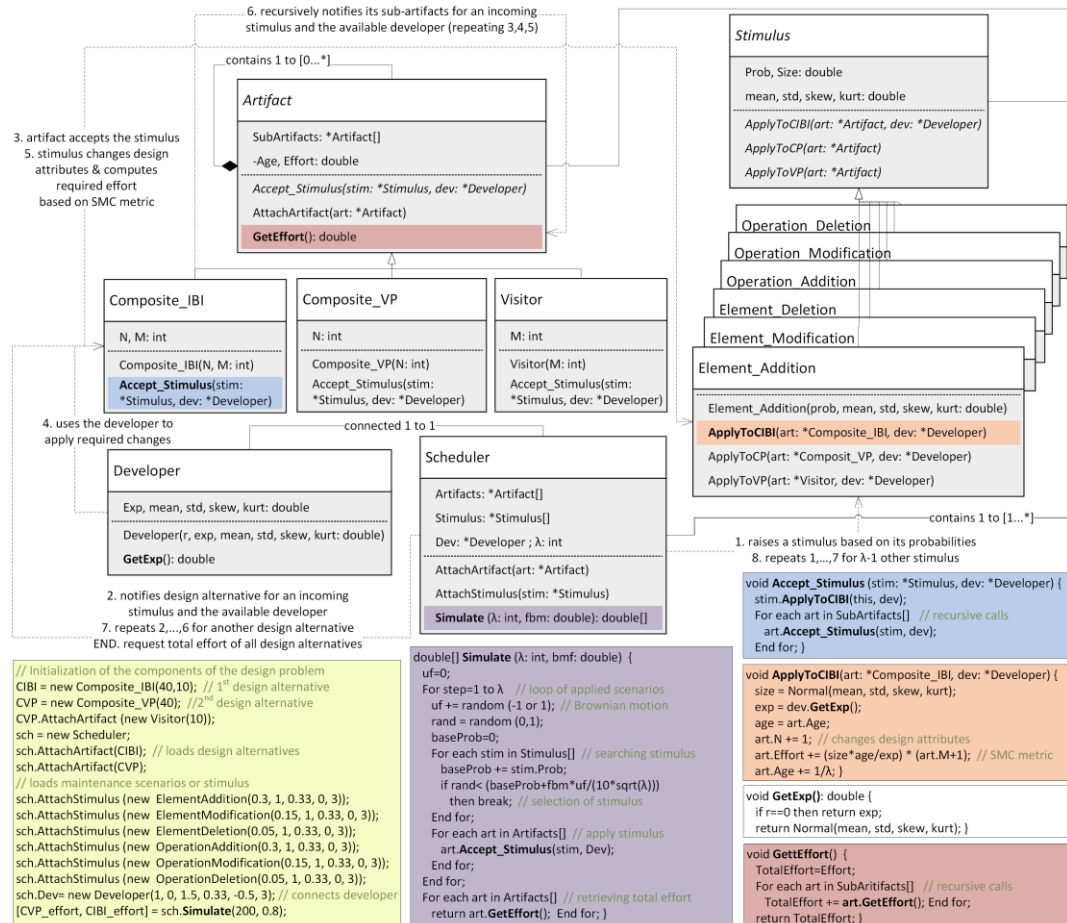


Figure 6-12: Class diagram of an indicative modular representation of the simulation model

Under visitor (CVP) approach, each stimulus (e.g., Element Addition) is represented in a hierarchy of sub-classes under the class “Stimulus”, including characteristics like its probability and its size (shape of distribution). In addition, each type of stimulus contains one method per artifact type (e.g., “ApplyToCIBI”) responsible to apply this specific stimulus on that specific artifact. The developer is represented as a separate class, including characteristics like its experience (shape of distribution). The “Scheduler” class represent the controller of the simulation model with which all the root artifacts (problem’s design alternatives), the stimulus, and the developer objects are associated. Initially the components of the general problem under study should be loaded in the form of associated instance of classes. The method “Simulate()” initiates the simulation process in a similar manner as in the functional variation by randomly raising stimulus based on their probability. The application of the randomly raised stimulus (e.g., Element Addition)



takes places through the method “Accept\_Stimulus()” which invites the stimulus to act on a specific root artifact. The critical difference is that this method acts recursively or propagates through all the sub-artifacts of each artifact. Thus, beginning from a root artifact (e.g., Composite\_VP), all its sub-artifacts (e.g., Visitor) will invite the same stimulus to act on them. Each stimulus through its corresponding method (e.g., ApplyToCIBI()) acts on an artifact by updating its design attributes, its age factor, and its total effort as affected by the overall uncertainty factor in equation (4). Notice that these methods are referred to specific artifacts (e.g., Visitor design pattern) and not to the entire design alternative (e.g., CVP), implying that the effect of the SMC metric should be properly distinguished per engaged design pattern. This approach allows the definition of other design patterns (as artifacts) which could be combined with others existing artifacts during run-time to represent more complex design alternatives. Furthermore, it provides the potential for disassociated characteristics (e.g., age levels) per artifact (design pattern) instead of a uniform characteristic for the entire design alternative. Respectively, it provides the potential for different characteristics (e.g., size distribution) per stimulus instead of a uniform characteristic for all stimuluses. In addition, simulations of higher resolution and stochastic behavior can be supported (e.g., by subclassing existing classes).

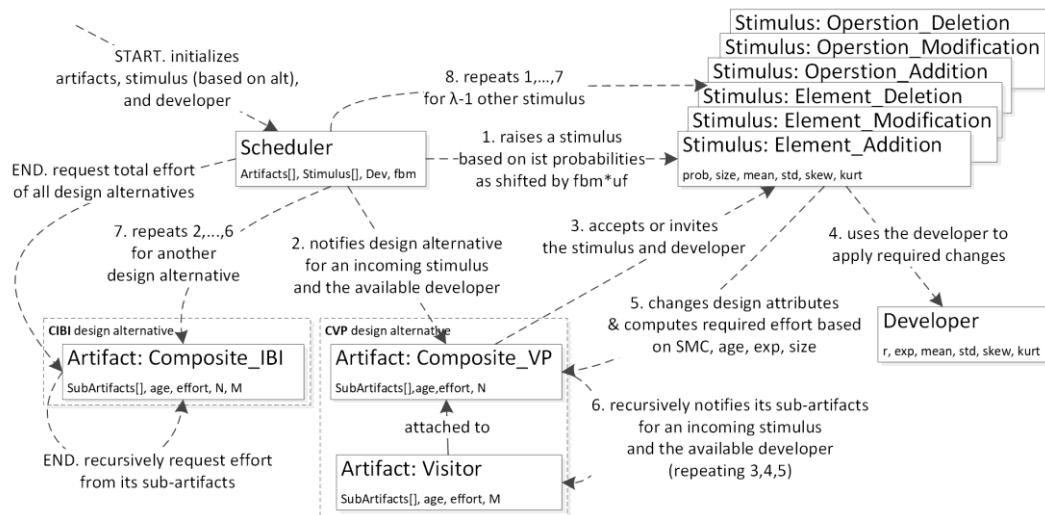


Figure 6-13: Object diagram of an indicative run-time representation of the modular simulation model

Indicative examples of intermediate results/outcomes of the simulation model for the CVP design combination are presented in Figure 6-14. Moreover, an example of the Graphic User Interface (GUI) of the simulation model’s implementation is demonstrated in Figure 6-15.

```

=====
Scheduler name: CVP Scheduler Iterations (λ):20
-----
Name: Ne p:0,275 | Np_CVP p:0,275 | Ee p:0,2 | Ep_CVP p:0,2 | Re p:0,025 | Rp_CVP p:0,025
-----
Scenarios' sequence: { Ne Ne Ep_CVP Ee Ep_CVP Ne Ep_CVP Ne Ne Np_CVP Ep_CVP Ee Ne Ep_CVP Ee Ep_CVP Ne Np_CVP Ne Ne }
Scenarios' size sequence: [ 1,06 0,93 1,15 0,78 0,26 1,54 0,97 1,51 0,68 0,98 1,61 0,73 0,55 0,65 0,73 1,20 0,80 1,42 1,05 1,82 ]
Aging factors sequence: [ 1,025 1,05 1,075 1,1 1,125 1,15 1,175 1,2 1,225 1,25 1,275 1,3 1,325 1,35 1,375 1,4 1,425 1,45 1,475 1,5 ]
Developers' experience (factors) sequence: [ 0,87 0,83 1,17 1,01 1,03 0,92 1,23 1,65 1,34 1,00 1,46 0,98 0,99 0,88 0,94 1,25 0,87 1,08 1,61 1,76 ]
=====
Update N:30 to 31
Run Scenario:Ne Action:1 Type:1 sub-MCost:20
Current λ:1 Age*Size/Exp: 1,24 Cumulative Cost/Effort: 24,97
Update N:31 to 32
Run Scenario:Ne Action:1 Type:1 sub-MCost:20
Current λ:2 Age*Size/Exp: 1,18 Cumulative Cost/Effort: 48,72
Run Scenario:Ep_CVP Action:2 Type:3 sub-MCost:33
Current λ:3 Age*Size/Exp: 1,06 Cumulative Cost/Effort: 83,76
Run Scenario:Ee Action:2 Type:1 sub-MCost:20
Current λ:4 Age*Size/Exp: 0,84 Cumulative Cost/Effort: 100,70
Run Scenario:Ep_CVP Action:2 Type:3 sub-MCost:33
Current λ:5 Age*Size/Exp: 0,29 Cumulative Cost/Effort: 110,31
Update N:32 to 33
Run Scenario:Ne Action:1 Type:1 sub-MCost:20
Current λ:6 Age*Size/Exp: 1,91 Cumulative Cost/Effort: 148,67
Run Scenario:Ep_CVP Action:2 Type:3 sub-MCost:34
Current λ:7 Age*Size/Exp: 0,92 Cumulative Cost/Effort: 180,27
Update N:33 to 34
Run Scenario:Ne Action:1 Type:1 sub-MCost:20
Current λ:8 Age*Size/Exp: 1,09 Cumulative Cost/Effort: 202,20
Update N:34 to 35
Run Scenario:Ne Action:1 Type:1 sub-MCost:20
Current λ:9 Age*Size/Exp: 0,62 Cumulative Cost/Effort: 214,66
Update M_CVP:10 to 11
Run Scenario:Np_CVP Action:1 Type:3 sub-MCost:36
Current λ:10 Age*Size/Exp: 1,22 Cumulative Cost/Effort: 258,70
Run Scenario:Ep_CVP Action:2 Type:3 sub-MCost:36
Current λ:11 Age*Size/Exp: 1,39 Cumulative Cost/Effort: 309,09
Update N:35 to 36
Run Scenario:Ne Action:1 Type:1 sub-MCost:22
Current λ:12 Age*Size/Exp: 0,96 Cumulative Cost/Effort: 330,26
Update N:36 to 37
Run Scenario:Ne Action:1 Type:1 sub-MCost:22
Current λ:13 Age*Size/Exp: 0,74 Cumulative Cost/Effort: 346,69
Run Scenario:Ep_CVP Action:2 Type:3 sub-MCost:37
Current λ:14 Age*Size/Exp: 0,99 Cumulative Cost/Effort: 383,62
Run Scenario:Ee Action:2 Type:1 sub-MCost:22
Current λ:15 Age*Size/Exp: 1,06 Cumulative Cost/Effort: 407,03
Run Scenario:Ep_CVP Action:2 Type:3 sub-MCost:37
Current λ:16 Age*Size/Exp: 1,33 Cumulative Cost/Effort: 456,56
Update N:36 to 37
Run Scenario:Ne Action:1 Type:1 sub-MCost:22
Current λ:17 Age*Size/Exp: 1,30 Cumulative Cost/Effort: 485,28
Update M_CVP:11 to 12
Run Scenario:Np_CVP Action:1 Type:3 sub-MCost:38
Current λ:18 Age*Size/Exp: 1,91 Cumulative Cost/Effort: 558,02
Update N:37 to 38
Run Scenario:Ne Action:1 Type:1 sub-MCost:24
Current λ:19 Age*Size/Exp: 0,95 Cumulative Cost/Effort: 581,01
Update N:38 to 39
Run Scenario:Ne Action:1 Type:1 sub-MCost:24
Current λ:20 Age*Size/Exp: 1,55 Cumulative Cost/Effort: 618,24
=====
Total MCost: 618,24
=====

```

Figure 6-14: Example of intermediate results/outcomes of DSL implementation of the Simulation Model for CVP design combination

Referring to Figure 6-14, the number of applied scenarios has been defined to 20. The probability for a scenario effecting an element is equal to 0.5 which further interpreted by the model as new element probability ( $p_{nE}=0.275$ ), edit element probability ( $p_{eE}=0.2$ ), and remove element probability ( $p_{rE}=0.025$ ). The probabilities for a scenario effecting an operation ( $p_{nP}$ ,  $p_{eP}$ , and  $p_{rP}$ ) are defined accordingly. The model generates sequences of a) scenarios' types based on previous scenarios' probabilities, b) scenarios' actual size factors, c) aging factors, and d) developers' experience factors. These sequences of events and factors are commonly applied on both design alternatives under comparison.

It is essential that the simulation model incorporates internal randomness. For example, considering a specific experiment scenario with independent variables  $N=30$ ,  $M=18$ ,  $p_{nE}=0.7$ ,  $\lambda=30$ , in 7<sup>th</sup> fully stochastic simulation state. If this simulation for the same experiment scenario is repeatedly executed, the results will be different, because of the internal randomness introduced by the random variables ( $a_{ge}$ ,  $s_{size}$ ,  $e_{xp}$ ) and random scenario sequences as it would be in a real case study. These results highlight the vastness of the potential cases under exploration that are not only limited to the independent variables but also to the internal randomness. For example, for  $\lambda=30$ , the simulations for the same experiment scenario would have approximately  $10^{72}$  different possible outcomes or  $\lambda! / ((p_{nE} \cdot \lambda)! ((1-p_{nE}) \cdot \lambda)!)$  different scenario's sequences  $\times \lambda!$  different scenario's sizes  $\times \lambda!$  developers' experience levels. Nevertheless, it is expected that all these outcomes would converge between them to some degree because of their inverse relationship and statistic behavior of normally distributed factors based on Central Limit Theorem claims. However, this random behavior helps to explore the authors' claims according to which the simplified proposed formal models and modeling method take into consideration all these random factors in an indirect but sufficient statistical way.

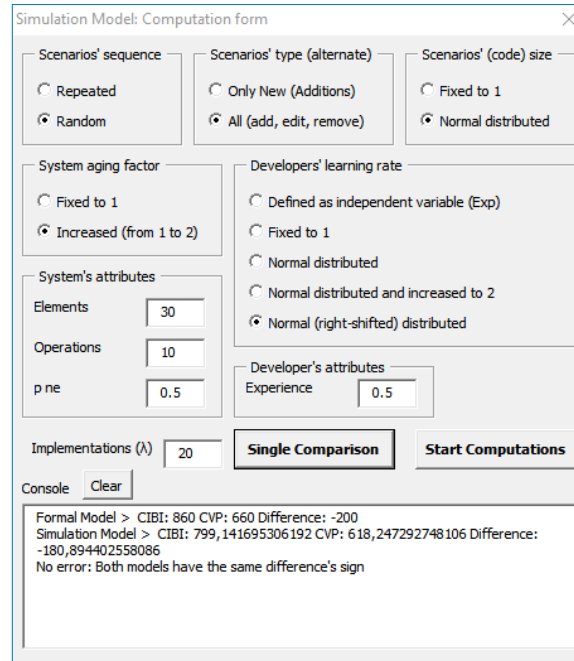


Figure 6-15: Example of GUI of the DSL Implementation of Simulation Model

#### 6.4.8 Conducting and Data Validation

*Preparation and Execution:* Initially, the sample of 1000 object-subject instances, represented by  $N$ ,  $M$ ,  $p_{nE}$ ,  $e_{xp}$  variables in Table 6-3, is randomly generated through a computer-aided generator. Next, the forma and simulation models or treatments are massively applied in all relevant experiment scenarios for several simulation states in Table 6-4. The experiment's (raw) results of the 7<sup>th</sup> simulation state are provided online for further research purposes in (Karanikolas, Dimitroulakos, & Masselos, 2021).

*Data normality control:* Initially, the frequency distributions of some indicative simulated ( $sc_m$ ) outcomes for all (1000) object-subject instances of the selected sample are presented in Figure 6-16. Respectively, indicative formal ( $c_m$ ) outcomes are presented in Figure 6-17. The number of scenario application is  $\lambda = 200$  relevant to the 7<sup>th</sup> simulation state in Table 6-4. The outcomes reflect effort assessments for CVP and CIBI combinations including their difference value (CVP-CIBI) which eventually defines the decision-making. All the outcome's distributions approximate the normal distribution without indications about outlier values, and hence there is no need for any data reduction. Furthermore, the sample's normality sufficiently implies population normality, thus amplifying the reliability of the conducted statistical analysis.

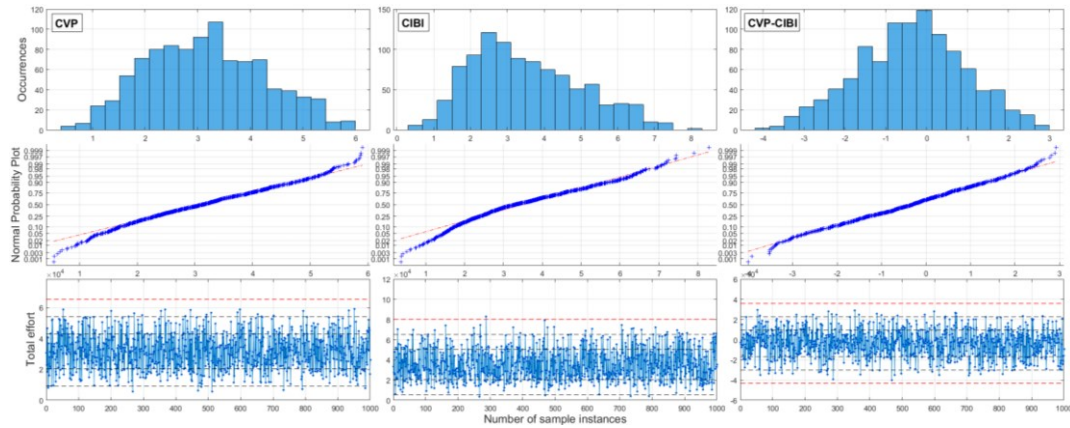


Figure 6-16: Frequency distributions of Simulation Model’s CVP, CIBI total effort assessments, and their differences, for all (1000) object-subject instances of the selected sample, where  $\lambda = 200$ , relevant to the 7<sup>th</sup> simulation state in Table 6-4.

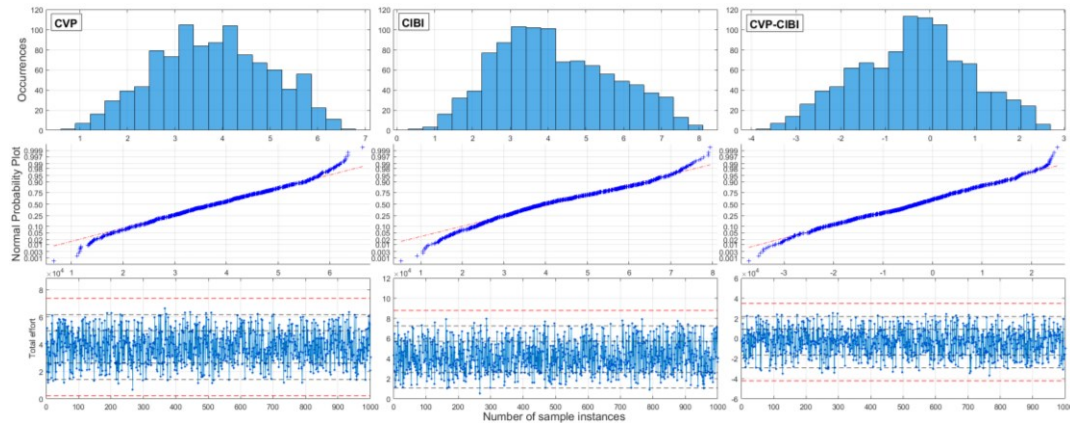


Figure 6-17: Frequency distributions of Formal Model’s CVP, CIBI total effort assessments, and their differences, for all (1000) object instances of the selected sample, where  $\lambda = 200$ , relevant to the 7<sup>th</sup> simulation state in Table 6-4.

*SM Internal convergence control:* Indicative frequency distributions and related scatter diagrams of simulated ( $sc_m$ ) outcomes, concerning 100 repeated simulations in a single object-subject instance of the selected sample ( $N:40, M:10, p_{ne}:0.5, e_{xp}:1$ ), are presented in Figure 6-18. The number of scenario application is  $\lambda = 200$  relevant to the 7<sup>th</sup> simulation state in Table 6-4. The outcomes include the simulated effort assessments for CVP and CIBI combinations as well as their difference values (CVP-CIBI). All the outcomes are normally distributed and thus, are targeted in a limited interval. Conceptually, if effort expectations, implied by formal model’s predictions, are rational, then the actual required effort approximated by the simulation model’s outcomes should be normally distributed around these expectations. Furthermore, the evidence confirms the stochastic behavior of the simulated maintenance process. As intuitively implied, for most human activities, even if a specific system had been repeatedly maintained for several times under similar conditions, the outcomes would be different in some degree, however, converging in a limited interval.

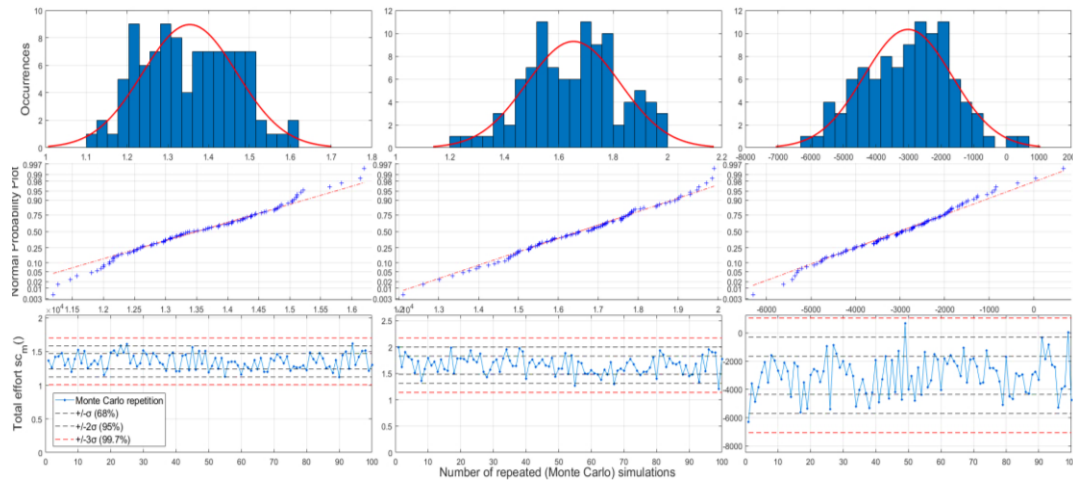


Figure 6-18: Indicative frequency distributions and scatter diagrams of Simulation Model's outcomes (CVP, CIBI, CVP-CIBI) for 100 repeated simulations in a single object-subject instance (N:40, M:10,  $p_{ne}$ :0.5,  $e_{xp}$ :1.0), where  $\lambda = 200$  relevant to the 7<sup>th</sup> simulation state in Table 6-4.

## 6.4.9 Calibration of Model's Stochastic Behavior

### 6.4.9.1 Multi-Resolution Modeling Approach

In principle, the simulation model should imitate the heavily stochastic and uncertain nature of the actual (real-world) maintenance process. The model incorporates developers' stochastic characteristics such as experience level and learning rate ( $e_{xp}$ ) as well as the actual size ( $s_{size}$ ) of the affected code for each scenario application, as they overall expressed by the stochastic factor in equation (6-4). Furthermore, the estimated (by the designers) values of scenarios' probabilities (i.e.,  $p_{nE}$ ,  $p_{nP}$ ) may deviate from the realized probabilities during actual maintenance process. The overall uncertainty factor  $u_f$  has been incorporated to address this issue. However, the stochastic behavior of the simulation model, as described in previous sub-sections and Table 6-3, arises questions about the realism of model's outcomes. To address these concerns, the simulation model has been calibrated based on frequency distributions of real-world evidence of relevant studies from the field of time series analysis (G. Antoniol et al., 2001; Raja et al., 2009; Shariat Yazdi et al., 2016).

More specifically, during the calibration process the multi-resolution modeling approach (Zeigler et al., 2018) has been followed. This approach is about gradually constructing variations of the simulation model in a try to accomplish the required (actual) behavior while at the same time the consistency of each model variation is checked compared to reliable evidence of the phenomenon under study. Furthermore, this approach increases the trustworthiness degree of the simulations through a methodology for constructing a multiresolution family of models as visualized in Figure 6-19. Initially, the desired stochastic behavior of the problem is expressed through a set of requirements and constraints. The technique targets on a simulation model that satisfies all these requirements, called base model. To reach the target model, lumped models are created by introducing assumptions (regarding the base model) such as dropping of requirements and relaxing of constraints. Next, models of higher resolution are created by removing the previously added assumptions. To this direction, more refined representations (i.e., by introducing underlying activities, or stochastic factors or/and different dimensions) are included to address the affected constraints and requirements. The critical point is whether this (lumped) model variation is trustworthy and consistent compared to real world circumstances or not. A possible validation based on real world observations is subject to the same constraints as the formal model validation (i.e., inadequate volume of homogenous observations for the specific design problem). Thus, frequency distributions

of real-world evidence from the field of time series analysis are used for testing the consistency of each (lumped) model variation, mostly regarding the variability of its outcomes (i.e., effort assessments). In case of higher consistency, then further assumptions are removed toward modes of even higher resolution, while in case of lower consistency, further assumptions are added toward modes of lower resolution. The targeted base model is achieved when all the initial assumptions have been eliminated. In this case, all the validity concerns (reported in subsection 6.3) are the assumptions made for the initial lumped model. In fact, all the simulation states in Table 6-4 are variations of (lumped) modes while the last two states are considered as the targeted base model. The introduced simulation model can represent different level of disaggregation through its parameters, thus supporting all the variations in Table 6-4. Furthermore, several other (lumped) modes have been tested through sensitivity analysis on model’s parameters including intervals of their values. Furthermore, the analysis showed that adding further stochastic behavior or detailed simulation of other activities does not significantly affect the model’s consistency mostly because their impact is common for all design alternatives, thus not affecting the comparison outcome and decision-making reliability. More specifically, several detailed simulations (lumped models) have been tested including the representation of a) the method interventions as separate sub-activities with distinct actual sizes, b) individual developers (or teams of developers) as separate sub-activities with distinct experience levels and learning rates, c) separate aging factors per engaged design pattern for each design alternative, d) minor maintenance scenarios as separate sub-activities with distinct actual sizes (similar for all design alternatives) which, however, not affect the design attributes of the addressed problem.

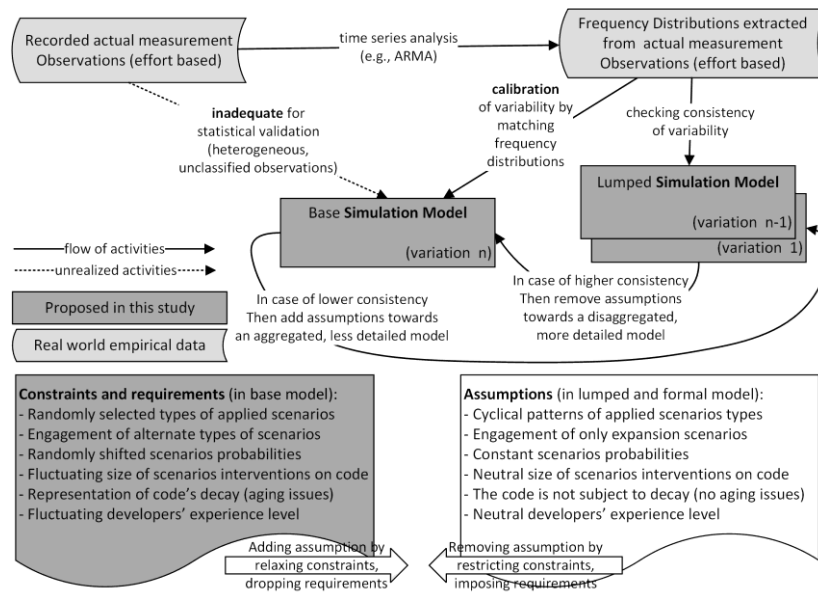


Figure 6-19: Multi-resolution modeling approach towards calibration of Simulation Model

#### 6.4.9.2 Consistency Criterion

Concerning the consistency criterion, the sequences of values for the stochastic factor in equation (6-4) and the intermediate values of effort assessments  $sc_m(CVP)$  and  $sc_m(CIBI)$  per scenario application have been transformed to time series in respect to the number of applied maintenance scenarios ( $\lambda$ ), as showed in Figure 6-20. Again, Figure 6-20 presents results for the Interpreter implementation ( $N:40, M:10, p_{nE}:0.5$ ) as an indicative instance of the CVP vs CIBI general problem. The histograms of their frequency distributions demonstrate a normal pattern. The characteristics of these time series and their distributions have been compared and coordinated with the empirical evidence of real

systems. More specifically, large volumes of recorded measurements concerning the changes (in terms of low and high level edit operations) made during software evolution between revisions of several real systems have been statistically modeled using Auto Regression Moving Average (ARMA) models in (Shariat Yazdi et al., 2016). The extracted time series and the characteristic of their frequency distributions can be used to calibrate and control the generation of realistic histories of relevant measurements by simulation models, as suggested in (Shariat Yazdi et al., 2016). Of course, changes per revision are not equivalent to changes per scenario application, however there is a direct correspondence and similarity between them at least from a statistical perspective, making these evidence suitable for calibration purposes. The most important characteristic of a frequency distribution is the coefficient of variation ( $CV=\sigma/\mu$ ) which is a dimensionless parameter, ideal for comparison between data sets with widely different means or different units. The introduced simulation model, due to its sensitivity on the structural behavior of the engaged design patterns and their increasing trend (confirmed in chapter 3), demonstrates distributions of intermediate effort measurements with a slightly lower coefficient of variation (CV), right skewed, and similar – near to average kurtosis compared to the empirical evidence in (Shariat Yazdi et al., 2016). Furthermore, given that the CV of the intermediate effort observations per revision lies between 3 and 4, the expected CV of the total (after  $\lambda=200$  revisions) effort observations per (real world) system lies between  $3/\sqrt{200}=0.22$  and  $4/\sqrt{200}=0.28$  (due to Central Limit theorem properties). After intensive calibration efforts, the introduced simulation model demonstrates an overall  $CV\approx 0.25$  concerning the total effort assessments of all sample's instances, referring to the 6<sup>th</sup> simulation state in Table 6-4. Respectively, under the 7<sup>th</sup> simulation state the model demonstrates an overall  $CV\approx 0.32$  as depicted in Figure 6-16.

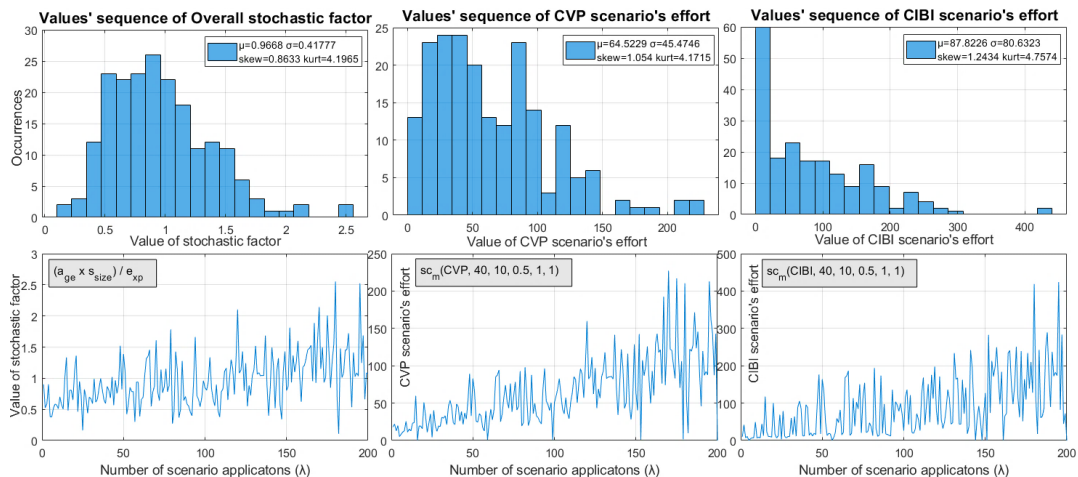


Figure 6-20: Frequency distributions and time series of Simulation Model's overall stochastic factor and intermediate outcomes (CVP, CIBI) of an indicative single object-subject instance ( $N:40, M:10, p_{ne}:0.5, e_{xp}:1.0$ ), where  $\lambda = [1, \dots, 200]$ , relevant to the 7<sup>th</sup> simulation state in Table 6-4.

#### 6.4.9.3 Coefficient of Variation for Decision-Making Reliability

In this subsection, the underlying concept of the followed consistency criterion during the calibration of the simulation model under the multi-resolution modeling technique is further analyzed and documented. The conceptual analysis is visually represented in Figure 6-21. The principal idea is in the proportional equivalent effort assessments returned by the simulation model due to the adoption of SMC metrics. Referring to the CVP vs CIBI design problem, whatever the actual (real-world) effort assessments of CVP and CIBI alternatives are, the corresponding simulated effort assessments would be in an analogy with the actuals. Visualizing the frequency distributions of actual effort

assessments in a horizontal axe, the proportional simulated assessments correspond to a stretched version of this axe. Furthermore, given the (normal) frequency distribution of possible (realized) effort assessments per design alternative (i.e., CVP, CIBI), the possibility of an incorrect decision is expressed by the overlapping areas of these distributions. The greater the surface of the overlapping area, the greater the likelihood of an incorrect design decision to occur. Thus, the reliability degree of decision-making among design alternatives depends on the percentage of the overlapping area of relevant frequency distributions as visualized in Figure 6-21. Given the analogy of measurements, the percentage of the overlapping area is irrelevant of effort assessments in terms of absolute values ( $\mu$ :mean) and mainly depends on the variability (or precision  $\sigma$ :standard deviation) of the corresponding frequency distributions as expressed by their coefficients of variation ( $CV=\sigma/\mu$ ) which is a dimensionless statistical parameter. Conclusively, the consistency of the simulation model against real-world circumstances, concerning its decision-making reliability, mainly depends on matching the coefficients of variation (CV) among simulated and real-world observations as showed in Figure 6-21.

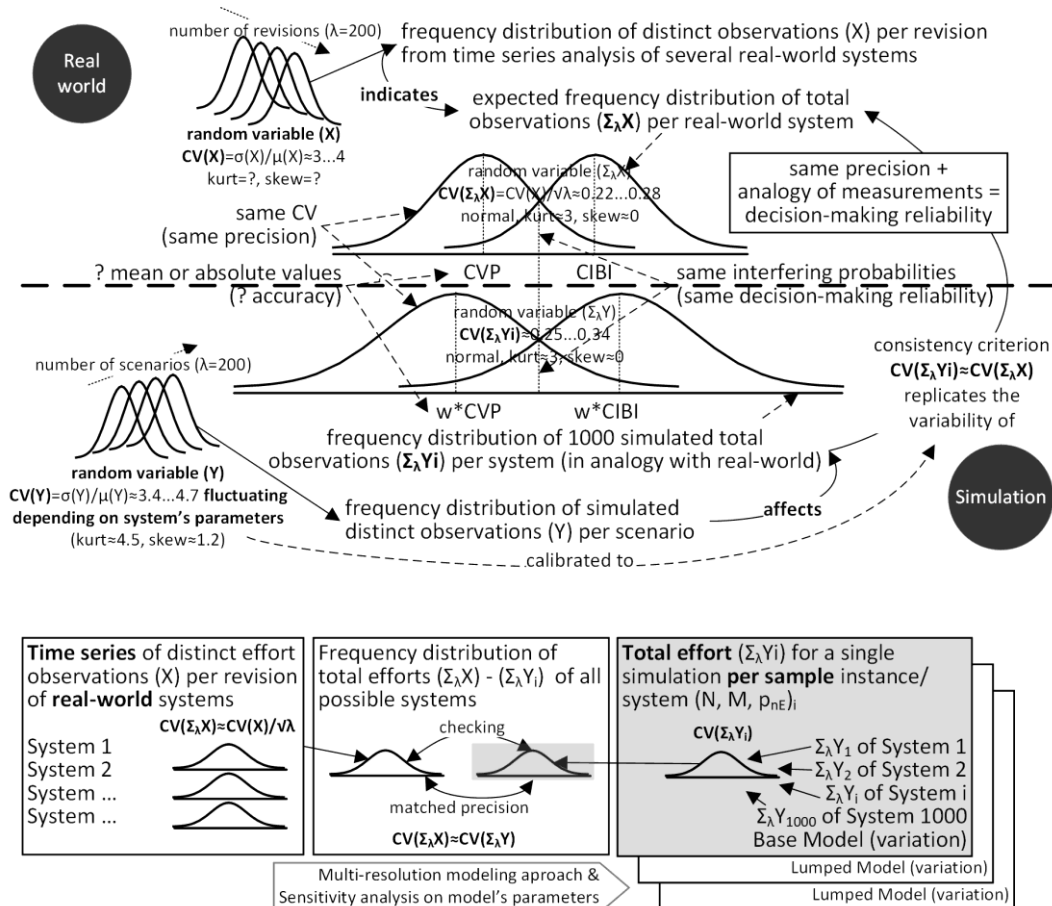


Figure 6-21: Consistency criterion of simulation model's calibration.

The discussion is now focused on the extraction of the CV statistical parameter from real world (effort-based) observations. Toward this direction, frequency distributions of real-world evidence of relevant studies from the field of time series analysis (G. Antoniol et al., 2001; Raja et al., 2009; Shariat Yazdi et al., 2016) are particularly suitable. Considering the random variable X as the possible effort-based assessment per revision during maintenance of real-world systems, the CV(X) parameter expresses the coefficient of variation of its frequency distribution. At this point the characteristics (i.e., skewness, kurtosis) of this frequency distribution are not necessarily known. Based on the statistical theory, the frequency distribution of the total maintenance effort (sum of several- $\lambda$



revisions  $\Sigma_\lambda X$ ) of real-world systems would approximate a normal distribution with  $CV(\Sigma_\lambda X) = CV(X) / \sqrt{\lambda}$  regardless of the characteristics of the initial frequency distribution of  $X$  variable. Thus, the  $CV(\Sigma_\lambda X)$  statistical parameter of real-world (actual effort-based) observations referred to the total maintenance effort can be easily extracted from empirical evidence.

The focus is now on the extraction of the corresponding  $CV$  statistical parameter from simulated (effort-based) observations. Considering the random variable  $Y_i$  as the possible effort-based assessment per applied maintenance scenario during the simulation of the  $i^{\text{th}}$  sample instance, the  $CV(Y_i)$  parameter expresses the coefficient of variation of its frequency distribution. Furthermore, the  $CV(\Sigma_\lambda Y_i^{\text{repeated}})$  parameter expresses the coefficient of variation of the frequency distribution of the total effort assessments (for  $\lambda$  applied scenarios) as extracted by repeated (Monte Carlo) simulations of the same ( $i^{\text{th}}$ ) sample instance. Even if this parameter is tempting, yet it is referred to a particular instance, and thus there some concerns regarding its ability to adequately represent any possible system (sample instance) or the entire design space of the general problem under study. In other words,  $CV(\Sigma_\lambda Y_i^{\text{repeated}})$  parameter is sufficiently informed about the variability of a particular instance of the problem but is insufficient about the overall variability of the general problem as a whole (for any sample instance). Given that the frequency distribution of real-world observations represents an overall esteem concerning several real-world systems, the frequency distribution of simulated observations should represent an overall esteem concerning several instances (systems) of the addressed general problem as well. To address this concern, the frequency distribution of the total efforts ( $\Sigma_\lambda Y_i^{\text{sample}}$ ) as extracted by single (one-time) simulation per sample instance is a more representative and informed parameter. This frequency distribution can be directly extracted by simulated observations while approximates a normal distribution with known  $CV(\Sigma_\lambda Y_i^{\text{sample}})$  as confirmed in Figure 6-16. Thus, the  $CV(\Sigma_\lambda Y_i^{\text{sample}})$  statistical parameter of simulated (effort-based) observations referred to the total maintenance effort can be easily extracted from simulated outcomes.

The argumentation in previous paragraph reviles one of the most important advantages of the introduced modeling and simulation methods. More specifically, the outcomes of the introduced formal and simulation modes do not represent the entire general problem in a universal way. Instead, they are sensitive to several design characteristic (parameters) of the addressed design problem. In other words, they fragment the problem in distinct instances by classifying their outcomes with regard to the parameters of each problem's instance. This differentiation per sample instance is not limited only to the effort assessments but also extends to their variability degree ( $CV$ ). Thus, the variability ( $CV$ ) of the total effort outcomes varies per sample instance. Under this perspective, the extracted variability  $CV(\Sigma_\lambda Y_i^{\text{repeated}})$  from repeated (Monte Carlo) simulations for a specific instance is usually narrower ( $\approx 60\%$  of instances with less than 0.10, and  $\approx 90\%$  of instances with less than 0.20) since the model's outcome are more precise and adapted to the specific design characteristics (parameters) as indicated in Figure 6-18. In fact, there some sample instances for which the variability  $CV(\Sigma_\lambda Y_i^{\text{repeated}})$  is surprisingly wide ( $\approx 3\%$  of instances with more than 0.30) in a range [0.03, ..., 0.51]. The characteristics of the frequency distributions of  $\Sigma_\lambda Y_i^{\text{repeated}}$  variable per sample instance are presented in Appendix C. In contrast, the extracted variability  $CV(\Sigma_\lambda Y_i^{\text{sample}})$  of single (one-time) simulation for all sample instances is wider ( $\approx 0.32$ ) since these outcomes represents the entire design space of the design problem under analysis as indicated in Figure 6-16. Forcing the simulation model to equalize  $CV(\Sigma_\lambda Y_i^{\text{repeated}})$  variability to real-world  $CV(\Sigma_\lambda X)$  variability, ignores and neutralizes the model's capability to adapt its behavior and variability to specific instances of the general problem. Hence, the  $CV(\Sigma_\lambda Y_i^{\text{sample}})$  variability, as a more representative parameter of the entire general problem, is equalized to real-world  $CV(\Sigma_\lambda X)$  variability.

One final concern is whether the real-world observations per revision are in accordance or synchronized to simulated observations per applied maintenance scenario. Thus, whether a revision during maintenance of real-world systems resembles to a maintenance scenario during simulation. In principle, there is no easy way to ensure that real-world observations recorded during the maintenance of different systems between revisions are adequately synchronized in terms of time or activities intervals. Respectively, there is no easy way to ensure that a simulated maintenance scenario is adequately synchronized with a real-world revision in terms of time or activities intervals. Notice that any proportional reduction (smaller time intervals) or increment (larger time intervals) of effort assessments causes the same analogical effect on the  $\sigma$  (standard deviation) and  $\mu$  (mean) parameters of their distributions, thus the  $CV=\sigma/\mu$  parameter remain unchanged. Nevertheless, the main issue is the number of revisions based on which the  $CV(\Sigma_\lambda X)=CV(X)/\sqrt{\lambda}$  of the total real-world efforts is calculated. It has been assumed that this number is equal to the number of applied scenarios ( $\lambda=200$ ) during simulation, but this may not be the case for a particular real-world system. However, due to the law of large numbers and statistical theory, it is expected that in a long-term perspective any desynchronization issues (among number of revisions and number of applied scenarios  $\lambda$ ) will be negligible regarding the dimensionless parameter  $CV$  of the corresponding frequency distributions (i.e.,  $\Sigma_\lambda X$  and  $\Sigma_\lambda Y_i^{sample}$ ). This is a confirmed argument based on the conducted sensitivity analysis. Several trial simulations with deviating values of  $\lambda$  parameter showed that the initial assumption of the consistency criterion is valid.

It is important that  $CV(\Sigma_\lambda X)$  and  $CV(\Sigma_\lambda Y_i^{sample})$  statistical parameters, as dimensionless, are irrelevant of absolute effort values. Thus, the accuracy of measurements is not of primary interest. Based on previous analysis, the consistency criterion is mathematically expressed by matching the values of  $CV(\Sigma_\lambda X)$  and  $CV(\Sigma_\lambda Y_i^{sample})$  statistical parameters. Under this equality, the statistical parameters of the real-world and simulated observations are interrelated as analyzed in Table 6-5. This criterion verifies the consistency of the simulation model against real-world circumstances concerning its variability degree and decision-making reliability.

Table 6-5: Statistical parameters of real-world against simulated effort-based observations

Random – stochastic variable	Shape	Statistical parameters of frequency distribution per variable
Referring to the random <b>variable X</b> as the possible <u>intermediate effort-based assessment</u> per revision during maintenance of real-world systems, extracted through time series analysis	Unknown OR Right-Skewed Normal	$\mu(X)$ $\sigma(X)$ $CV(X) = \sigma(X)/\mu(X)$
Referring to the possible <u>total effort-based assessment</u> $\Sigma_\lambda X$ of several ( $\lambda$ ) revisions during maintenance of real-world systems	Normal	$\mu(\Sigma_\lambda X) = \lambda \cdot \mu(X)$ $\sigma(\Sigma_\lambda X) = \lambda \cdot \sigma(X)/\sqrt{\lambda}$ $CV(\Sigma_\lambda X) = \sigma(\Sigma_\lambda X)/\mu(\Sigma_\lambda X) = \lambda \cdot \sigma(X)/\sqrt{\lambda} / \lambda \cdot \mu(X) = \sigma(X)/\sqrt{\lambda} / \mu(X) = CV(X)/\sqrt{\lambda}$
Referring to the random <b>variable Y<sub>i</sub></b> as the possible <u>intermediate effort-based assessment</u> per applied maintenance scenario during the simulation of the $i^{th}$ sample instance	Right-Skewed Normal	$\mu(Y_i) = \mu(w \cdot X) = w \cdot \mu(X)$ given the analogy of measurements $\sigma(Y_i) = \sigma(w \cdot X) = w \cdot \sigma(X)$ $CV(Y_i) = \sigma(Y_i)/\mu(Y_i) = w \cdot \sigma(X)/w \cdot \mu(X) = \sigma(X)/\mu(X) = CV(X)$ is the coefficient of variation
Referring to the <u>total effort-based assessment</u> $\Sigma_\lambda Y_i^{sample}$ of several ( $\lambda$ ) applied scenarios during simulation of single (one-time) simulation per sample instance	Normal	$\mu(\Sigma_\lambda Y_i^{sample}) = \lambda \cdot \mu(Y_i^{sample}) = \lambda \cdot w \cdot \mu(X)$ $\sigma(\Sigma_\lambda Y_i^{sample}) = \lambda \cdot \sigma(Y_i^{sample})/\sqrt{\lambda} = \lambda \cdot w \cdot \sigma(X)/\sqrt{\lambda}$ $CV(\Sigma_\lambda Y_i^{sample}) = \sigma(\Sigma_\lambda Y_i^{sample})/\mu(\Sigma_\lambda Y_i^{sample}) = \lambda \cdot w \cdot \sigma(X)/\sqrt{\lambda} / \lambda \cdot w \cdot \mu(X) = \sigma(X)/\sqrt{\lambda} / \mu(X) = CV(X)/\sqrt{\lambda} = CV(\Sigma_\lambda X)$

$\mu$ : mean value,  $\sigma$ : standard deviation,  $CV=\sigma/\mu$ : coefficient of variation (dimensionless)

$\lambda=200$  number of applied maintenance scenarios during simulation or revisions during maintenance of real-world systems

sample=1000 instances of the general design problem as defined by their design attributes and scenarios probabilities

$w$ : constant factor representing the analogy of measurements (not necessarily known)

#### 6.4.9.4 Overall Statistical Parameters of the Sample Instances

In this subsection, an overall (graphical) assessment of the statistical parameters of the  $\Sigma_\lambda Y_i^{repeated}$  and  $Y_i$  variables concerning all the sample instances ( $i:[1, \dots, 1000]$ ) of CVP vs CIBI problem is presented. The analysis concentrates on the characteristics ( $CV$ , skewness,

and kurtosis) of the frequency distributions of  $\Sigma_{\lambda}Y_i^{\text{repeated}}$  and  $Y_i$  variables as they numerically presented per sample instance in Appendix C.

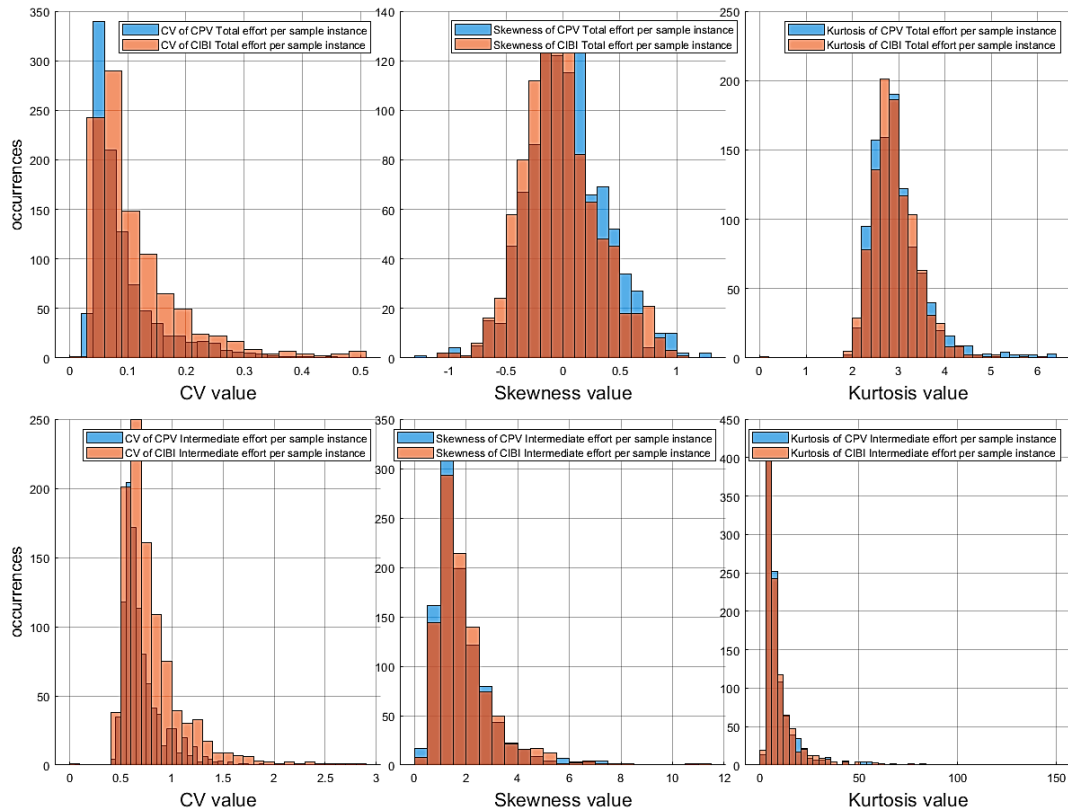


Figure 6-22: Overall assessment of the statistical parameters of the  $\Sigma_{\lambda}Y_i^{\text{repeated}}$  and  $Y_i$  variables concerning all the sample instances.

Considering  $Y_i$  variable as the intermediate required effort per applied scenario during the simulation of the  $i^{\text{th}}$  sample instance, the frequency distribution of this variable represents the statistical pattern of intermediate effort assessments for this particular sample instance. The parameters ( $\mu$ ,  $\sigma$ , CV, skewness, kurtosis) of such frequency distributions per sample instance are presented in Appendix C. Each  $Y_i$  distribution is the result of a single (one time) simulation per sample instance. Figure 6-22 (2<sup>nd</sup> row) presents the overall distribution per statistical parameter (CV, skewness, kurtosis) concerning all the sample instances. Since the intermediate effort outcome ( $Y_i$ ) is a heavily stochastic variable expressing the required effort per applied scenario, all the parameters of its frequency distribution variate significantly among different sample instances. More specifically, CV lies between 0.43 and 2.88, skewness between 0.16 and 11.43, and kurtosis between 2.31 and 149.62. Thus, depending on the design characteristics of each sample instance the variability of  $Y_i$  variable variates significantly. Notice that all  $Y_i$  distributions are right skewed and most of them with high kurtosis.

Considering  $\Sigma_{\lambda}Y_i$  variable as the total required effort during the simulation of the  $i^{\text{th}}$  sample instance, the frequency distribution of this variable represents the statistical pattern of total effort assessments for this particular sample instance. The parameters ( $\mu$ ,  $\sigma$ , CV, skewness, kurtosis) of such frequency distributions per sample instance are presented in Appendix C. Each  $\Sigma_{\lambda}Y_i$  distribution is the result of several repeated (Monte Carlo) simulations for the same sample instance. Figure 6-22 (1<sup>st</sup> row) presents the overall distribution per statistical parameter (CV, skewness, kurtosis) concerning all the sample instances. Since the total effort outcome ( $\Sigma_{\lambda}Y_i$ ) is the sum of several ( $\lambda=200$ ) intermediate effort assessments ( $Y_i$ ) per applied scenario, its frequency distribution follows (or

resembles) a normal distribution pattern with narrower variation (than  $Y_i$  variable) for all the sample instances. This is the result of central limit theorem’s properties according to which  $CV(\Sigma_\lambda Y_i) = CV(Y_i) / \sqrt{\lambda}$ . Thus, its skewness for all sample instances lies around 0 [-1.09, ..., 1.26] and the kurtosis around 3 [1.90, ..., 6.06], while CV lies between 0.03 and 0.51. Yet, the CV parameter considerably variates per sample instance with  $\approx 60\%$  of instances have a CV less than 0.10,  $\approx 90\%$  of instances have a CV less than 0.20, and  $\approx 3\%$  of instances have a CV more than 0.30.

#### 6.4.9.5 Calibration Process (Flow Chart)

The entire process of simulation model’s calibration based on the multi-resolution modeling approach is provided in Listing 6-1 in the form of pseudocode. This code presents the flow-chart of all the repeated sub-activities including the imposed requirements and assumptions, giving emphasis on the required controls concerning the fulfilment of the consistency criterion. Notice that sub-activities like add or change assumptions and build or change a model’s variation are very creative and intelligent tasks that require human capabilities and skills. Such activities are strongly linked to the specifications and characteristics of the addressed design problem and the used design patterns. Furthermore, the presented process holds a copy of all the intermediate (lumped) variations of the simulation model as each assumption is removed and corresponding requirement is satisfied.

Listing 6-1: Pseudocode of simulation model calibration based on multi-resolution modeling

```

81.  $\lambda = 200$  % number of applied scenarios or revisions during maintenance
82.  $CV(X) = 4$  % coefficient of variation of distinct observations (X) per revision from time series analysis of several real-world systems
83.  $CV(\Sigma_\lambda X) = CV(X) / \sqrt{\lambda}$  % expected coefficient of variation (variability) of total observations ( $\Sigma_\lambda X$ ) per real-world system
84.
85. Add all requirements to Requirements[] set % reflecting the underlying activities and/or desired stochastic behavior of the problem
86. For each requirement in Requirements[]
87.     Add corresponding assumption(s) in Assumptions[] set % exclude underlying activities and/or desired stochastic behavior
88. End For
89.
90. variation=1 % initialization of current variation of simulation model
91. Build SM(variation) based on all Assumptions[] % build initial lumped model that fulfills all the imposed assumptions
92. Repeat
93.     If Assumptions[] is NOT empty Then
94.         Pick a current_assumption from Assumptions[] % for analysis
95.         Change SM(variation) % to satisfy the corresponding requirement towards higher resolution or stochastic behavior
96.     Else
97.         Pick a requirement from Requirements[] % for refinement of consistency
98.         Change SM(variation) % by calibrating its distinct outcomes (Y) to achieve higher consistency (variability)
99.     End If
100. For instance=[1:1000] % referring to each sample instance (system) of the general problem
101.     Total_Effort(instance) = SM(variation, instance,  $\lambda$ ) % runs simulation, returning total effort ( $\Sigma_\lambda Y_i$ ) per system/instance
102. End For
103.  $CV(\Sigma_\lambda Y_i, \text{variation}) = CV(\text{Total\_Effort}(:))$  % calculates the overall  $CV(\Sigma_\lambda Y_i)$  of all sample instances of the general problem
104. If  $Abs(CV(\Sigma_\lambda X) - CV(\Sigma_\lambda Y_i, \text{variation})) < Abs(CV(\Sigma_\lambda X) - CV(\Sigma_\lambda Y_i, \text{variation}-1))$  Then % higher consistency from previous variation
105.     Remove current_assumption (if exist) from Assumptions[]
106.     Flag the corresponding requirement to Requirements[] % that neutralizes the current assumption
107.     variation += 1 % increases variation
108.     Copy SM(variation-1) to SM(variation) % copying the previous model’s variation to a new variation
109. Else % lower consistency from previous variation
110.     Change current_assumption or Add alternate assumption (if exist) to Assumptions[]
111.     Discard changes made in SM(variation) % resets current model’s variation
112. End If
113. Until (Assumptions[] is empty) AND ( $CV(\Sigma_\lambda X) \approx CV(\Sigma_\lambda Y_i, \text{variation})$ ) % all requirements have been satisfied with high consistency
114. Return SM(variation) % as the base model

```

#### 6.4.9.6 Times Series Analysis of Simulated Effort Assessments

In this subsection, an indicative analysis of the simulated effort assessments per applied scenario from the perspective of time series analysis is attempted. Again,  $Y_i$  variable represents the intermediate required effort per applied scenario during the simulation of the  $i^{\text{th}}$  sample instance. The analysis focus on a (single) simulation of the sample instance N.002 with parameters  $N=56$ ,  $M=90$ ,  $p_{nE}=0.32$ ,  $p_{nP}=0.68$ , while examines the effort

assessments of CVP design alternative for a sequence (or time series) of  $\lambda=200*20=4000$  applied scenarios. Usually, time series analysis requires long time series of several values to be effective and consistent, thus a larger number of applied scenarios ( $\lambda=4000$ ) has been selected even if a such value is not so realistic under real-world circumstances.

In time series analysis, each value is expressed or predicted based on the values of previous instances of the time series. To conclude on a model able to predict future values based on previous values, time series analysis examines two separate aspects: a) autoregressive (AR) analysis which is similar to regular regression where the dependent variable (current value  $y(\lambda)$ ) is predicted based on a number of independent variables (previous values i.e.,  $y(\lambda-1)$ ,  $y(\lambda-2)$ , etc.) of the same time series, and b) moving average (MA) analysis which tries to predict the current level of noise ( $\varepsilon$ ) based on its current and previous levels (i.e.,  $\varepsilon(\lambda)$ ,  $\varepsilon(\lambda-1)$ ,  $\varepsilon(\lambda-2)$ , etc.).

An autoregressive (AR) model predicts the current value based on a linear combination of past values of the same time series of values. Normally, autoregressive models are applied to stationary time series only. Mathematically, an AR(p) model is expressed as follows:

$$y_t = c + \varphi_1 y_{t-1} + \varphi_2 y_{t-2} + \dots + \varphi_p y_{t-p} + \varepsilon_t$$

Where p: is the order of regression, c: is a constant, epsilon( $\varepsilon$ ): noise, and t:time (or the number applied scenarios  $\lambda$ ).

A moving average (MA) model predicts the effect of noise ( $\varepsilon$ ) based on a linear combination of past noise levels of the same time series. Mathematically, an MA(q) model is expressed as follows:

$$y_t = c + \theta_1 \varepsilon_{t-1} + \theta_2 \varepsilon_{t-2} + \dots + \theta_q \varepsilon_{t-q} + \varepsilon_t$$

Where p: is the order, c: is a constant, epsilon( $\varepsilon$ ): noise, and t:time (or the number applied scenarios  $\lambda$ ).

An ARMA(p,q) model is simply the combination of both models into the following single equation:

$$y_t = c + \varphi_1 y_{t-1} + \varphi_2 y_{t-2} + \dots + \varphi_p y_{t-p} + \theta_1 \varepsilon_{t-1} + \theta_2 \varepsilon_{t-2} + \dots + \theta_q \varepsilon_{t-q} + \varepsilon_t$$

ARMA(p,q) model tries to explain the relationship of a time series with both a) random noise (moving average part) and b) itself at a previous step (autoregressive part). Given a time series of values under analysis, the estimation of the weights (i.e.,  $\theta$  and  $\varphi$  factors) of the ARMA(p,q) model can be performed only for specific (predefined) orders or lags (i.e., p and q). Furthermore, the estimation of ARMA model can be assisted by statistical tools such as the MATLAB environment and its Econometric Modeler.

Concentrating on the time series of the total simulated effort assessments per applied scenario of the sample N.002, an indicative ARMA (2,3) model is visualized in Figure 6-23. There is both a trend and perhaps a change in variance in this time series, thus there is a clear indication of non-stationarity behavior. More specifically:

- KPSS (Kwiatkowski–Phillips–Schmidt–Shin) test for trend stationary confirms this inference by rejecting Null hypothesis of trend stationary (CL=0.05, p\_val=0.010).
- ADF (Augmented Dickey–Fuller) test for unit root (non-stationary) confirms this inference by not rejecting Null hypothesis of non-stationary (CL=0.05, p\_val=0.9990 for lag =[1..10]).

Since autoregressive models are normally applied to stationary time series only, the time series need to be transformed to resemble a stationary behavior.

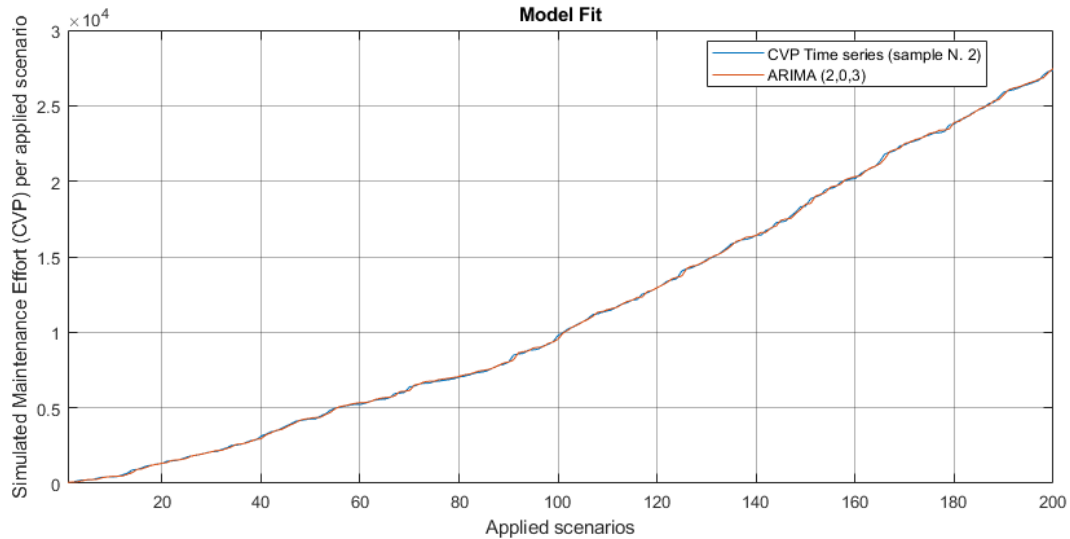


Figure 6-23: Time series analysis (ARIMA) of total CVP effort assessments

ARIMA model stands for Auto Regressive Integrated Moving Average. This model combines the autoregression model, the moving average model, and differencing. Under this perspective, integration is the opposite of differentiation. Differencing is suitable to remove the trend and transform a time series to stationary. It simply involves subtracting each value in  $t-1$  from time  $t$ . Mathematically, an ARIMA( $p,d,q$ ) model is expressed by the following equation:

$$y'_t = c + \varphi_1 y'_{t-1} + \varphi_2 y'_{t-2} + \dots + \varphi_p y'_{t-p} + \theta_1 \varepsilon_{t-1} + \theta_2 \varepsilon_{t-2} + \dots + \theta_q \varepsilon_{t-q} + \varepsilon_t$$

Where  $d$ : is the degree of differencing (number of times it was differenced)

By differentiating (transforming) the time series of the total simulated effort assessments, the intermediate effort assessments per applied scenario of the sample N.002 are derived. An indicative ARIMA (2,1,3) model of the transformed time series is visualized in Figure 6-24. However, there is still a trend in this time series, thus there is an indication of non-stationarity behavior. More specifically:

- KPSS (Kwiatkowski–Phillips–Schmidt–Shin) test for trend stationary confirms this inference by rejecting Null hypothesis of trend stationary (CL=0.05,  $p\_val=0.010$ ).
- Nevertheless, ADF (Augmented Dickey–Fuller) test for unit root (non-stationary) does not confirm this inference by rejecting Null hypothesis of non-stationary (CL=0.05,  $p\_val=0.0010$  for lag = [1..10]).

Since autoregressive models are normally applied to stationary time series only, the time series need to be further transformed to resemble a stationary behavior.

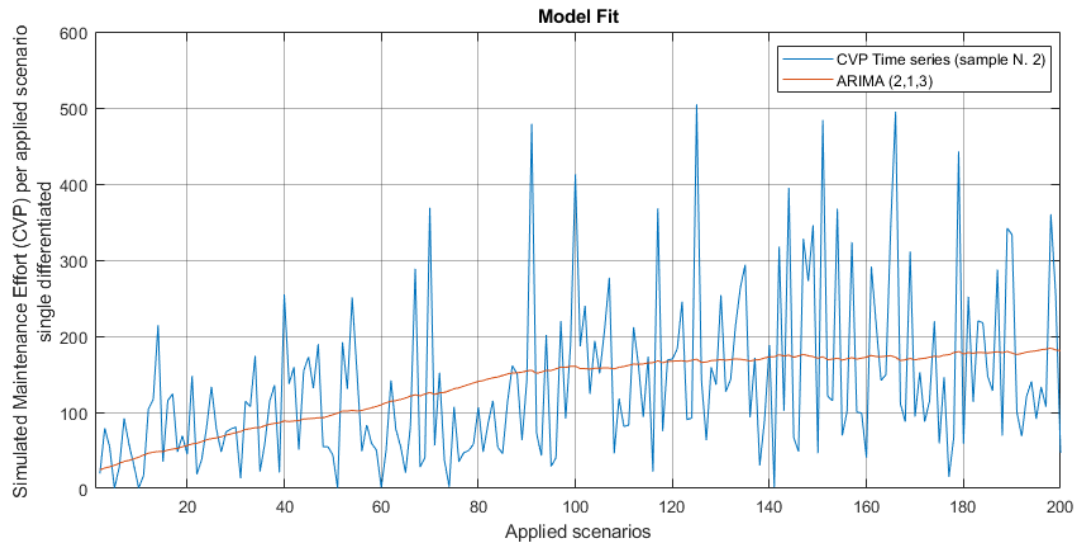


Figure 6-24: Time series analysis (ARIMA) of (single differentiated) intermediate CVP effort assessments

By double differentiating (transforming) the time series of the total simulated effort assessments, the marginal changes among intermediate effort assessments per applied scenario of the sample N.002 are derived. An indicative ARIMA (2,2,3) model of the transformed time series is visualized in Figure 6-25. This time, there is no trend nor significant change in variance in this time series, thus there is an indication of stationarity behavior. More specifically:

- KPSS (Kwiatkowski–Phillips–Schmidt–Shin) test for trend stationary confirms this inference by not rejecting Null hypothesis of trend stationary (CL=0.05, p\_val=0.100).
- ADF (Augmented Dickey–Fuller) test for unit root (non-stationary) confirm this inference by rejecting Null hypothesis of non-stationary (CL=0.05, p\_val=0.0010 for lag =[1..10]).

Furthermore, the residual plot is presented in Figure 6-25. At this point the ARIMA (2,2,3) model is rather weak since its residuals are significant (of high values) and disbalanced around zero level.

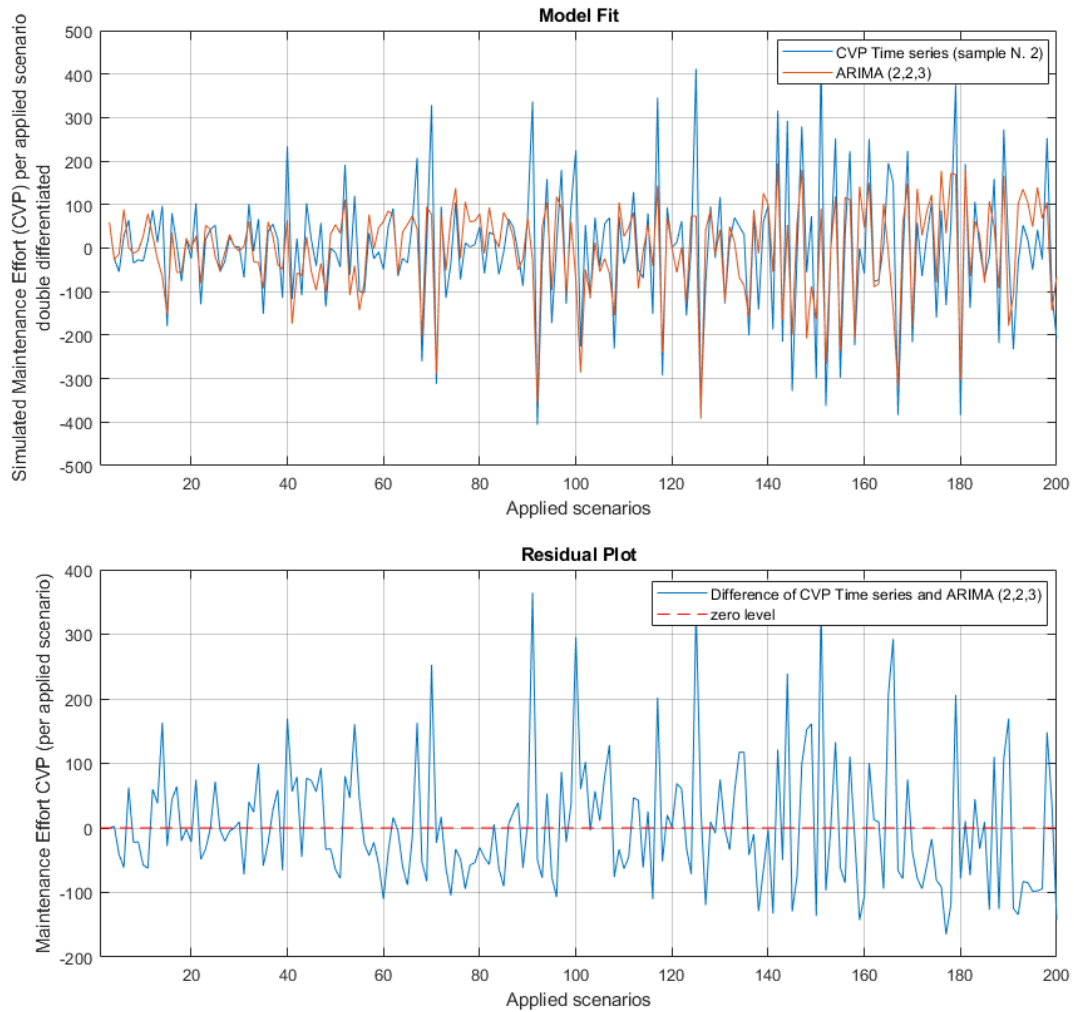


Figure 6-25: Time series analysis (ARIMA) of (double differentiated) intermediate CVP effort assessments (simulation of sample instance N. 002)

To conclude on the most consistent ARIMA model, different combinations of orders (i.e.,  $p$  and  $q$ ) should be tried, next fit each ARIMA model with those orders, and use a criterion for the selection of proper combination of orders. More specifically, the criterion of Akaike's Information Criterion (AIC) is suitable for selecting the order  $(p,d,q)$  of an ARIMA model. In practice, the model with the lowest AIC compared to other models is the most consistent and, thus preferable. AIC is a criterion (goodness of fit) only for comparison purposes relative to other models. It is possible more parameters (higher  $p$  and  $q$  orders) to increase the predictability of the model, however more parameters will increase the AIC score and thus penalize the model. Hence, AIC is suitable to discover the ARIMA model with the fewer number of parameters that still provide good results. In the context of the simulated effort assessments, AIC is used for a constant order of differencing ( $d=2$ ).

By analyzing a time series (simulated effort assessments) of increased length (4000 applied scenarios) in MATLAB, the ARIMA(0,2,1) model with orders  $p=0$  and  $q=1$  demonstrating the lowest AIC compared to other combinations of orders (i.e.,  $p$  and  $q$ ) has been selected. Even if the ACF (Auto Correlation Function) and PACF (Partial Auto Correlation Function) cannot be used to identify reliable values for  $p$  and  $q$  orders, in the case of an ARIMA(0, $d$ , $q$ ) process: a) the PACF is exponentially decaying or sinusoidal, and b) the ACF has a significant spike at lag  $q$  (in this case  $q=1$ ) but none after, as confirmed by the evidence in Figure 6-26.



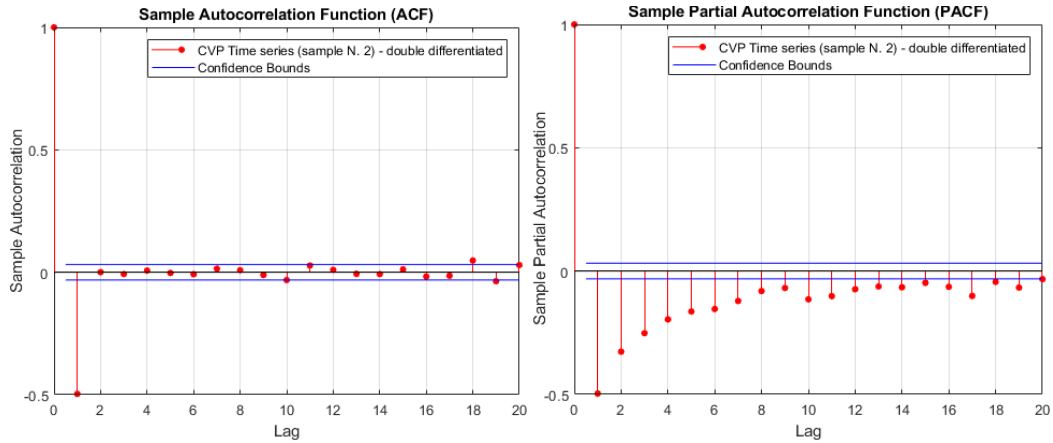


Figure 6-26: ACF and PACF diagrams of (double differentiated) effort time series

An indicative ARIMA (0,2,1) model of the transformed (double differentiated) time series of 4000 values, escorted by the residual plot, is visualized in Figure 6-27. The statistical parameters of this model are provided in Table 6-6. Again, the ARIMA (0,2,1) model is rather weak since its residuals are significant (of high values) and disbalanced around zero level (especially after the first 1000 applied scenarios). Alternatively, the ARIMA model seems that fails to adequately predict the positive and peak values of the time series leading to increased and disbalanced residuals (especially after 100 applied scenarios). However, there is no autocorrelation evidence as confirmed by the Ljung-Box Q-Test for autocorrelation ( $H_0$ : of zero 20lags autocorrelation is not rejected [CL=0.05, p\_val=0.2158]), thus residual values correspond to pure (white) noise. This (white) noise is the result of the random and stochastic behavior of the simulation model. Similar ARIMA models can be estimated for alternate time series for different set of parameters (i.e., N, M,  $p_{nE}$ ) of sample instances, however with different values in their parameters (i.e., c,  $\epsilon$ , and AR{1}).

Table 6-6: Statistical Parameters of ARIMA Analysis of Simulated Effort Assessments

Parameter	Value	Standard Error	T Statistic	P-Value
Constant	0.4884	0.1404	3.4776	5.0595e-004
MA{1}	-0.9938	0.0014	-697.1537	0
Variance	9.1196e+05	1.0421e+04	87.5109	0

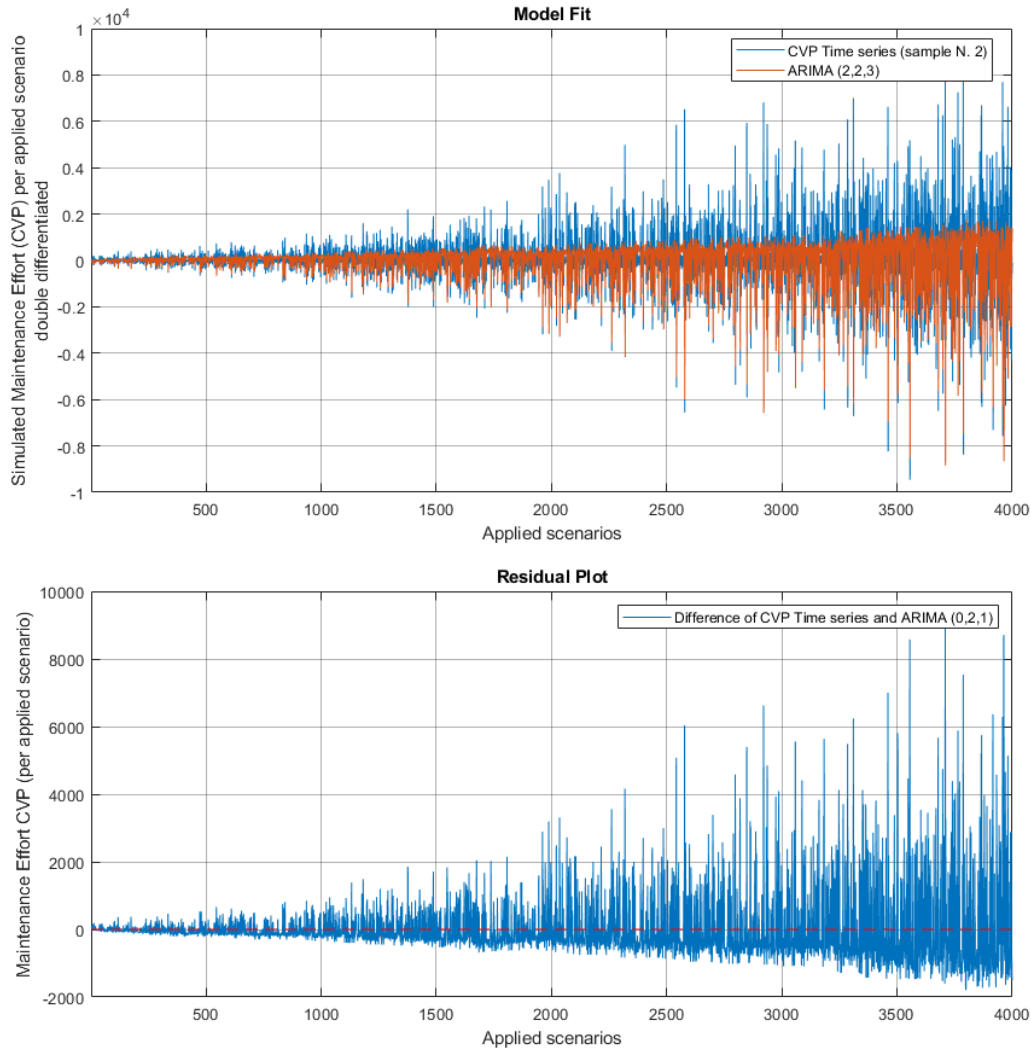


Figure 6-27: Time series analysis ARIMA(0,2,1) of (double differentiated) intermediate CVP effort assessments (simulation of sample instance N. 002)

An ARIMA model with order  $p=0$  for its autoregressive part indicates that previous values provide no information about the future values of time series. The only relation concerns the previous and current levels of noise (i.e.,  $\varepsilon(t-1)$  and  $\varepsilon(t)$ ) as captured by the moving average part of order  $q=1$ . This evidence led us to the following interesting inference. Without the engagement of exogenous factors (e.g.,  $N$ ,  $M$ ,  $p_{NE}$ ) related to the nature of the confronted problem, the time series analysis is unable to conclude on ARIMA models of high consistency by only relying on previous values of the time series. Moreover, this evidence confirms the complicated and heavily stochastic nature of actual maintenance process as successfully approached by the introduced simulation model. From time series perspective, the maintenance process (and relevant effort-based measurements) remains a black-box without a practical envision or even a sense of the phenomenon under study.

## 6.5 Results & Inferences

### 6.5.1 Analysis and Interpretation

The overall evidence of the analysis per simulation state are summarized in Table 6-7 and analyzed next. Indicative graphs of the final and fully stochastic 7<sup>th</sup> simulation state are presented in this subsection.

*External correlation control:* Indicative scatter diagrams of formal model’s predictions ( $c_m$ ) against simulation model’s computations ( $sc_m$ ) regarding effort assessments of CVP, CIBI, and their difference, are presented in Figure 6-28. The number of scenario application emphasizes on early ( $\lambda=10$ ), mid ( $\lambda=100$ ), and long ( $\lambda=200$ ) term perspectives. Respectively, the correlation coefficient regarding effort assessments of CVP, CIBI, and CVP-CIBI difference against the number of scenario applications ( $\lambda$ ) is graphically presented in Figure 6-29 (left side).

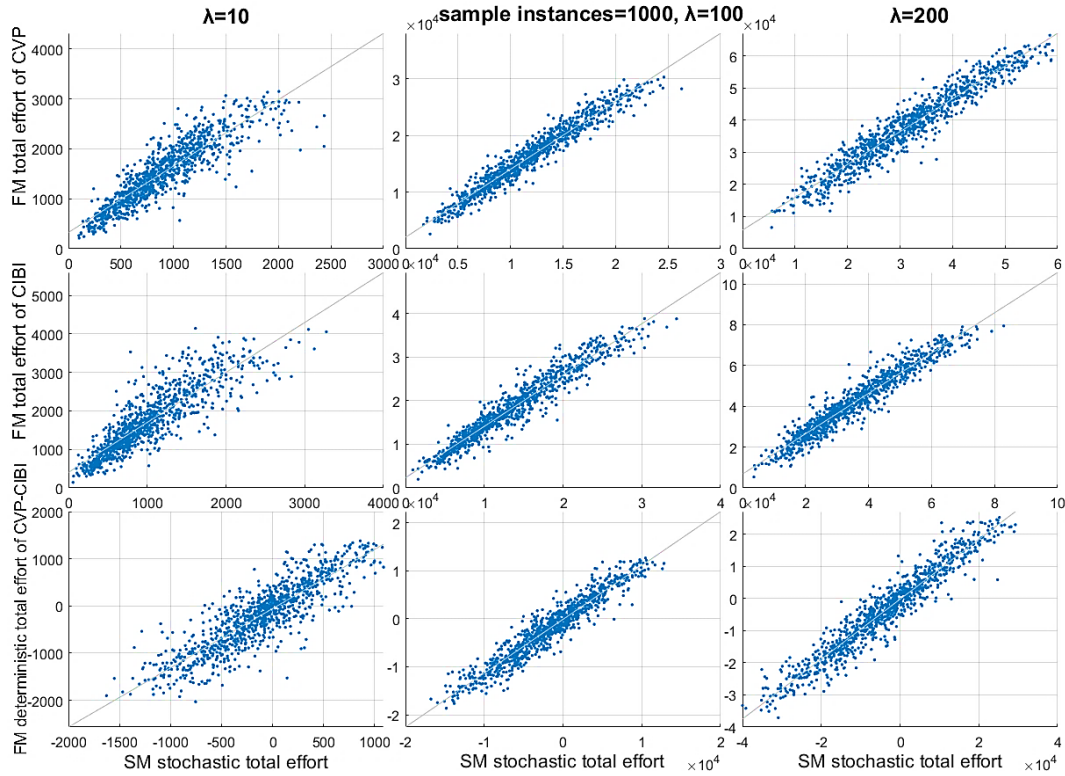


Figure 6-28: Scatter diagrams of external correlation among formal and simulated values of total effort (CVP, CIBI, CVP-CIBI), concerning all (1000) sample instances, of the 7<sup>th</sup> fully stochastic simulation state in Table 6-4.

*Hypothesis testing control:* The T-Test results for the CVP-CIBI distance’s crucial values against the number of scenario applications ( $\lambda$ ) are presented in Figure 6-29 (right side).

*Error rate control and critical error analysis:* Figure 6-29 (middle) summarizes the results concerning the overall average error rates ( $Er$ ) for different  $\lambda$  values. In addition, the critical errors with high impact in terms of wasted effort and high possibility to occur are further investigated. For this purpose, the critical error is defined:

*Critical Error definition:* A critical error ( $cEr$ ) occurs when, for a specific experiment scenario, a) the rate of average wasted effort (difference) against the minimum average required effort among design alternatives is greater than 10%, and b) the Error rate  $Er(\lambda)$  of this experiment scenario is greater than the average  $Er(\lambda)$  of all experiment scenarios for each specific  $\lambda$  value. Thus, critical are those errors with high severity degree and likelihood of occurrence.

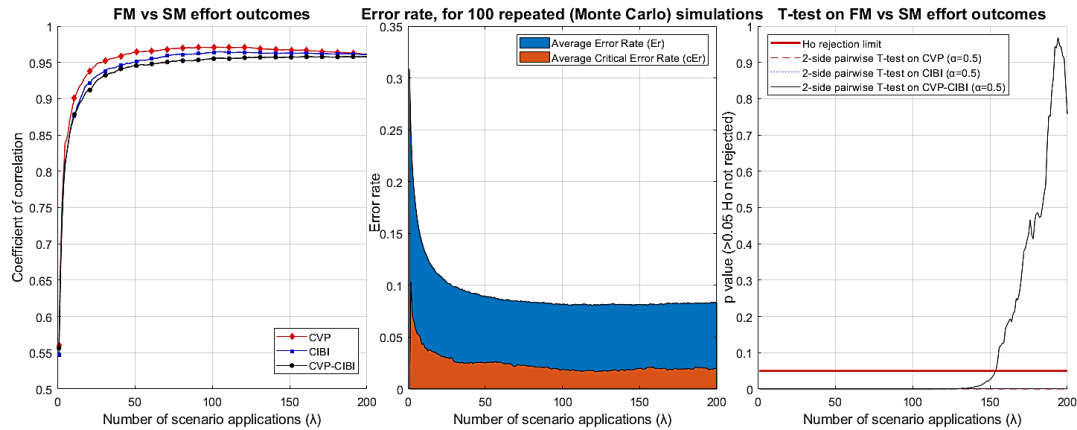


Figure 6-29: Overall control diagrams (left):coefficient of correlation, (right):hypothesis T-test, (mid):average error rate & critical error rate, concerning 1000 sample instances, of the 7<sup>th</sup> fully stochastic simulation state in Table 6-4.

*Time orientation of analysis:* The analysis for short-term system maintenance (e.g.,  $\lambda < 10$ ) does not provide any particular benefit since the importance of the estimated gain or loss in terms of effort would be negligible. Therefore, the analysis of inferences is concentrated in a mid-to-long maintenance period, thus for  $\lambda > 50$  up to 200.

*Value orientation of analysis:* The analysis mainly focuses on the difference between design alternatives for Formal and Simulated effort assessments [ $c_m(\text{CVP}) - c_m(\text{CIBI})$  and  $sc_m(\text{CVP}) - sc_m(\text{CIBI})$ ] since these values define the selection-making process.

Since formal and simulation models are focused on reliable decision-making among design alternatives, the precision of their measurements (i.e., correlation, error rate, variability, standard deviation, coefficient of variation, kurtosis, and skewness) is of primary importance, while the accuracy of their measurements (i.e., absolute values, and mean) are less important. In principle, higher correlation combined with lower average Er and cEr suggests higher selection reliability of formal model and vice versa. Finally, when p-value of hypothesis testing is above the rejection limit, the total effort/size predictions of formal model are highly precise in terms of absolute values.

### 6.5.2 Inference Extraction

In this subsection, the experiment’s evidence is gradually analyzed per simulation state, and summarized in Table 6-7. Indicative graphs of the 1<sup>st</sup> to 6<sup>th</sup> simulation states are presented in this subsection.

Table 6-7: Overall Control Evidence for all Experiment Scenarios and Simulation Model States

Simulation Model state / Gradually Engaged factor	External correlation <sup>1,3</sup> of CVP-CIBI distance ( $\lambda > 50$ to 200)	Hypothesis control <sup>2,3</sup> on the difference CVP-CIBI (T-test, 2tailed, paired), $CL=0.95 / \alpha=0.05$		Error rate <sup>1,4</sup> control ( $\lambda > 50$ to 200)		FM’s reliability (mid-to-long-term) control on the difference CVP-CIBI	
	Coefficient of correlation (r)	p-value	H <sub>0</sub> rejection	Average (Er)	Critical (cEr)	Selection <sub>1</sub>	Absolute values <sup>2</sup>
1. Variable scenario sequences	0.98 $\nearrow$ 1.00	$> 0.20$	No	5.5% $\searrow$ 3.4%	1.0% $\searrow$ 0.0%	High	High
2. Shifting scenarios probabilities	0.97 $\nearrow$ 0.99	$> 0.50$	No	6.4% $\searrow$ 5.1%	1.2% $\searrow$ 0.6%	High	High
3. Alternate maintenance scenarios	0.97 $\nearrow$ 0.99	$> 0.10$	No	6.4% $\searrow$ 4.8%	1.3% $\searrow$ 0.8%	High	High
4. Variable interventions’ size	0.97 $\nearrow$ 0.98	$> 0.20$	No	6.7% $\searrow$ 4.8%	1.6% $\searrow$ 0.9%	High	High
5. Code aging & learning rate	0.76 $\nearrow$ 0.78	$\approx 0$	Yes	6.5% $\searrow$ 5.2%	1.5% $\searrow$ 0.8%	Medium	Low
6. Variable developers’ experience level	0.97 $\nearrow$ 0.98	$> 0.05$ ( $\lambda > 170$ )	Yes/No	6.7% $\searrow$ 5.4%	1.4% $\searrow$ 0.8%	High	Low
7. Highly shifting scenarios probabilities	0.94 $\nearrow$ 0.96	$> 0.05$ ( $\lambda > 150$ )	Yes/No	9.0% $\searrow$ 8.1%	2.5% $\searrow$ 2.0%	High	Medium

↗ slightly increased/decreased to, <sup>1</sup>decision-making reliability, <sup>2</sup>accuracy reliability, <sup>3</sup>single simulation per sample instance, <sup>4</sup>repeated (Monte Carlo) simulations per sample instance

1. In this initial state, most of the random behavior of the simulation model has been disabled. The only stochastic behavior is the random selection of maintenance scenario types based on their individual probabilities, thus generating variant sequences of scenario applications. These results provide sufficient evidence that the implied continuous integration, used by the formal models, converges to the discrete calculations of the actual maintenance process as performed by the simulation model. Furthermore, the repeated scenario sequences, assumed by the formal models, are in accordance with the random-pattern sequences of the actual maintenance process.

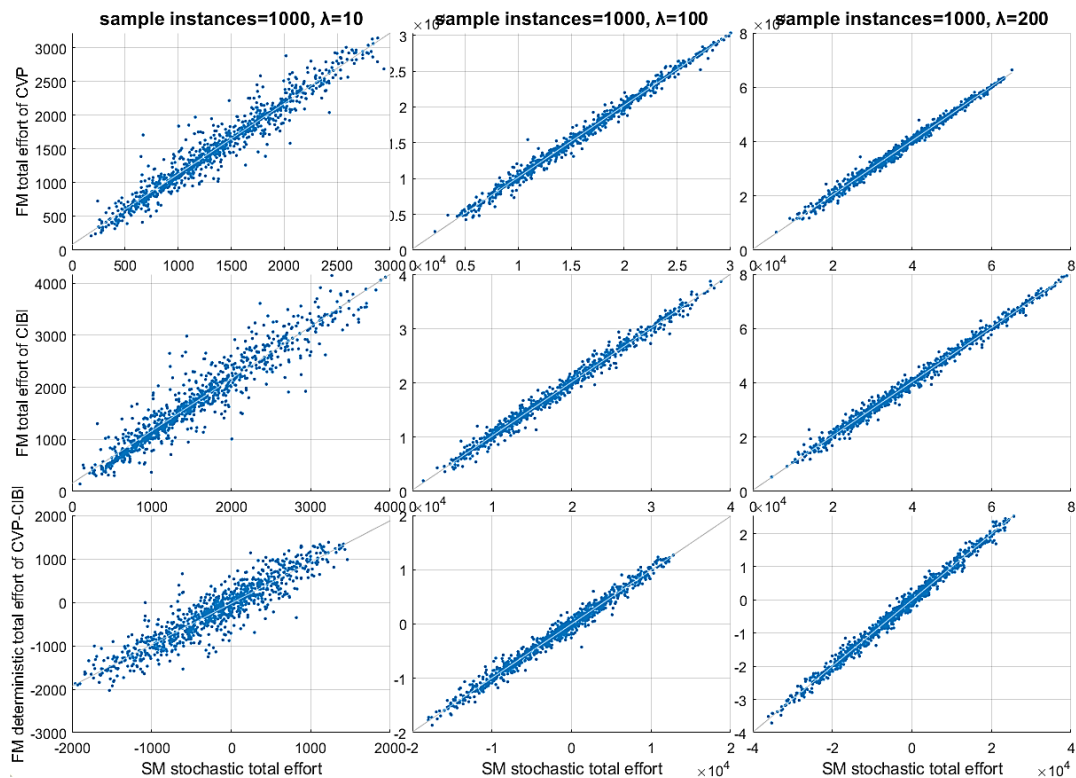


Figure 6-30: Scatter diagrams of external correlation among formal and simulated values of total effort (CVP, CIBI, CVP-CIBI), concerning all (1000) sample instances, of the 1<sup>st</sup> simulation state in Table 6-4.

As indicated by the diagrams in Figure 6-44 and Figure 6-45, the long-term coefficient of correlation is almost perfect (approaching to 1.0), while the critical error rate fades to 0% in a long-term perspective. In addition, according to T-rest results, there is high accuracy in terms of absolute effort assessments concerning their difference of CVP minus CIBI.

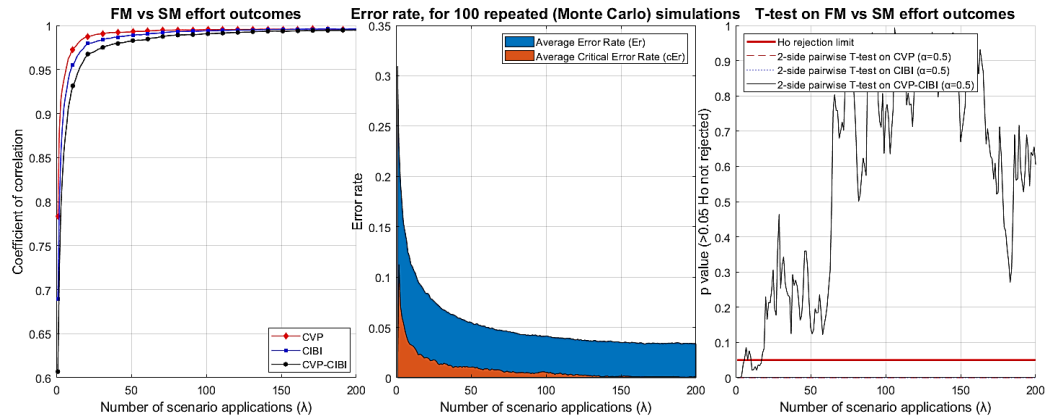


Figure 6-31: Overall control diagrams (left):coefficient of correlation, (right):hypothesis T-test, (mid):average error rate & critical error rate, concerning 1000 sample instances, of the 1<sup>st</sup> simulation state in Table 6-4.

2. In this state, the uncertainty factor effecting scenarios' probabilities during simulation is engaged. The value of  $f_{BM}$  factor is set to 0.5 implying a medium uncertainty level. Thus, the overall uncertainty factor  $f_{BM} \times u_f$  ranges in the  $\pm 3\sigma$  interval of  $(-0.15, \dots, 0.15)$ . These results provide sufficient evidence that the repeated scenario sequences based on constant probabilities, assumed by formal models, approximate the medium uncertainty level imposed by shifting scenarios' probabilities during the actual maintenance process.

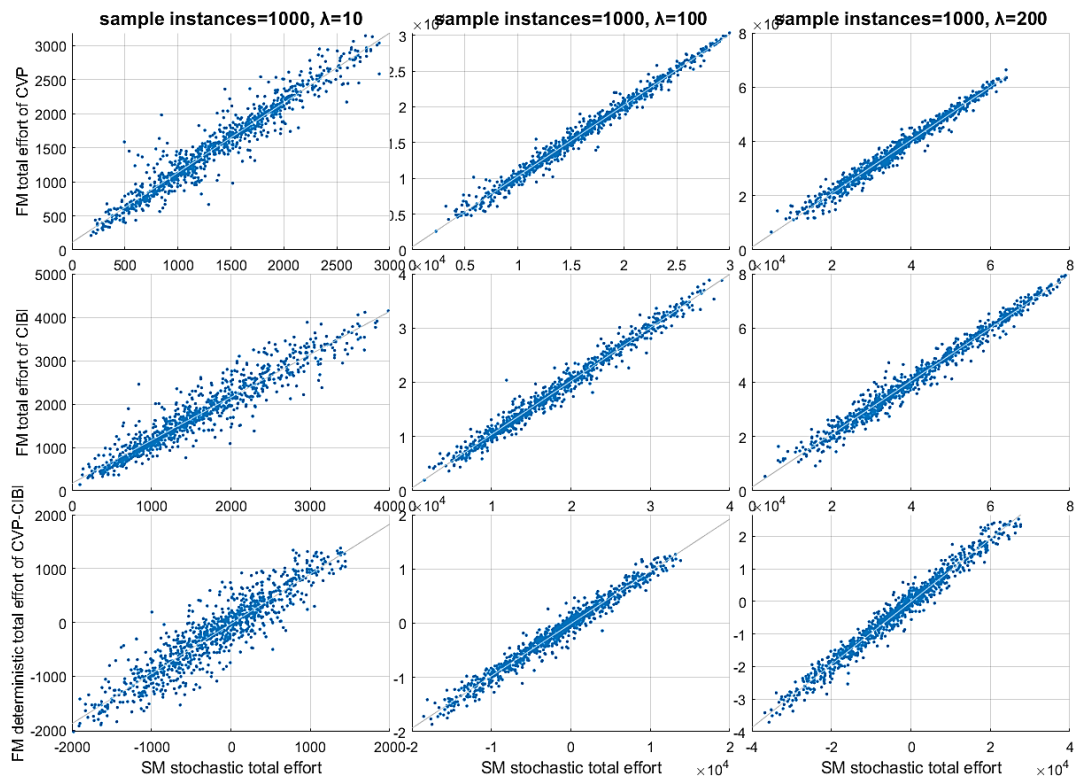


Figure 6-32: Scatter diagrams of external correlation among formal and simulated values of total effort (CVP, CIBI, CVP-CIBI), concerning all (1000) sample instances, of the 2<sup>nd</sup> simulation state in Table 6-4.

As indicated by the diagrams in Figure 6-32 and Figure 6-33, the long-term coefficient of correlation is almost perfect (approaching to 1.0), while the critical error rate fades to

0.6% in a long-term perspective. In addition, according to T-test results, there is high accuracy in terms of absolute effort assessments concerning their difference of CVP minus CIBI.

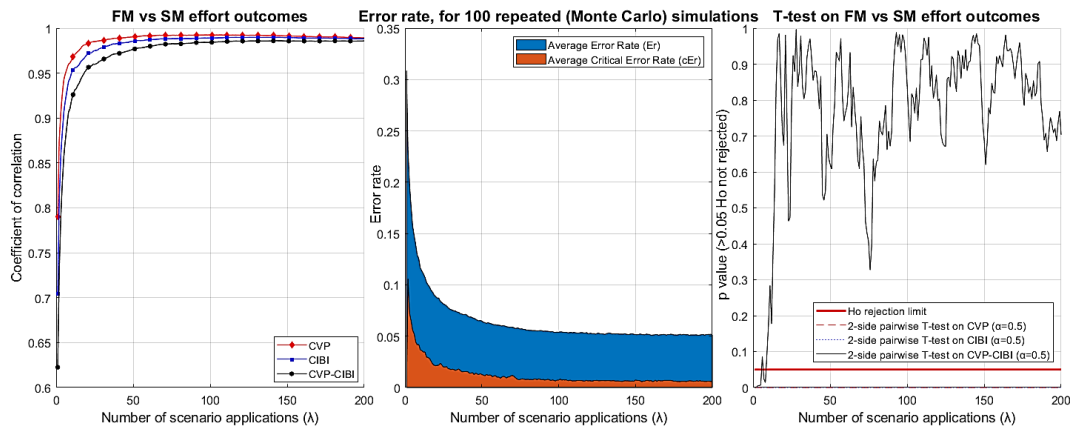


Figure 6-33: Overall control diagrams (left):coefficient of correlation, (right):hypothesis T-test, (mid):average error rate & critical error rate, concerning 1000 sample instances, of the 2<sup>nd</sup> simulation state in Table 6-4.

3. In this state, the alternative maintenance scenarios such as modifications (including debugging) and deletions are engaged. These results provide sufficient evidence that the explicit analysis of expansion scenarios, followed by formal models, has a dominant impact in maintainability assessment, as suggested by the modeling method in chapters 3, 4, and SMC metric. Thus, the expansion analysis of the engaged design patterns covers all the essence of the actual maintenance process.

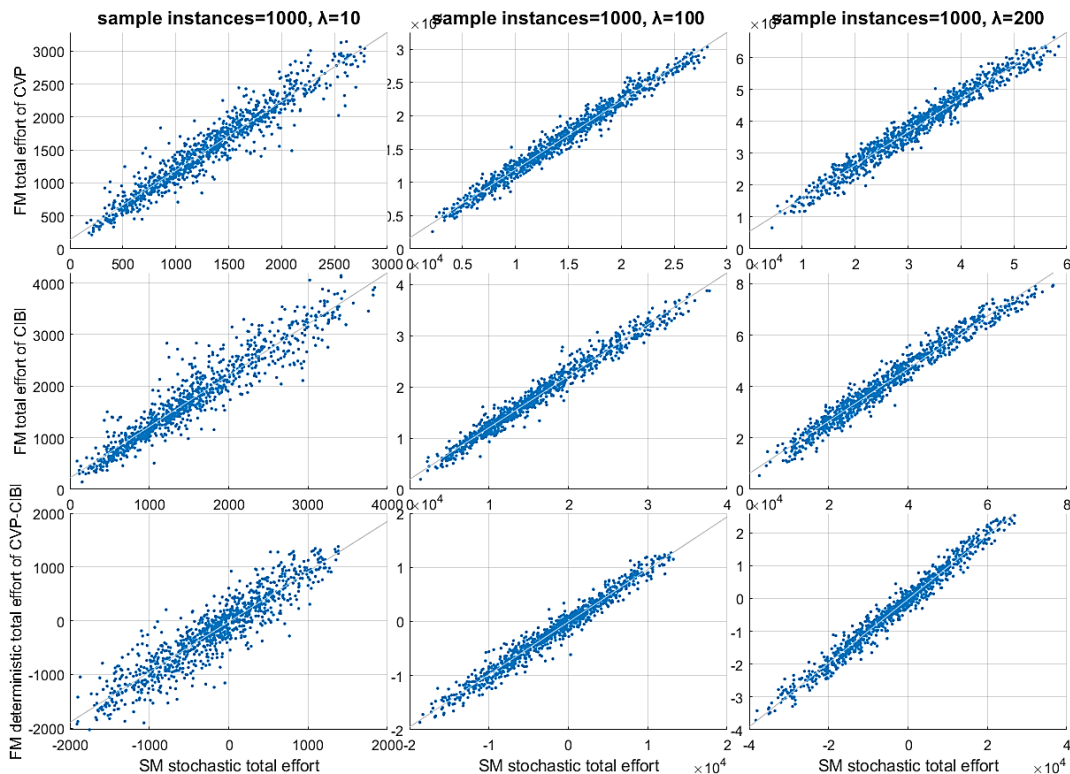


Figure 6-34: Scatter diagrams of external correlation among formal and simulated values of total effort (CVP, CIBI, CVP-CIBI), concerning all (1000) sample instances, of the 3<sup>rd</sup> simulation state in Table 6-4.

As indicated by the diagrams in Figure 6-34 and Figure 6-35, the long-term coefficient of correlation is almost perfect (approaching to 1.0), while the critical error rate fades to 0.8% in a long-term perspective. In addition, according to T-test results, there is high accuracy in terms of absolute effort assessments concerning their difference of CVP minus CIBI.

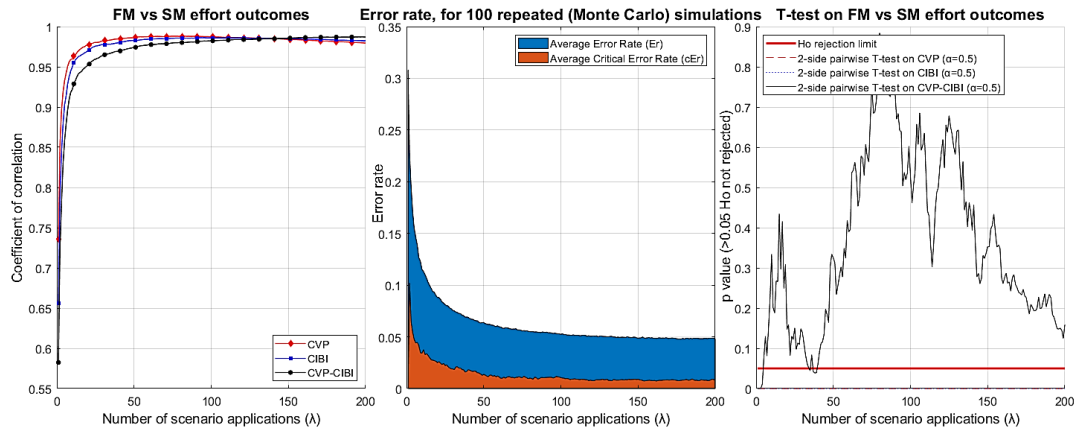


Figure 6-35: Overall control diagrams (left):coefficient of correlation, (right):hypothesis T-test, (mid):average error rate & critical error rate, concerning 1000 sample instances, of the 3<sup>rd</sup> simulation state in Table 6-4.

4. In this state, the scenario’s actual size factor is engaged. These results provide sufficient evidence that effort/size assessments in terms of number of interventions are reliable measurement (proxy) units for comparison purposes, as assumed by the used SMC metric in Table 6-2. Evidence showed that in a mid-to-long-term perspective, the scenarios’ actual size is statistically neutral. Hence, the SMC metric (introduced in chapter 3) is a reliable comparison measure for mid-to-long term size/effort predictions.



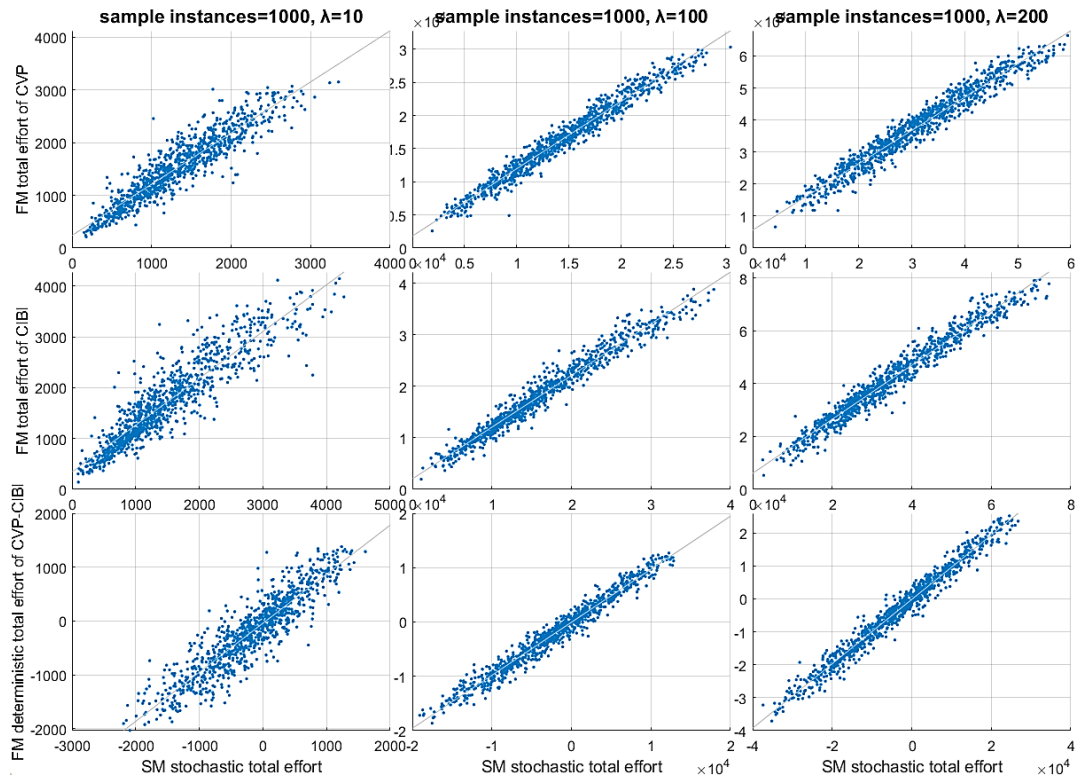


Figure 6-36: Scatter diagrams of external correlation among formal and simulated values of total effort (CVP, CIBI, CVP-CIBI), concerning all (1000) sample instances, of the 4<sup>th</sup> simulation state in Table 6-4.

As indicated by the diagrams in Figure 6-36 and Figure 6-37, the long-term coefficient of correlation is almost perfect (approaching to 1.0), while the critical error rate fades to 0.9% in a long-term perspective. In addition, according to T-rest results, there is high accuracy in terms of absolute effort assessments concerning their difference of CVP minus CIBI.

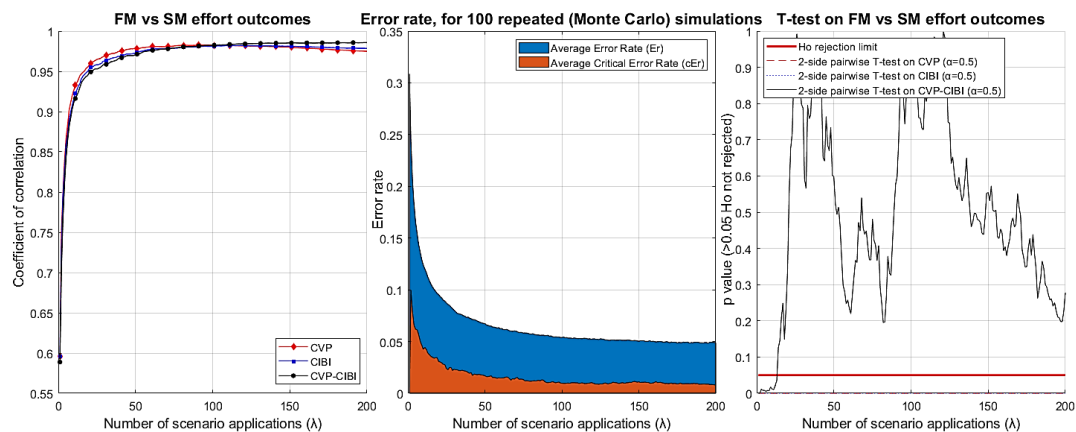


Figure 6-37: Overall control diagrams (left):coefficient of correlation, (right):hypothesis T-test, (mid):average error rate & critical error rate, concerning 1000 sample instances, of the 4<sup>th</sup> simulation state in Table 6-4.

5. In this state, the aging and learning rate factors are simultaneously engaged. The developers’ experience level retains a constant value ( $e_{xp}$ ), initially selected from a fully random or horizontal distribution during sample selection. Under these settings, the formal model loses its correlation power, also failing to provide precise estimations in terms

of absolute values on the effort difference (CVP-CIBI) as indicated in Figure 6-39. On the other hand, its selection reliability, expressed by a low average  $E_r$  (5.2%), remains high. Even if the individual effort assessments of CVP and CIBI demonstrate a low coefficient of correlation (less than 0.5), their difference (CVP-CIBI) demonstrates a relatively high coefficient of correlation (near to 0.8) in a long-term perspective, as also indicated in Figure 6-38. Logically, this contradictory outcome is explained by the fact that developers with same experience work on both design alternatives under comparison, counterbalancing the side effect of constant experience level. However, this is not a usual case since companies and developers have the trend to improve their experience level during the maintenance process. In addition, many developers with different experience level may work during the system’s maintenance. Thus, different frequency distributions of developers’ experience level are explored in the following simulation state.

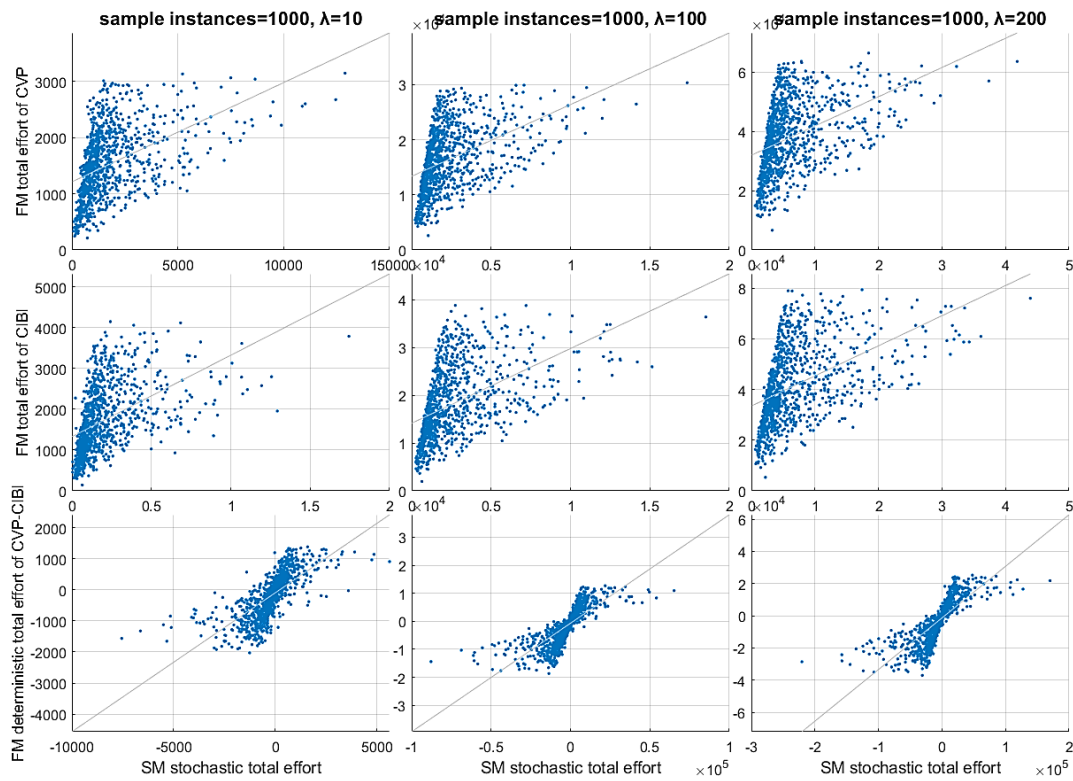


Figure 6-38: Scatter diagrams of external correlation among formal and simulated values of total effort (CVP, CIBI, CVP-CIBI), concerning all (1000) sample instances, of the 5<sup>th</sup> simulation state in Table 6-4.

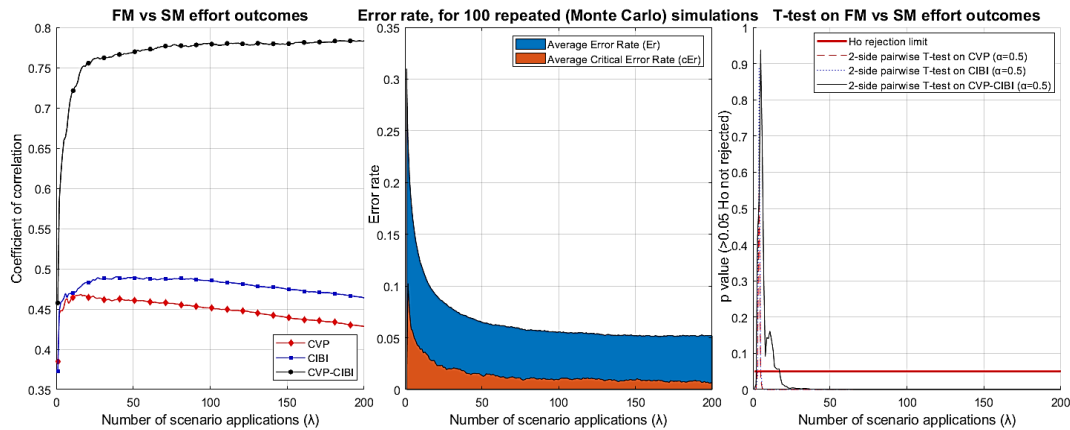


Figure 6-39: Overall control diagrams (left):coefficient of correlation, (right):hypothesis T-test, (mid):average error rate & critical error rate, concerning 1000 sample instances, of the 5<sup>th</sup> simulation state in Table 6-4.

6. In this state, developers’ experience level is defined based on random values of a left-skewed normal distribution, which is probably the most realistic assumption, as supported by empirical evidence in (Woolf, 2016). The results provide sufficient evidence that factors representing code aging issues and developers’ experience, skills, learning rate, or other factors that may influence their productivity are negligible since they are common (for both design alternatives under comparison) and therefore neutral in a mid-to-long-term perspective, as assumed by the evaluated modeling method in chapters 3 and 4.

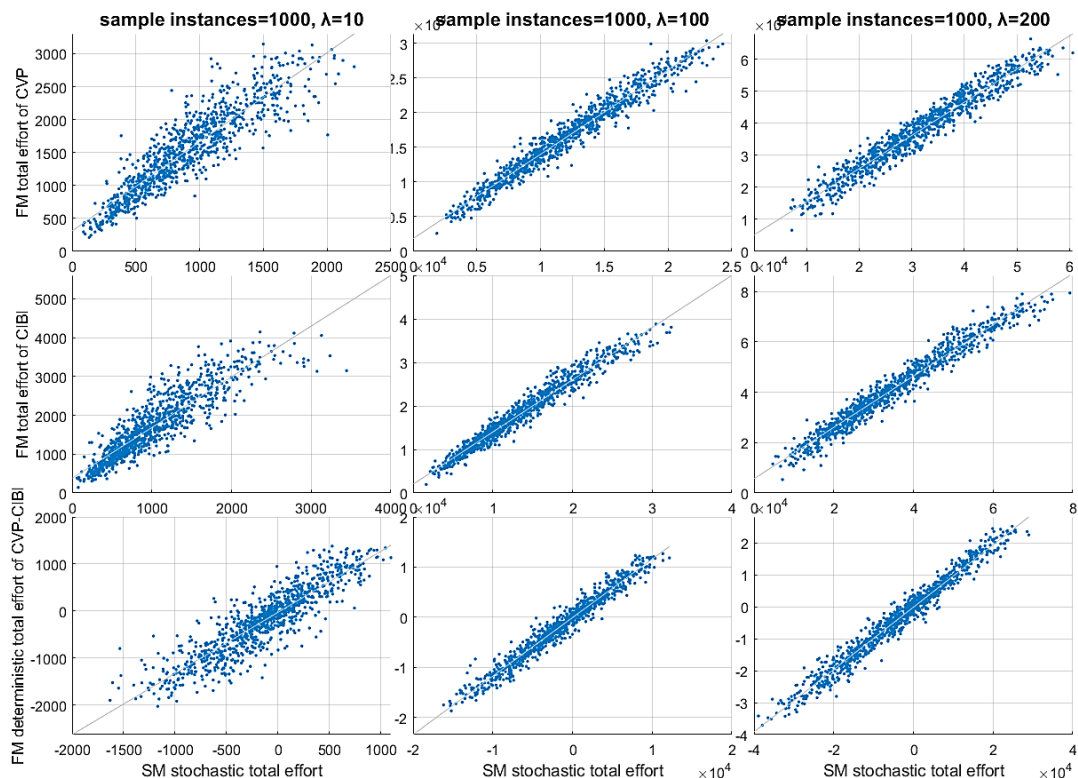


Figure 6-40: Scatter diagrams of external correlation among formal and simulated values of total effort (CVP, CIBI, CVP-CIBI), concerning all (1000) sample instances, of the 6<sup>th</sup> simulation state in Table 6-4.

As indicated by the diagrams in Figure 6-40 and Figure 6-41, the long-term coefficient of correlation is almost perfect (approaching to 1.0), while the critical error rate fades to

0.8% in a long-term perspective. In addition, according to T-rest results, there is low accuracy in terms of absolute effort assessments concerning their difference of CVP minus CIBI. Only in a long-term ( $\lambda > 170$ ) perspective their difference seems to be accurate in terms of absolute effort assessments.

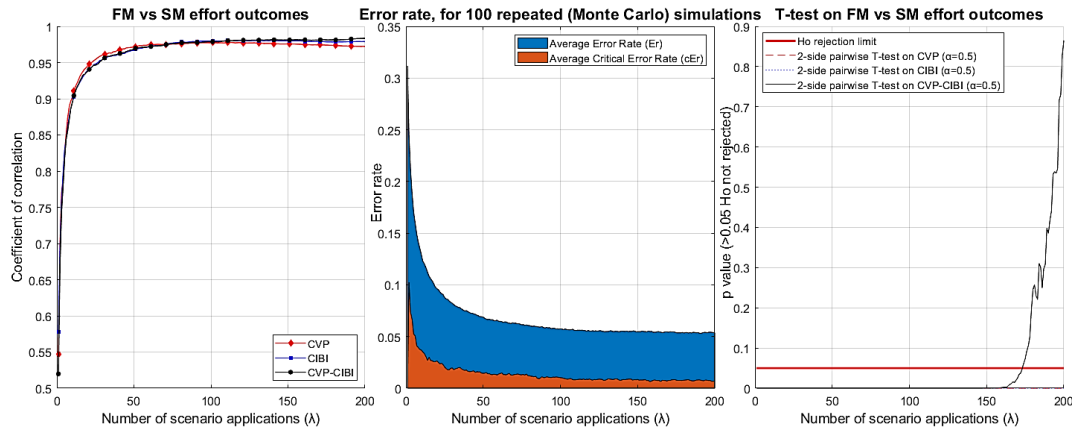


Figure 6-41: Overall control diagrams (left):coefficient of correlation, (right):hypothesis T-test, (mid):average error rate & critical error rate, concerning 1000 sample instances, of the 6<sup>th</sup> simulation state in Table 6-4.

7. In this final and fully stochastic state, the uncertainty factor effecting scenarios' probabilities during simulation is increased. The value of  $f_{BM}$  factor is set to 1.0 implying a high uncertainty level. Thus, the overall uncertainty factor  $f_{BM} \times u_f$  ranges in the  $\pm 3\sigma$  interval of (-0.3, ..., 0.3). The latest results in Figure 6-28 and Figure 6-29 showed that the long-term coefficient of correlation is almost perfect (approaching to 1.0), while the critical error rate fades to 2.0% in a long-term perspective. In addition, according to T-rest results, there is a medium long-term ( $\lambda > 170$ ) accuracy in terms of absolute effort assessments concerning their difference of CVP minus CIBI. These results provide sufficient evidence that the repeated scenario sequences based on constant probabilities, assumed by formal models, approximate the high uncertainty level imposed by shifting scenarios' probabilities during the actual maintenance process. Moreover, the results provide sufficient evidence that the fully stochastic behavior delivered by the simulation model can be adequately expressed or approximated by the limited set of variables of the deterministic formal models introduced in in chapters 3 and 4.

In addition, the formal models' prediction ability, in terms of absolute effort values, is confirmed only for the difference (CVP-CIBI) while they falling to provide similar effort estimations for each individual design alternative as indicated by T-test results in Figure 6-29. Simulation model decreases system's expansion end required effort due to the engagement of alternate scenarios, as indicated by the application example in Figure 6-6.

Even if the states 1 to 5 reflect lumped simulation models of low consistency compared to actual (real-world) maintenance process, they provide insightful evidence and inferences regarding several aspects of the modeling theory and formal models under validation. The most important intermediate inferences are presented below:

- the implied continuous integration, used by the formal models, converges to the discrete calculations of the actual maintenance process as performed by the simulation model.
- the repeated scenario sequences based on constant probabilities, assumed by formal models, approximate the medium uncertainty level imposed by shifting scenarios' probabilities during the actual maintenance process.

- the explicit analysis of expansion scenarios, followed by formal models, has a dominant impact in maintainability assessment, as suggested by the modeling method in chapters 3 and 4, the derived formal models, and SMC metric. Thus, the expansion analysis of the engaged design patterns covers all the essence of the actual maintenance process among design alternatives mainly for comparison purposes.
- effort/size assessments in terms of number of (classes and method) interventions are reliable measurement (proxy) units for comparison purposes in a mid-to-long term perspective, as assumed by the used SMC metric in Table 6-2.
- factors representing code aging issues and developers' experience, skills, learning rate, or other factors that may influence their productivity are negligible since they are common (for all design alternatives under comparison) and therefore neutral concerning the decision-making process in a mid-to-long-term perspective, as assumed by the evaluated modeling method in chapters 3 and 4.
- the repeated scenario sequences based on constant probabilities, assumed by formal models, approximate the high uncertainty level imposed by shifting scenarios' probabilities during the actual maintenance process.

Finally, it is important, that even under slightly different assumptions such as frequency distributions (including fat-tails variations of limited exposure to tail-risk), initial values, and intervals regarding the stochastic variables of the simulation model in Table 6-3, the results (not included in this thesis) of the performed sensitivity analysis revealed that the evaluated formal models exhibit a sustainable decision-making performance.

### 6.5.3 Pattern Exploration of Decision Errors

In this subsection, a further analysis regarding the pattern of critical error occurrences is presented. Different design implementations that require different amounts of effort during maintenance can be seen by managers as alternative investment options. Under this perspective, managers want to reduce the required maintenance effort/size, retaining a low risk of possible wrong selection. Prediction models, like the derived formal models, involve various types of events or classes of resembling maintenance scenarios based on assessments of their probabilities. As a result, probabilistic models could have some accuracy issues for some marginal cases, introducing an error rate of wrong selections. Through further analysis of individual errors, critical errors with high negative impact in terms of wasted effort, and high severity degree in terms of high probability to occur are spotted. A more in-depth analysis of critical errors can reveal a classification pattern of those marginal cases, for which the prediction model is likely to fall into a critical error. Thus, the error rate is another sophisticated measure of the model's reliability degree, focusing on decision-risk taken by the software designers and managers.

#### 6.5.3.1 Error Rate Assessment

More specifically, the error rate ( $Er$ ) is computed through repeated simulations for different  $\lambda$  values and 100 repetitions for each  $\lambda$  value and each sample instance, as illustrated in Figure 6-42 (left side). Repeated executions on the parameters of the same sample instance are suggested in the context of several studies about testing randomized software for simulation purposes (Jabangwe et al., 2015). Next, the intermediate results are compared to formal model's (deterministic) results, while occurring errors are counted. The error rate ( $Er$ ) of all the repeated simulations corresponds to the error rate ( $Er$ ) of each specific sample instance. Furthermore, error measurements of all sample instances are averaged in average error rates (avg  $Er$ ) per  $\lambda$  value, as analyzed in Figure 6-42. Thus, the average error rates (avg  $Er$ ) correspond to the entire design space (of all random sample

instances) of the general problem under study. Further analysis of error rate distribution provides evidence about the actual percentage occurrences of critical errors.

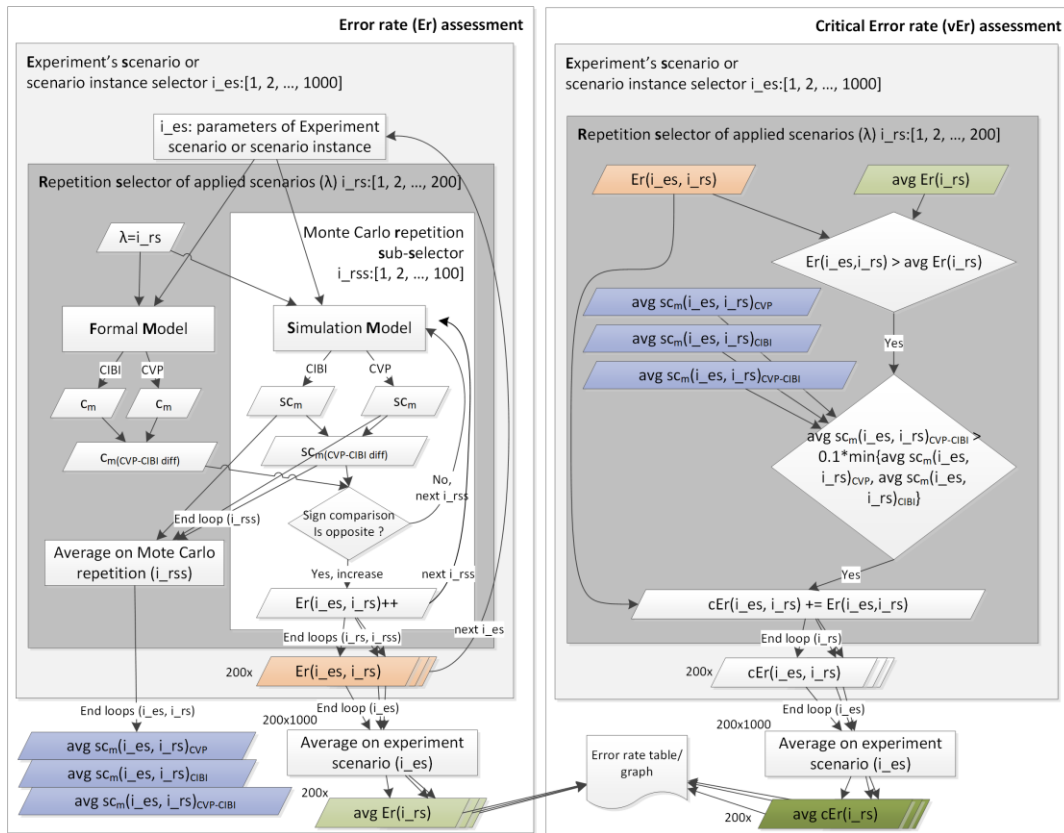


Figure 6-42: Experimental error rate assessment.

The analysis of the results provides average error rates (avg Er) for different  $\lambda$  values, as presented in Figure 6-29 (middle), referred to the 7<sup>th</sup> simulation state in Table 6-4. At first glance, the average error rate (avg Er) is decreased as the number of scenario applications ( $\lambda$ ) is increased. For  $\lambda > 50$ , the average Er is less than 9.0%, converging to 8.1% for  $\lambda$  near to 200 scenario applications. The frequency distribution of the error rates (Er) of each sample instance for  $\lambda$  near to 200, is presented inside Figure 6-43. Almost 68% of the sample instances have an Er between 0% and 3%. The rest 32% of the sample instances have an Er more than 3%, and only 23% of the sample instances have an Er more than the average  $Er(\lambda) = Er(200) = 8.1\%$ . Thus, there are sufficient evidence that the proposed formal models deliver accurate results with a (long-term) average error rate near to 8.1%, even under the 7<sup>th</sup> simulation state of high uncertainty in Table 6-4. However, it is important the critical errors in terms of wasted effort be further investigated.

### 6.5.3.2 Critical Error Rate Assessment

A critical error arises when there is a significant amount of wasted effort and a high probability to occur. Thus, high wasted effort and high probability indicate the significant impact of a critical error and the risk taken during the decision-making process. More specifically, the critical error rate (cEr) is computed through the conditional comparison of intermediate results from error rate assessment in previous step, for each  $\lambda$  value and each sample instance, as illustrated in Figure 6-42 (right side). Thus, error rate assessment must precede critical error rate assessment since the results of the first step are prerequisites in the second step.

The analysis of the results provides average critical error rates (avg cEr) for different  $\lambda$  values, as also presented in Figure 6-29 (middle), referred to the 7<sup>th</sup> simulation state in Table 6-4. The average critical error rate (avg cEr) is also decreased as the number of scenario applications ( $\lambda$ ) is increased. For  $\lambda > 50$ , the average critical cEr is less than 2.5%, converging to 2.0% for  $\lambda$  near to 200 scenario applications. Thus, there are sufficient evidence that the proposed formal models deliver accurate results with a (long-term) average critical error rate near to 2.0%, even under the 7<sup>th</sup> simulation state of high uncertainty in Table 6-4.

### 6.5.3.3 Pattern of Sample Instances Prone to Critical Errors

The entire design space of the general problem of CVP vs. CIBI (recursive hierarchies of part-whole representations) is statistically represented by the sample of 1000 random instances (defined in subsection 6.4.4) which are graphically presented in the two-dimensional scatter diagram in Figure 6-43. Nevertheless, this scatter diagram represents a seventh-dimensional data space:

- *Data dimension 1 and 2:* Axis x represents the auxiliary factor  $\mu=M/N$  which is the rate of number of initial operations (M) and number of initial elements (N).
- *Data dimension 3 and 4:* Axis y represents the probability factor ( $p_{nE}=1-p_{nP}$ ) for a new element which also indirectly expresses the probability factor ( $p_{nP}$ ) for a new operation.
- *Data dimension 5:* The mark's size of each sample instance reflects the magnitude of its error rate (Er) or else the probability or likelihood to occur an incorrect decision. The larger the mark's size of the sample instance, the higher the probability the formal model's predictions to lead to a incorrect decision for that sample instance.
- *Data dimension 6:* The mark's color of each sample instance reflects the magnitude of the decision's impact in terms of rate of gained or wasted effort. This magnitude of decision's impact is expressed by the rate of the average (for all repeated simulations) effort difference (CVP-CIBI) to the minimum of the average efforts of CVP and CIBI design alternatives. The mark's color of the sample instance, as arranged in the side color-bar, indicates the severity degree or the (%) average rate of the wasted effort in case of wrong decision for the specific sample instance.
- *Data dimension 7:* Finally, the red bordered marks of each sample instance reflect critical error occurrences. As illustrated in Figure 6-42 (right side), a sample instance is characterized as prone to critical errors if:
  - its error rate is greater than the average error rate of all sample instances ( $Er > \text{avg } Er$ ), reflecting the high probability to occur a wrong decision, AND
  - its average (for all repeated simulations) effort difference (CVP-CIBI) is greater than the 10% of the minimum of the average efforts of CVP and CIBI design alternatives, reflecting the high severity degree or the high rate of wasted effort of possible wrong decisions.

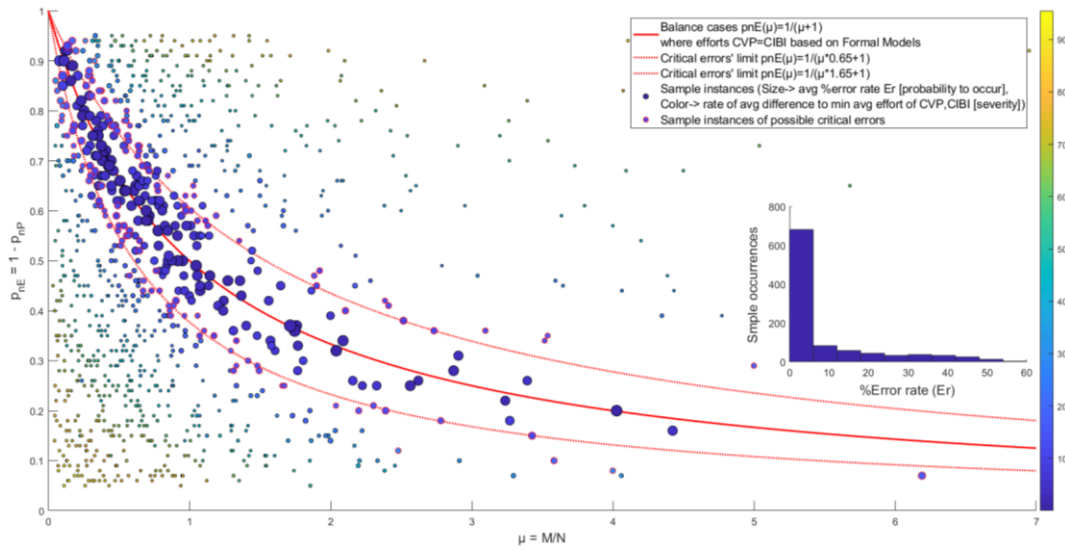


Figure 6-43: Design space representation through 1000 sample instances of CVP vs. CIBI general problem, including long-term error and critical error rate assessment ( $\lambda=200$ , 7<sup>th</sup> simulation state in Table 6-4).

In principle, a decision error occurs when the sign of the simulation model distance (CIBI-CVP difference) is different from this returned by formal models. Thus, setting the difference equation equal to zero,  $p_{nE}=1-p_{nP}$ ,  $\mu = M/N$ , for  $\lambda > 0$ , the equation of balance cases is derived  $p_{nE}(\mu) = 1/(\mu + 1)$ , which indicates the relation of independent variables with equal effort estimations between CVP and CIBI alternatives, as showed in subsections 3.5 and 3.6. Scenario instances that satisfy this equation should be prone to potential errors, as illustrated in Figure 6-43, for  $\lambda = 200$  in the 7<sup>rd</sup> simulation state in Table 6-4. In did, decision errors are more likely to occur (having higher  $Er$  or larger mark's size) for those sample instances closer to the trace of balance cases. Respectively, moving away from the trace of balance cases, decision errors are less likely to occur (having lower  $Er$  or smaller mark's size). However, approaching the trace of balance cases, possible decision errors have lower severity degree or rate of wasted effort (blue colored marks). Respectively, moving away from the trace of balance cases, decision errors are more severe with higher rate of wasted effort (yellow colored mark). This contradiction is confirmed by the sample instance prone to critical errors, indicated by red bordered marks. Referring to Figure 6-43, sample instances prone to critical error occurrences follow two separate traces (limits), which are correlated to the balance equation trace. All the sample instances prone to critical errors (high probability to occur and high severity degree) are aligned up on those traces. Moving away from these traces, error occurrences are either almost impossible to arise (away from the trace of balance cases) or have negligible severity degree or possible rate of wasted effort (towards the trace of balance cases). Finally, all the sample instances below the trace of balance cases are suited for CVP design alternative, while all the sample instances above the trace of balance cases are suited for CIBI design alternative.

Evidence shows that by setting distance equations equal to zero, balance equations can be derived which reveal the pattern or the traces of non-critical and critical error occurrences. Furthermore, by placing critical design attributes or factors on diagrams such as in Figure 6-43, software engineers can develop a “feel” about the design spectrum of possible instances of a significant and general design problem.

#### 6.5.4 Uncertainty Considerations

As indicated in Figure 6-20, the intermediate effort outcomes per scenario application resemble to stationary time series with constant mean and finite variation. The irregular



fluctuations of these series correspond to white noise which is the result of the uncorrelated random variables engaged by the simulation model. Such time series are not deterministic, and it is difficult to forecast with certainty what will occur in the future. Time series analysis (e.g., through ARMA models (G. Antoniol et al., 2001; Shariat Yazdi et al., 2016)) attempts to understand the nature of time series and it is often useful for future forecasting. These models simply predict the statistical properties (i.e., mean, variance, auto correlation) of the time series, assuming that they will be the same in the future as they have been in the past. However, this type of analysis completely ignores the nature of the phenomenon under study. In contrast, the introduced simulation model generates time series of effort outcomes the values and the variability of which are determined by the model parameters, the design attributes, specific events, and a standard measurement approach, while several high uncertainty factors are incorporated. Thus, since the simulation model returns future effort values per applied scenario in the form of time series, the variability of these series would be more informative and probably more realistic about the nature of the phenomenon under study or else about the software evolution during maintenance process. Besides that, these time series have been calibrated regarding their variability based on frequency distributions of real-world observations from time series analysis, as discussed in subsection 6.4.9.

Given that the simulation model provides representative effort assessments in respect to the design structure of each specific problem, it can be assumed that the average value of the effort assessments of several repeated (Monte Carlo) simulations for a specific problem’s instance would approximate the most probable or else the actual or realized maintenance effort. This exactly implied by the indicative frequency distributions of the effort outcomes for 100 repeated simulations in Figure 6-18. Any deviations from that mean value express the uncertainty or the possibility of an incorrect prediction and consequent decision during the design phase. By analyzing these frequency distributions for each scenario application ( $\lambda$ ), the progressive uncertainty is depicted in Figure 6-44. Concerning the difference of the effort outcomes (CVP-CIBI), their outlier values in the distributions explain the error rate of the decision-making. If, for example, the initial effort predictions during the design stage are near to the mean value, a possible and not anticipated realization of the maintenance process can be end up to one of the outliers. In the specific example in Figure 6-44, most of these outliers have values with reverse sign, implying that the initial effort predictions and the decision made was incorrect.

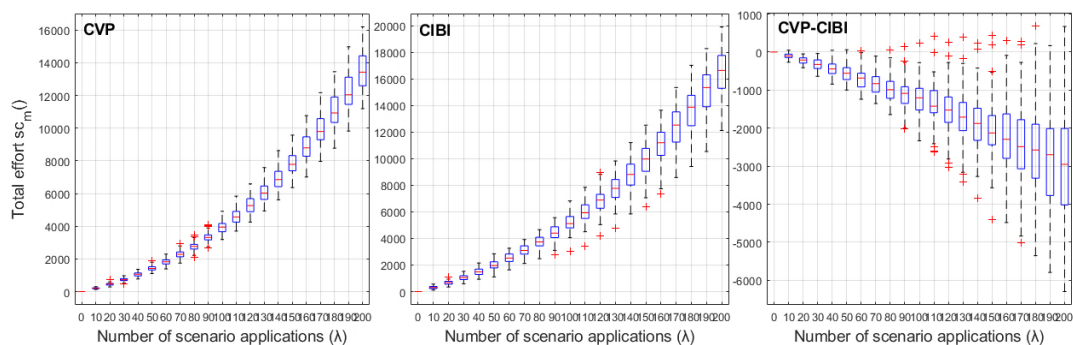


Figure 6-44: Frequency distributions (Box Plots) of Simulation Model’s outcomes (CVP, CIBI, CVP-CIBI) for 100 repeated simulations in a single object-subject instance ( $N:40, M:10, p_{ne}:0.5, e_{xp}:1.0$ ), of the 7<sup>th</sup> fully stochastic simulation state in Table 6-4.

However, even if the level of uncertainty in Figure 6-44 seems to increase for longer maintenance period, occasionally, the uncertainty level in software life cycle is expressed differently. More specifically, it is expressed by the ratio of mean plus deviation to the mean value or else by the ratio of a possible prediction to realized value. This is the interpretation

of the uncertainty levels as expressed by the famous cone of uncertainty in (B. Boehm, 2008; B. W. Boehm, 1984). The earlier a decision is made the greater the uncertainty of this prediction to deviate from the realized cost or size as represented by the unit value of the horizontal axis in Figure 6-45.

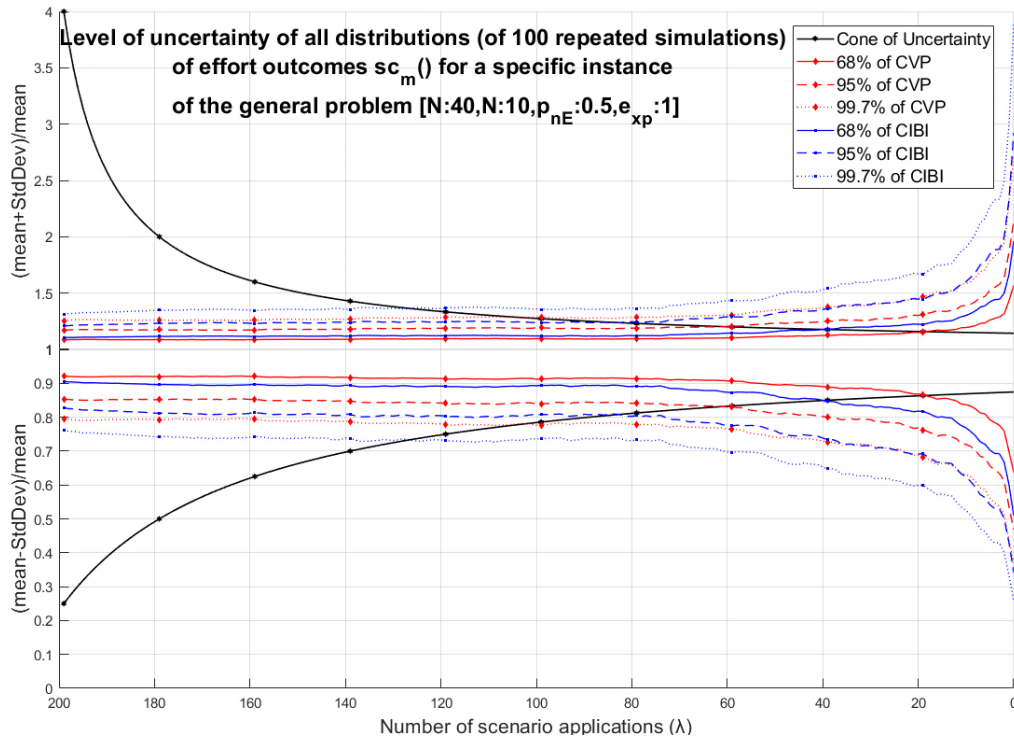


Figure 6-45: Levels of uncertainty distributions of Simulation Model’s outcomes (CVP, CIBI) for 100 repeated simulations in a single object-subject instance (N:40, M:10,  $p_{ne}$ :0.5,  $e_{xp}$ :1.0), of the 7<sup>th</sup> fully stochastic simulation state in Table 6-4.

The ratio of mean plus deviation to the mean value for each effort distribution (CVP and CIBI) in Figure 6-44 is combined with the cone of uncertainty in Figure 6-45. The order of scenarios’ application has been reversed since the earlier the design decision is made, the greater the number of future scenario applications ( $\lambda$ ). Surprisingly, under this representation, the earlier (higher  $\lambda$  value) the design decision is made, the narrower the level of uncertainty, practically reversing the cone of uncertainty. This means that the effort predictions and consequent decisions based on the simulated outcomes are more reliable for longer forecasts. Thus, the longer the maintenance period of software the higher the probability of the realized cost (effort, size) to be near the predicted (mean) value returned by the simulation model. Since the simulation model successfully validated the reliability of the formal modes, the evaluated formal modes and relevant modeling theory in chapters 3 and 4 also exhibit limited uncertainty levels. It is important that reassessment attempts by reapplying the formal models in a later stage providing it with updated information, as suggested in (Aroonvatanaporn, Sinthop, & Boehm, 2010; Eveleens & Verhoef, 2009), is almost meaningless. Even the realization that the initial design selection was incorrect, the adjustment of the existing code to a different design alternative would require the redesign of code structure which is an extremely complicated and effort consuming process as indicated in chapter 3. This is an argument that explains how critical the early design decisions are since after a specific design alternative has been selected, all the following interventions during maintenance should be conformed with the initially selected design structure.

The reason of the limited uncertainty levels in Figure 6-45 is the fact that the formal models are sensitive and well fitted in the design characteristic of each specific instance of

a general problem. Since the maintenance process is dictated by the structural behavior of the engaged design patterns, any random variation or white noise caused by unpredictable influences would have stationary statistical characteristics, thus minimizing their overall long-term effect. Notice that all the discussion is about ratios (or analogies) of variations against means, and not about absolute values which have not been strictly validated against real-world observations.

### 6.5.5 Statistical Evaluation per Sample Instance

In this subsection, a statistical evaluation of an indicative instance of the CVP vs CIBI general design problem is presented. The analyzed instance is referred to the practical motivation example of GUI implementations in Table 1-1 with the following parameters:  $N=15$ ,  $M=14$ ,  $p_{nE}=0.70$ , and  $p_{nP}=0.30$ . This implementation corresponds to a marginal case since the difference of effort assessments among design alternatives is relatively small. The evaluation is based on the results of the conducted simulations as graphically represented in Figure 6-46. Furthermore, statistical evaluations for several indicative sample instances of the CVP vs CIBI general design problem are provided in Appendix D.

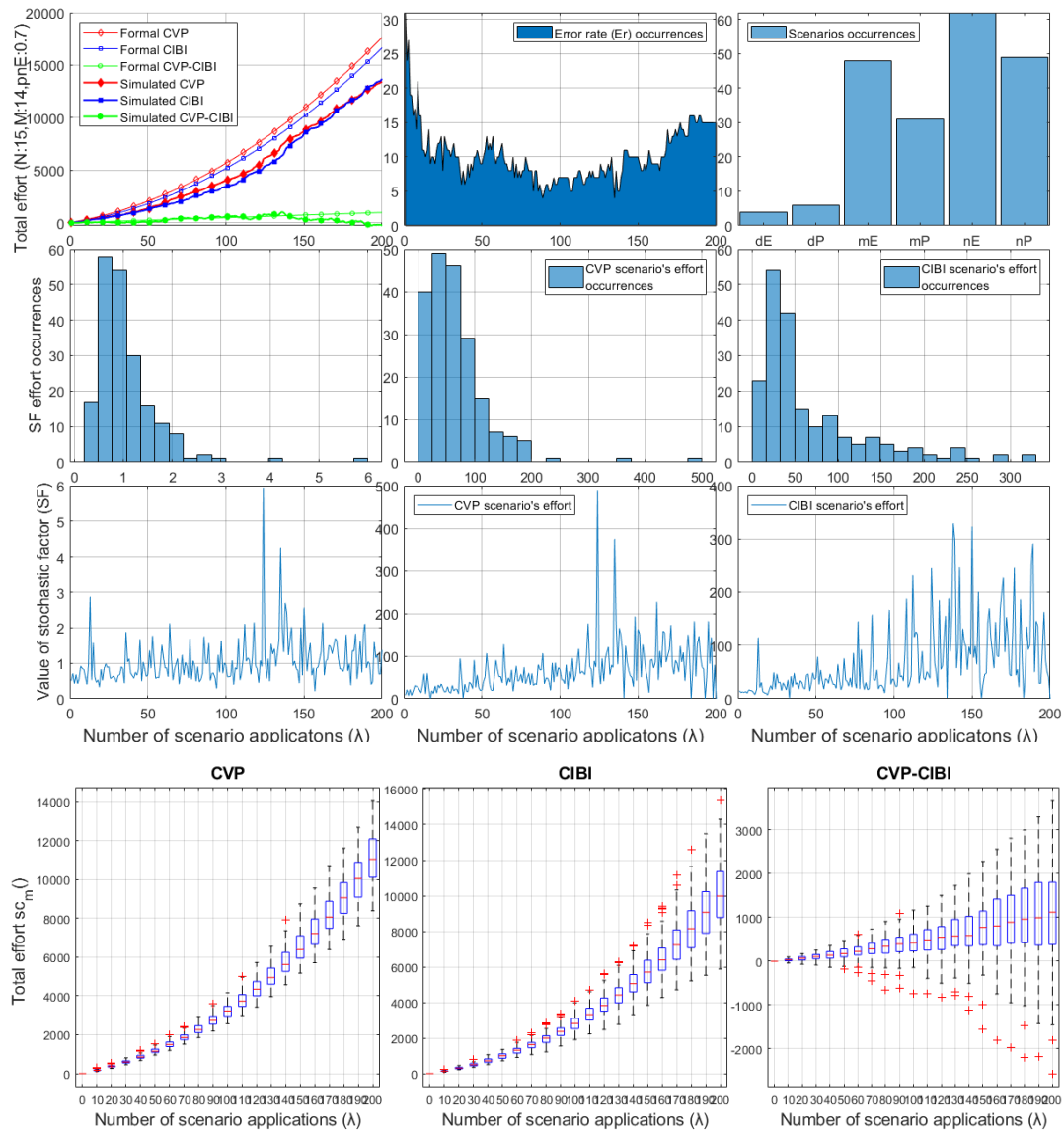


Figure 6-46: Statistical evaluation of GUI implementation based on simulation outcomes

Referring to Figure 6-46, several subgraphs are presented to provide an in deep understanding of conducted simulations concerning the GUI implementation (problem's instance). The simulated effort assessments are referred to 7<sup>th</sup> fully stochastic simulation state in Table 6-4. More specifically, a detailed analysis per subgraph follows:

*The upper left graph:* provides the deterministic (total) effort assessments returned by the formal models for CVP and CIBI design alternatives as well as for their difference (CVP-CIBI). In addition, it provides the stochastic (total) effort assessments returned by the simulation model for a single indicative simulation. Formal model's results indicate that CIBI design alternative is preferable since requires the lesser maintenance effort. However, it is obvious that the difference of effort assessments is relatively small, thus the specific GUI implementation (problem's instance) is probably a marginal case where the decision outcome provided by formal models may be subject to significant risk. Thus, due to the stochastic nature of the simulation model, the simulated effort assessments are subject to several stochastic factors. In this marginal case, the simulated effort assessments per design alternative are interchanged during maintenance, implying that the decision outcome is not straightforward as in the case of formal model's (deterministic) outcomes.

*The upper middle graph:* provides a clear assessment of the decision risk taken through the error rate (Er). This assessment is the result of the comparison between (deterministic) formal models' outcome and several repeated (Monte Carlo) simulations for the same problem's instance. The graph represents the evolution of the error rate (Er) during the maintenance process as expressed by the number of applied scenarios ( $\lambda$ ). Usually, the error rate constantly decreases during simulation while for the first applied scenarios ( $\lambda < 50$ ) its decrement is quite sharply. However, in this marginal case, the result shows that in a mid-term perspective ( $\lambda \approx 100$ ), the design decision supported by formal modes is subject to a reasonable risk near 5%, while in a long-term perspective is subject to a significant risk near to 15%. Because GUI implementation is a marginal case, the results are quite interesting as the error rate initially decreases and later increases. The initial (expected) decrement rate is justified by the statistical convergence of the engaged stochastic factors around their mean values, thus as new scenarios are applied the overall effect of the stochastic factors tend to become neutral. This reduces possible sharp fluctuations and the likelihood of unexpected interchanges among design alternatives and, thus decreasing decision-risk. However, the (unexpected) increment of error rate is justified by the increased severity (impact) of future interventions that cause sharp fluctuations that counterbalance and eventually overcome the statistical convergence. From a different perspective, as the maintenance process evolves and the impact of interventions increases, instant and sharp fluctuations increase the likelihood of unexpected interchanges among design alternatives and thus to higher decision-risk. Again, it is important that this peculiar behavior is due to the marginal nature of the GUI implementation or the limited difference of effort assessments among design alternatives. The interpretation of this behavior is insightful about the ability of the simulation model to imitate the actual software evolution during maintenance as well as its sensitivity to adapt its behavior to specific instances of the design problem under study.

*The upper right graph:* provides the number of occurrences per major scenario type that have been randomly generated (base on their probabilities) and applied during a single indicative simulation. dE and dP correspond to the maintenance activities for deleting an element and operation respectively. mE and mP correspond to the maintenance activities for modifying an element and operation respectively. nE and nP correspond to the maintenance activities for adding an element and operation respectively. The result shows that modification and addition of elements have the greater portion since under GUI implementation the probability of adding new elements is higher ( $p_{nE} = 0.70$ ) as assessed by the system's specifications in Table 1-1.

*The second left graph:* provides the frequency distribution of the values of the overall stochastic factor of the simulation model per applied scenario for a single indicative simulation. The values of this stochastic factor encompass all the variability introduced by several random and stochastic factors such as method's actual size, aging issues, and developers experience.

*The second middle graph:* provides the frequency distribution of the intermediate (per applied scenario) CVP effort assessments of the simulation model for a single indicative simulation. The shape of this distribution is right skewed as a result of the intensive calibration efforts to imitate the statistical characteristics of real-world (effort-based) assessments from the field of time series analysis as analyzed in subsection 6.4.9.

*The second right graph:* provides the frequency distribution of the intermediate (per applied scenario) CIBI effort assessments of the simulation model for a single indicative simulation. Again, the shape of this distribution is right skewed as a result of the intensive calibration efforts as in the case of CVP effort assessments.

*The third left graph:* provides, in the form of time series, the values of the overall stochastic factor of the simulation model per applied scenario for a single indicative simulation.

*The third middle graph:* provides, in the form of time series, the intermediate (per applied scenario) CVP effort assessments of the simulation model for a single indicative simulation. The observed fluctuations correspond to the typical behavior of random variables representing stochastic processes.

*The third right graph:* provides, in the form of time series, the intermediate (per applied scenario) CIBI effort assessments of the simulation model for a single indicative simulation. Again, the observed fluctuations correspond to the typical behavior of random variables representing stochastic processes.

*The fourth left graph:* provides the frequency distributions (per applied scenario) of the CVP effort assessments of the simulation model for several repeated (Monte Carlo) simulations for the same problem's instance. This type of graph represents the evolution of uncertainty degree during maintenance as discussed in subsection 6.5.4.

*The fourth middle graph:* provides the frequency distributions (per applied scenario) of the values of CIBI effort assessments of the simulation model for several repeated (Monte Carlo) simulations for the same problem's instance.

*The fourth right graph:* provides the frequency distributions (per applied scenario) of the difference of CVP-CIBI effort assessments of the simulation model for several repeated (Monte Carlo) simulations for the same problem's instance. This type of graph represents the evolution of uncertainty degree during maintenance concerning the decision-making process. The result shows that in an early ( $\lambda < 50$ ) and long term ( $\lambda > 150$ ) perspective there several values below zero, thus representing an opposite decision (CVP) against formal models' decision (CIBI). In practice, these (opposite) values confirm the significant error rate ( $E_r$ ) as quantitatively expressed in the upper middle graph. The more the opposite values, the higher the likelihood of the formal models to conclude in an incorrect decision.

Referring to the implementation of GUI, the formal models demonstrate a significant long-term decision-risk (near to 15%) or else a moderate decision-making reliability near to 85%. Thus, there 85% chance for the formal model to conclude to the right design decision for the specific GUI implementation (problem's instance). However, problem's instances with high error rate have small severity, thus the possible gain or loss (in terms of effort) would be limited or even insignificant as discussed in subsection 6.5.3. In general, even when for a specific problem's instance, the error rate is significant, its severity or the possible gain or loss (in terms of effort) would be probably limited. Only a small portion of error occurrences may be critical in terms of severity as expressed by the critical error rate ( $cE_r$ ) in Figure 6-29 (middle).

Referring to the entire sample of instances, the formal models demonstrate variant decision-risk or decision-making reliability. For example, the sample instance N.001 demonstrate a long-term error rate (Er) near to 25%, the sample instances N.011, 012 less than 10%, the sample instances N. 14, 15 less than 5%, and the sample instances N.004, 007, 008, 009, 010 fade to zero (0%) as presented in Appendix D. The average of decision-risk (expressed by the error rate and critical error rate) for the entire sample of problem's instances (representing the problem's design space) is depicted in Figure 6-29 (middle) and summarized in Table 6-7.

Conclusively, the decision-making reliability of the derived formal models variates and depends on the parameters of each specific instance of the general problem under study. Marginal cases with narrow effort differences are the most prone to potential error, however, most of these errors would be of low severity in terms of wasted effort. The overall decision-making reliability of the derived formal models is expressed by the average error rate of a sufficient sample of instances that represents the entire design space of the design problem under study.

### 6.5.6 Summarizing Results and Inferences

In general, under medium to high uncertainty assumptions, the evaluated formal models and modeling method are valid in a mid-to-long-term ( $\lambda > 50$ ) perspective regarding their selection ability, demonstrating a) high correlation coefficient ranging from 0.94 to 0.96, b) decreasing average error rate between 9% and 5.4%, and c) decreasing critical error rate from 2.5% to 0.8%. Furthermore, the formal models are also valid in a long-term ( $\lambda > 150$ ) perspective regarding its prediction ability in absolute effort/size magnitudes, under the assumption that companies and developers gradually improve their experience and skill level. However, evidence that imply prediction ability in terms of absolute values may be of low importance as stressed in subsection 6.6.1. Summarizing, the decisions based on the formal models, in a long-term ( $\lambda \rightarrow 200$ ) perspective, demonstrate a) high coefficient of correlation 0.96, b) low average error rate 8%, and c) low critical error rate 2%.

The evidence suggests that simplified modeling approaches such as the introduced modeling method in chapters 3 and 4 are particularly reliable able to approximate dynamic system's behavior mostly because of mid-to-long-term statistical convergence. Conceptually, the introduced modeling theory can be considered as a reverse analysis or a regression analysis on the underlying activities of actual maintenance process, managing to eliminate transitory and stochastic factors which demonstrate a statistically neutral long-term effect. Thus, the introduced approach in chapters 3 and 4 seems that achieves to eliminate transitory and biased factors to enhance mid-to-long-term decision ability.

## 6.6 Conclusions

### 6.6.1 General Requirements and Limitations

The introduced validation procedure requires a sophisticated simulation model, properly adapted to the characteristics of each specific general problem under validation. Thus, the validation of formal models for different general problems requires the development of an alternate simulation model that should incorporate all the relevant scenario types, design attributes, and problem's parameters. In this case, the coefficient of correlation and the error rate could be different.

Furthermore, the simulated outcomes are focused only on the required effort for maintaining the main code of each design alternative. Any additionally required effort for maintaining linked modules or legacy code would be common for all design alternatives under comparison, and thus neutral concerning the decision making. The main objective of the derived formal models in chapters 3 and 4 and therefore of the proposed simulation approach is to provide proportionally equivalent effort estimations primarily for comparison purposes. Therefore, in principle, absolute values of simulated effort outcomes

do not necessarily reflect the actual or realized effort for the entire system. In addition, as in chapters 3 and 4, absolute maintenance cost assessments in terms of salaries, resources, assets, expenses, etc., are out of the scope of the proposed simulation approach.

The modeling method in chapters 3 and 4 emphasizes in the maintainability perspective of general design problems by deriving and analyzing design alternatives, design attributes, and major maintenance scenarios for each of them. A possible failure to conclude on the complete set of problem's parameters, may negatively affect the reliability of the simulation model which replicates the same structural evolution pattern under the same parameters.

The proposed simulation model has been calibrated regarding its variability based on frequency distributions of real-world observations. Thus, further statistical inferences (i.e., correlation, error rate) about formal modes decision-making reliability (or precision) are well supported. However, due to the inadequate volume of homogeneous data, it lacks a strictly validation of simulated effort predictions against real-world observations. Thus, further statistical inferences (i.e., t-test) about formal modes accuracy are rather weak. Moreover, a finer calibration of the simulation model by matching its variability with frequency distributions from an even larger repository of real-world observations would be strengthening model's ability to imitate actual maintenance process.

Referring to the randomly selected sample, all the selected instances adequately represent the entire design space of the problem or the possible systems' instances or the population of the general problem under study. Thus, the population validity, as a type of external validity, is high enough to reasonably generalize the findings of the experiment from the selected sample of instances to the entire design space of the general design problem under study.

Furthermore, since the conducted simulations and the experimental settings and conditions are adequately controlled by the researcher, it is ensured that there are no extraneous factors that could explain (or affect) the returned effort outcome by the simulation model. Thus, since the experiment and the simulation model have high internal validity, it is confidently concluded that the defined independent variables of the simulation model adequately predict the depended variable of the required maintenance effort. However, a possible external threat to validity, as classified in (T. D. Cook & Campbell, 1979), is the concern regarding the ability of the introduced simulation model to imitate the actual or real-world maintenance process in a more realistic way (e.g., by incorporating different independent variables and stochastic factors), and thus to limit the generalization of the experiment results or to create alternate explanations (D. L. Parnas & Curtis, 2009). This threat encompasses any possible concern about the suitability, limitations, or the proper interpretation of the selected (or possible other) parameters, stochastic factors, assumption, constraints, and requirements towards the objectives of the study.

### 6.6.2 Extensions and Further Research

The introduced approach is a starting point for further research in the domain of software evolution throughout simulation models which could engage other stochastic factors in different frequency distributions under other measurement methods and units, or even for other quality characteristics of the software. These research perspectives could be assisted by using general (e.g., MATLAB®) or targeted (e.g., VENSIM®) purpose simulation languages and tools. For example, both formal models as a continuous models and simulation model as an event-driven model can be implemented and further explored through the VENSIM® (dynamic, stochastic and quantitative) simulation tool, as supported in (Müller & Pfahl, 2008) and presented in subsection 7.4. In addition, machine learning or artificial intelligence techniques may be used to assist simulation model reaching even more realistic results.

At the same time, this approach targets on motivating researchers toward the evaluation of other general and significant problems in software architecture domain for which proper selection among design alternatives could be modeled through the theoretical framework in chapters 3 and 4 and statistically validated by simulated observations. For example, several potential problems under evaluation and validation can emerge by taking under consideration the implication of several other competitive design patterns such as Strategy, Decorator, or Prototype against Abstract Factory, or Mediator against Observer, as introduced in (Gamma et al., 1994) and presented in chapter 5. The simulation of such problems can be assisted by integrating the proposed simulation model to perform jointly as a synthesized simulation or co-simulation, as proposed in (Zeigler et al., 2018). In such cases, existing base models might serve as lumped components of a broader simulation model forming a hierarchical structure. These potentials highlight the possible usability of the proposed theoretical framework in (Karanikolas et al., 2017) and the introduced evaluation method through simulations in a wide spectrum of general and difficult designing problems in the software architecture field.

### 6.6.3 Overall Assessment

Decisions made during design stage heavily affect maintainability of software and related time and effort. The proposed modeling method in chapters 3 and 4 generates probabilistic comparison models that estimate the maintainability degree of design alternatives through effort predictions in a formal and deterministic way. This approach manages to limit the ambiguity imposed by the stochastic nature of the actual maintenance process by relying on a limited set of problem's parameters such as design attributes and probabilities of major maintenance scenarios.

The results of the extensive statistical validation indicate that the evaluated formal models provide reliable estimations of the expected effort, especially for comparison purposes. Thus, decisions concerning design alternatives exhibit very limited selection-risk even under high uncertainty levels regarding the initial estimation of problem's parameters. The reliability of the evaluated probabilistic models increases in a mid-to-long term perspective, and thus, as the maintenance process evolves and decisions' benefits become more significant, the models' decision ability to conclude in the most beneficial design alternative in terms of maintainability is increased. Such parsimonious models eliminate transitory and biased factors to enhance mid-to-long-term decision ability. Methods that yield such reliable, formal, general, and reusable models reduce the long-term uncertainty of design decisions and help developers and engineers to elevate the quality of their decision-making. Thus, early structural analysis of the engaged design patterns can significantly improve effort assessments and comparison among design alternatives, practically reversing the (cone of) uncertainty levels. Even if such early and critical design decisions is not the primary concern in software industry, developers and designers should turn their attention on them since the cumulative benefits (avoided wasted effort) from the repeating use of these formal models significantly overcomes their initial derivation cost.

Finally, the proposed validation approach introduces a new perception about the statistical evaluation of formal comparison models and relevant theories regarding their reliability to support design decisions. It relies on massive and homogeneous validation data, sensitive to several design attributes, generated by widely stochastic simulation models which have been thoroughly calibrated to replicate the underlying activities and variability of actual software evolution during maintenance process. Researchers are encouraged and hopefully inspired to possibly apply the introduced concepts in similar or different context.



## 7 Alternate Use of Formal Comparison Models

### 7.1 Chapter Overview

In this chapter, several alternate and future perspectives of the introduced modeling method, derived formal comparison models, and event-driven simulation models are presented. The purpose of this chapter is to demonstrate and explore further potential applications of the introduced theory and models. In addition, this material tries to highlight possible perspectives for further analysis and interpretation of similar or other design problems in the field of software engineering.

More specifically, the introduced modeling method and derived formal models in chapters 3 and 4 are further analyzed to support decision-making under partial or full uncertainty. Thus, when software designers are unable to forecast the scenarios' probabilities in a precise manner. For that purpose, the derived formal models are further integrated on their probability factors. Partial uncertainty refers to integration on a specific interval of possible scenarios' probabilities, while full uncertainty refers to integration on the entire range of scenarios' probabilities. The technique is demonstrated on the formal models of the general problem of part-whole aggregations.

Furthermore, the horizon analysis technique is analyzed. This technique separates the entire maintenance period to subperiods, where for each subperiod different scenarios' probabilities are applied. Under this perspective, even the code development period can be considered as a separate subperiod or a preliminary maintenance subperiod. In particular, the derived formal models in chapter 4 are repeatedly applied on a specific instance of the general problem for different scenarios' probabilities. The initial values of design attributes for each subperiod are estimated based on the last step of previous subperiod. The technique is demonstrated on the example of Interpreter implementation as an instance of the general problem of part-whole aggregations.

Moreover, alternate computer-aided implementations of the introduced models with the assistance of VENSIM tool are presented. This software tool can simulate physical and other phenomena and systems through the analysis of their key variables and their change rates. In this environment, constant and intermediate variables are combined to compute other variables. A special type of variables, called 'levels', is computed through integration based on the change rates that effecting it. The software runs the simulation and computes its variables by performing integration on a special parameter which generally represents the time dimension of the model. This tool can represent both continuous and event-driven models while provide a variety of capabilities for representing and comparing the results of simulations. More specifically, the introduced discrete models in chapter 3, the continuous formal models in chapter 4, and the event-driven simulation model in chapter 6, concerning the general problem of part-whole aggregations, are implemented in VENSIM tool. The alternate modes are visually represented while the documentation of the code of each model is provided. Indicative comparison results of several basic variables among all models are presented. The tool is demonstrated on an example of GUI implementation as an instance of the general problem of part-whole aggregations.

Finally, a discussion about seeing software design process as investment is provided. This discussion examines the decision-making during design phase and the produced software under the view of financial and accounting analysis.

The context of this chapter is based on the quantitative analysis in chapter 3, the introduced modelling method and derived formal models in chapter 4, and the introduced simulation model in chapter 6. The rest of this chapter is organized as follows. Subsection 7.2 demonstrates the decision-making under partial or full uncertainty. Subsection 7.3 introduces the horizon analysis technique. Subsection 7.4 presents alternate computer-aided implementations of the introduced models. Subsection 7.5 examines the software

design process under the view of investments. Finally, in subsection 7.6 conclusions are presented.

## 7.2 Decisions Under Uncertainty

The introduced modeling method in chapters 3 and 4 is based on estimations of major scenarios' probabilities according to the scope of each specific problem's instance. Referring to CIBI vs CVP general problem,  $p_{nE}$  and  $p_{nP}$  are the probabilities for the scenarios of adding a new composition's element and adding a new type of operation, respectively. The equations of the generated formal models are based on specific values of those probabilities' factors as independent variables. Thus, for each specific instance of the general design problem, software engineers should estimate those scenario probabilities as absolute values. For instance, in the case of the Interpreter specific problem with design attributes  $N=40$  and  $M=10$ , the scenario probabilities are estimated as  $p_{nE}=p_{nP}=0.5$ . Nevertheless, in many cases, it is difficult to obtain such absolute probabilities with satisfactory certainty. In such cases, however, a confidence interval of these probabilities is more likely to be estimated. In this subsection, a technique that allows the use of the derived formal models as they are fed by intervals instead of absolute values of scenario's probabilities is presented.

### 7.2.1 Transforming Formal Models to Support Decision-Making Under Uncertainty

Having extracted formal model equations, the decision-making can be supported based on estimations of intervals instead of a single value for a specific probability factor. This approach could be achieved through further integration of formal model equations on the probability factor of interest. For example, referring to CIBI vs. CVP general problem, the formal model equation of total effort for CVP design combination is presented in equation (7-1) where  $p_{nP}$  factor has been replaced by  $1-p_{nE}$ .

$$c(CVP, N, M, p_{nE}, p_{nP}, \lambda) = \frac{3}{2}\lambda^2 p_{nE} p_{nP} + \lambda p_{nP} N + 2\lambda p_{nE} M + 4\lambda p_{nE} + \lambda p_{nP} = \quad (7-1)$$

$$\frac{3}{2}\lambda^2 p_{nE} (1 - p_{nE}) + \lambda (1 - p_{nE}) N + 2\lambda p_{nE} M + 4\lambda p_{nE} + \lambda (1 - p_{nE})$$

By integrating equation (7-1) on the  $p_{nE}$  factor, the equation (7-2) is derived, which is a general equation. Furthermore, by integrating  $p_{nE}$  factor for a specific interval of min and max values, the cumulative prediction of the required effort for all probabilities in this interval is returned.

$$c(CVP, N, M, [p_{nE(min)}, p_{nE(max)}], [[1 - p_{nE(min)}, 1 - p_{nE(max)}], \lambda) = \quad (7-2)$$

$$\int_{p_{nE(min)}}^{p_{nE(max)}} \frac{3}{2}\lambda^2 p_{nE} (1 - p_{nE}) + \lambda (1 - p_{nE}) N + 2\lambda p_{nE} M + 4\lambda p_{nE} + \lambda (1 - p_{nE}) dp_{nE}$$

A similar equation (7-3) can be derived for CIBI design combination.

$$c(CIBI, N, M, [p_{nE(min)}, p_{nE(max)}], [[1 - p_{nE(min)}, 1 - p_{nE(max)}], \lambda) = \quad (7-3)$$

$$\int_{p_{nE(min)}}^{p_{nE(max)}} \frac{3}{2}\lambda^2 p_{nE} (1 - p_{nE}) + 2\lambda (1 - p_{nE}) N + \lambda p_{nE} M + \lambda p_{nE} + 2\lambda (1 - p_{nE}) dp_{nE}$$

The equations (7-2) and (7-3) are able to support decision making among design alternatives of the CVP vs. CIBI general problem under conditions of partial or even full uncertainty. Thus, when designers are unable to estimate scenarios' probabilities (i.e.,  $p_{nE}=1-p_{nP}$ ) in an absolute manner. In such cases, the cumulative effort assessments, referred to intervals of probabilities, are convenient for comparison purposes even if their

distinct cumulative values are essentially meaningless. Application examples of these equations are provided in next subsection.

## 7.2.2 Example of Decision-Making Under Uncertainty

### 7.2.2.1 Decision-Making Under Partial Uncertainty

As an indicative example, supposing that  $p_{nE}$  factor is estimated to the interval of  $[0.0, \dots, 0.5]$ , the equation (7-2) returns the equation (7-4) in which  $p_{nE}$  factor has been eliminated due to the conducted integration.

$$\begin{aligned}
 c(CVP, N, M, [0.0, 0.5], [1.0, 0.5], \lambda) &= \\
 &= \int_0^{0.5} \frac{3}{2} \lambda^2 p_{nE} (1 - p_{nE}) + \lambda(1 - p_{nE})N + 2\lambda p_{nE}M + 4\lambda p_{nE} + \lambda(1 - p_{nE}) dp_{nE} = \quad (7-4) \\
 &= \frac{M\lambda}{4} + \frac{3N\lambda}{8} + \frac{\lambda^2}{8} + \frac{7\lambda}{8}
 \end{aligned}$$

Respectively, the equation (7-3) returns the equation (7-5) which returns the cumulative prediction of the required effort for all probabilities in the same interval for CIBI design combination.

$$\begin{aligned}
 c(CIBI, N, M, [0.0, 0.5], [1.0, 0.5], \lambda) &= \\
 &= \int_0^{0.5} \frac{3}{2} \lambda^2 p_{nE} (1 - p_{nE}) + 2\lambda(1 - p_{nE})N + \lambda p_{nE}M + \lambda p_{nE} + 2\lambda(1 - p_{nE}) dp_{nE} = \quad (7-5) \\
 &= \frac{M\lambda}{8} + \frac{3N\lambda}{4} + \frac{\lambda^2}{8} + \frac{7\lambda}{8}
 \end{aligned}$$

In the case of the Interpreter specific problem with design attributes  $N=40$ , and  $M=10$ , the equations (7-4), (7-5), and similar equations for CIBI design combination are simplified to equations (7-6). These equations imply a clear advantage of CVP design combination since CVP always requires the lowest cumulative maintenance effort for any number of scenarios' applications  $\lambda$ .

$$\begin{aligned}
 c(CVP, 40, 10, [0.0, \dots, 0.5], [1.0, \dots, 0.5], \lambda) &= \frac{\lambda^2 + 147\lambda}{8} \quad (7-6) \\
 c(CIBI, 40, 10, [0.0, \dots, 0.5], [1.0, \dots, 0.5], \lambda) &= \frac{\lambda^2 + 257\lambda}{8}
 \end{aligned}$$

The general equations (7-2) and (7-3) for CVP and CIBI design combinations can provide cumulative effort estimations for any interval of  $p_{nE}$  factor referring to CIBI vs CVP general problem. Thus, the generated formal models, through further integration, can sufficiently support decision making for arbitrary intervals of probabilities factors or else under partial uncertainty.

### 7.2.2.2 Decision-Making Under Full Uncertainty

Based on previous logic, decision making can be supported even under full uncertainty or else for the whole range of a probability factor. Thus, by integrating equation (7-2) and similar equation for CIBI combination on  $p_{nE}$  factor for its whole range  $[0.0, \dots, 1.0]$ , the simplified equations (7-7) are derived (again for the Interpreter specific problem with design attributes  $N=40$ , and  $M=10$ ).

$$c(CVP, 40, 10, [0, \dots, 1], [1, \dots, 0], \lambda) = \frac{\lambda^2}{4} + \frac{65\lambda}{2} \quad (7-7)$$

$$c(CIBI, 40, 10, [0, \dots, 1], [1, \dots, 0], \lambda) = \frac{\lambda^2}{4} + \frac{93\lambda}{2}$$

The equations (7-7) imply a general (under full uncertainty or any value of  $p_{nE}$  factor) advantage of CVP design combination since CVP requires the lowest cumulative maintenance effort for any number of scenarios' applications  $\lambda$ . This inference is visually confirmed by the design space representation of the CVP vs. CIBI general problem in Figure 3-18 and Figure 6-43. In Figure 6-43, all the sample instances below the trace of balance cases are suited for CVP design alternative. Respectively, all the sample instances above the trace of balance cases are suited for CIBI design alternative. For the Interpreter example  $\mu = M/N = 10/40 = 0.25$ . Thus, for  $\mu = 0.25$ , most of the interval of  $p_{nE} = [0, \dots, 1]$  is allocated below the trace of balance cases which is referred to the CVP design alternative.

However, when  $\mu$  factor is near to 1, it is difficult to visually infer the most proper design alternative in Figure 6-43. This is because, in Figure 6-43 there is no graphical representation of the cumulative required effort of an interval of  $p_{nE}$  factor. In such cases, the Figure 3-18 is more informative since the conceptual surface across z axle for a particular  $\mu$  value and  $p_{nE}$  interval is a visual representation of the cumulative required effort. Part of this surface may be below and above the trace of balance cases. The difference on the area of these contrary parts gives the most beneficial design alternative. Thus, the larger portion of this area relative to the trace of balance cases indicates the most beneficial design alternative for the specific  $\mu$  value and  $p_{nE}$  interval. Nevertheless, the graphical representation of the design space is not always an easy matter especially in the cases of difficult design problems with multiple design attributes and scenario probabilities. In such cases, the mathematical analysis through integration presented so far provides a formal and direct mean which bypasses conceptual and graphical representations. This type of calculus computations can be easily performed through several software tools such as MATLAB and Microsoft Mathematics.

### 7.2.2.3 Decision-Making Under Uncertainty with Multiple Factors of Probabilities

Based on previous logic, decision making can be supported even in the case of multiple factors of probabilities. The extended problem in subsection 5.2 with Decorator design pattern attached in the basic CVP vs. CIBI design problem is offered as a suitable example. The new design problem CVP-DP vs. CIBI-DP has three major maintenance scenarios, three design attributes, and thus tree probability factors ( $p_{nE}$ ,  $p_{nP}$ ,  $p_{nD}$ ), one for each design attribute. The extra scenario, design attribute, and probability factor ( $p_{nD}$ ) is referred to the event of adding a new decorator element into the design structure.

Again, the key concept is the integration of the formal models on the probability factor of interest. This time we need to perform repeated integrations in different probability factor each time. Since the sum of all probability factors is equal to one, one factor is related and defined by the values of the rest factors. Supposing  $n$  as the number of probability factors, the degree of freedom on setting values on these factors is  $n-1$ . Due to this property, one probability factor should be replaced and expressed by the other probabilities before the first integration round. Next, we gradually integrate for the rest probability factors, one at a time, according to the desired intervals. In the case of CVP-DP vs. CIBI-DP design problem the cumulative effort for CVP-DP design combination is given by the general equation (7-8) as derive by the equation (5-1) where  $p_{nD} = (1 - p_{nE} - p_{nP})$ .

$$c\left(CVP\_DP, N, M, D, [p_{nE(min)}, \dots, p_{nE(max)}], [p_{nP(min)}, \dots, p_{nP(max)}], \left[ (1 - p_{nE(min)} - p_{nP(min)}), \dots, (1 - p_{nE(max)} - p_{nP(max)}) \right], \lambda\right) = \quad (7-8)$$

$$= \int_{p_{nP(\min)}}^{p_{nP(\max)}} \left\{ \int_{p_{nE(\min)}}^{p_{nE(\max)}} \left( \frac{3}{2} \cdot p_{nP} \cdot ((1 - p_{nE} - p_{nP}) + p_{nE}) \cdot \lambda^2 + (p_{nP} + 2 \cdot (1 - p_{nE} - p_{nP}) \cdot (M + 2) + 2 \cdot p_{nE} \cdot (M + 2) + p_{nP} \cdot (D + N)) \lambda \right) p_{nE} \right\} p_{nP}$$

Respectively, the cumulative effort for CIBI-DP design combination is given by the general equation (7-9) as derive by the equation (5-2) where  $p_{nD}=(1-p_{nE}-p_{nP})$ .

$$c \left( \begin{array}{c} CIBI\_DP, N, M, D, [p_{nE(\min)}, \dots, p_{nE(\max)}], [p_{nP(\min)}, \dots, p_{nP(\max)}], \\ [(1 - p_{nE(\min)} - p_{nP(\min)}), \dots, (1 - p_{nE(\max)} - p_{nP(\max)})], \lambda \end{array} \right) =$$

$$= \int_{p_{nP(\min)}}^{p_{nP(\max)}} \left\{ \int_{p_{nE(\min)}}^{p_{nE(\max)}} \left( \frac{3}{2} \cdot p_{nP} \cdot ((1 - p_{nE} - p_{nP}) + p_{nE}) \cdot \lambda^2 + ((1 - p_{nE} - p_{nP}) + p_{nE} + 4 \cdot p_{nP} + 2 \cdot D \cdot p_{nP} + M \cdot (1 - p_{nE} - p_{nP}) + M \cdot p_{nE} + 2 \cdot N \cdot p_{nP}) \lambda \right) p_{nE} \right\} p_{nP} \quad (7-9)$$

For demonstration purposes, the above equations are applied and computed for the interval of scenarios probabilities  $p_{nE}=[0.0, \dots, 0.4]$  and  $p_{nP}=[0.0, \dots, 0.3]$ . The  $p_{nD}$  factor depends on the values of the rest probability factors as previously discussed. Thus, the equation (7-8) returns the equation (7-10).

$$c \left( \begin{array}{c} CVP\_DP, N, M, D, [0.0, \dots, 0.4], [0.0, \dots, 0.3], \\ [(1 - p_{nE(\min)} - p_{nP(\min)}), \dots, (1 - p_{nE(\max)} - p_{nP(\max)})], \lambda \end{array} \right) =$$

$$= \lambda \left( \frac{9D}{500} + \frac{9N}{500} + \frac{51M}{250} + \frac{213}{500} \right) + \frac{27\lambda^2}{1250} \quad (7-10)$$

Respectively, the equation (7-9) returns the equation (7-11).

$$c \left( \begin{array}{c} CIBI\_DP, N, M, D, [0.0, \dots, 0.4], [0.0, \dots, 0.3], \\ [(1 - p_{nE(\min)} - p_{nP(\min)}), \dots, (1 - p_{nE(\max)} - p_{nP(\max)})], \lambda \end{array} \right) =$$

$$= \lambda \left( \frac{9D}{250} + \frac{9N}{250} + \frac{51M}{500} + \frac{87}{500} \right) + \frac{27\lambda^2}{1250} \quad (7-11)$$

The difference of the equations (7-10) and (7-11) is given by the equation (7-12)

$$c \left( \begin{array}{c} CVP\_DP - CIBI\_DP, N, M, D, [0.0, \dots, 0.4], [0.0, \dots, 0.3], \\ [(1 - p_{nE(\min)} - p_{nP(\min)}), \dots, (1 - p_{nE(\max)} - p_{nP(\max)})], \lambda \end{array} \right) =$$

$$= \lambda \frac{51M - 9D - 9N + 126}{500} \quad (7-12)$$

The equation (7-12) is a general equation that indicates the most beneficial design alternative for the specific intervals of probability factors  $\{p_{nE}=[0.0, \dots, 0.4]$  and  $p_{nP}=[0.0, \dots, 0.3]\}$ . The sign of this equation depends on the values of the design attributes (i.e., N, M, and D). For example, by setting  $N=10$ ,  $M=20$ , and  $D=5$ , the sign of equation (7-12) is positive and, thus the CIBI-DP design alternative is preferable. In the case of  $N=40$ ,  $M=5$ , and  $D=10$ , the sign of equation (7-12) is negative and, thus the CVP-DP design alternative is preferable. Thus, even in the case of decisions under uncertainty (referred to intervals of scenarios probabilities) the derived models remain sensitive and informative regarding all their parameters (design attributes).

### 7.2.3 Additional Decision-Criteria to Decision-Making Process

The introduced modeling method implies that the selection of the most beneficial or maintainable design combination is based on a single and dominant criterion; the design alternative with the lowest required effort. Nevertheless, the decision-making process can

be further enhanced by introducing additional criteria such as worst or optimum scenario outcomes.

For instance, referring to CIBI vs. CVP general problem, consider a set of design attributes  $N$  and  $M$  for which equations (7-6) referred to the limited interval  $p_{nE}=[0.0, \dots, 0.5]$  suggests CVP design combination. In addition, suppose that equations (7-7) for the whole  $p_{nE}$  range  $[0.0, \dots, 1.0]$  suggest CIBI design combination as the most beneficial. Under this perspective, the limited interval  $p_{nE}=[0.0, \dots, 0.5]$  reflects an optimum or more desirable situation with limited uncertainty degree. Respectively, the wider range  $p_{nE}=[0.0, \dots, 1.0]$ , reflects the worst possible or less desirable situation with higher uncertainty degree. Under these circumstances, the decision-making process can be supported by the criterion of minimizing the required effort for the worst against optimum possible situation. If, for example, our decision-criterion is the selection of the most beneficial design alternative in the worst case of high uncertainty degree, then the CIBI design combination is the most appropriate option. Accordingly, if our decision-criterion is the selection of the most beneficial design alternative in the optimum case of limited uncertainty degree, then the CVP design combination is the most appropriate option.

Hence, the combination of different decision-criteria, supported by transformed formal models for arbitrary intervals of probabilities factors, provides a robust and formal background able to support decision-making among design alternatives considering several comparison perspectives.

## 7.3 Horizon Analysis

### 7.3.1 Separating Maintenance Process to Sub-Periods

In practice, attempting to estimate the change-trends of the maintenance process is not always straightforward and uniform. In some cases, the scenarios' probabilities may have different values for different periods of the software life cycle. In such cases, the analysis can be separated in distinct sub-periods, a technique usually called horizon analysis or multi-period analysis. This technique helps forecasting the realized effort over various maintaining periods or horizons where each period has different scenarios' probabilities. Under this perspective, even the code's development stage can be viewed as a preliminary maintenance period.

For instance, referring to the general problem of part-whole aggregation, the required development effort can be analyzed in two separate periods. Usually, the CP elements are implemented first, followed by the implementation of different operations through the inheritance-based implementation (IBI) or Visitor design pattern (VP) structure according to the selected design alternative. Thus, focusing on the specific instance of Interpreter with initial design attributes  $N=40$ ,  $M=10$ , and  $p_{nE}=0.5$  (during maintenance), the development period can be separated into two phases or sub-periods, one for elements' development, where  $N=0$ ,  $M=1$ ,  $p_{nE}=1$ ,  $p_{nP}=0$ , and  $\lambda=40$ , and secondly for operations' development, where  $N=40$ ,  $M=1$ ,  $p_{nE}=0$ ,  $p_{nP}=1$ , and  $\lambda=9$ . Consequently, after the end of the development period and entering the maintenance period, the design attributes are equal to the initial attributes  $N=40$ ,  $M=10$  of the maintenance process, as presented in Table 7-1. From this point and on, the maintenance horizon can also be separated in sub-periods with different scenarios' probabilities. For example, supposing that the scenarios' probabilities for the next 50 applied scenarios are estimated to  $p_{nE}=0.5$  and  $p_{nP}=0.5$ . Thus, for the first maintenance period,  $N=40$ ,  $M=10$ ,  $p_{nE}=0.5$ ,  $p_{nP}=0.5$ , and  $\lambda=50$ . Hence, after the end of the first maintenance period, the design attributes are equal to  $N=40+p_{nE}*50=65$  and  $M=10+p_{nP}*50=35$ . Now, suppose that the scenarios' probabilities for the next 50 applied scenarios are estimated to  $p_{nE}=0.9$  and  $p_{nP}=0.1$ . Thus, for the second maintenance period,  $N=65$ ,  $M=35$ ,  $p_{nE}=0.9$ ,  $p_{nP}=0.1$ , and  $\lambda=50$ . As a result, horizon analysis can provide forecasts of the total required effort, including almost all the software life cycle. Of course,

the horizon analysis can be limited or expanded to as many periods as required according to the interest of the software engineer or quality manager. The total required effort is equal to the sum of the required effort of each sub-period.

Table 7-1: Horizon Analysis Data referred to the Interpreter Specific Instance of CVP vs. CIBI General Problem

Sub-period (horizon)	Initial design attributes		Scenarios' probabilities		Scenarios applications
	N	M	$p_{nE}$	$p_{nP}$	$\lambda$
Development of elements (types or methods or classes)	0	1 <sup>i</sup>	1.0	0.0	40
Development of operations (types or methods or classes)	40	1	0.0	1.0	9
Early maintenance	40 <sup>ii</sup>	10 <sup>ii</sup>	0.5 <sup>iii</sup>	0.5	50
Later maintenance	65	35	0.9 <sup>iii</sup>	0.1	50
End of maintenance (software life cycle)	110	40	-	-	-

<sup>i</sup> As a starting state referred to development period, the initial design attribute of operations' variable ( $M$ ) starts from value one; otherwise, the Formal Model equations return zero effort

<sup>ii</sup> Initial design attributes, referred to the maintenance period, based on the scope of the Interpreter example:  $N=40$ ,  $M=10$

<sup>iii</sup> Estimations of scenarios probabilities during maintenance:  $p_{nE}=0.5$  (early period) and  $p_{nE}=0.9$  (later period)

### 7.3.2 Example of Decision-Making Supported by Horizon Analysis

The generated formal model equations can be applied for each sub-period in Table 7-1, where the returned effort per design alternative is progressively added to the returned effort of the previous sub-period. The results can be graphically presented in a unified plot, as demonstrated in Figure 7-1. The graph in Figure 7-1 provides excellent insight about the required effort per design alternative for different periods (horizons) of the software life cycle. Looking at the graph in Figure 7-1, it seems that from a long-term perspective, there is no significant difference between the total efforts of CVP and CIBI design alternatives. If the decision is based on a short-term perspective, you should choose the CVP design combination. However, if the selection criterion is focused only on a long-term aspect, which in the case of the interpreter is a more realistic orientation, then the CIBI design combination seems to have better perspectives. But be aware, in this case, the management should suffer the extra wasted effort until the time where the CIBI combination becomes the most maintainable option.

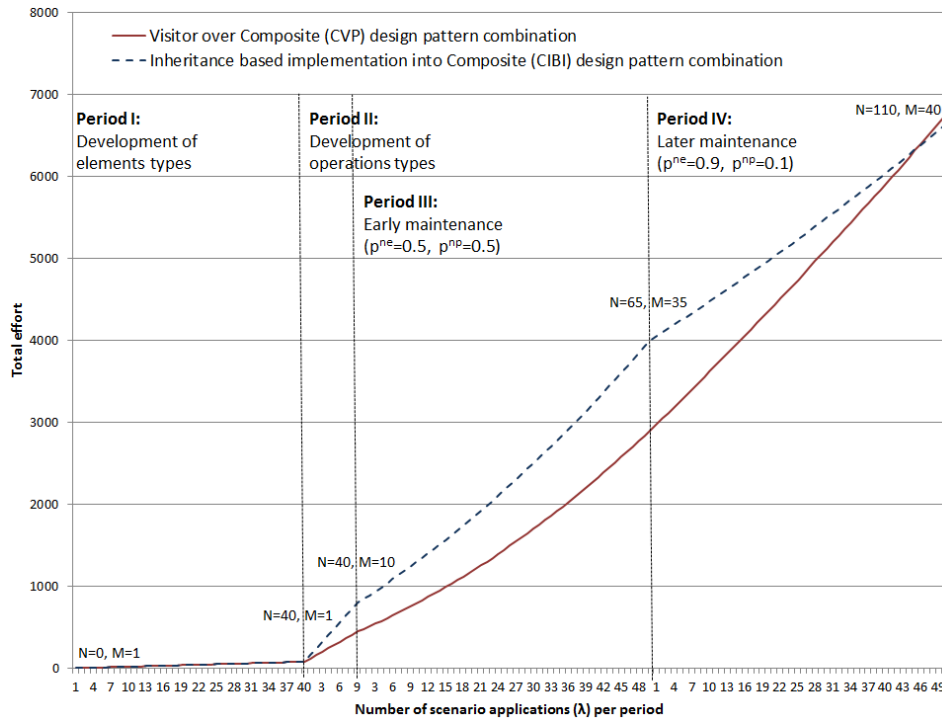


Figure 7-1: Graph of total effort per design alternative (CVP vs. CIBI) for different horizons (sub-periods) of the software lifecycle, including preliminary development stage, referred to the example of Interpreter requirements with initial design attributes during maintenance  $N=40$  and  $M=10$

In such cases, where the preferable design options are interchanged across different periods, the management should consider other investment-oriented criteria such as time-value or present-value of money and opportunity cost or cost of capital. More specifically, in Figure 7-1 example, the extra wasted effort of CIBI implies that extra salaries and expenditures should be invested in a short to mid-term horizon. Even if CIBI combination is more maintainable (requires less effort) from a long-term perspective, the cost of the short to mid-term (extra or wasted) used capital may exceed the long-term benefit. For instance, referred to the beginning of the later maintenance period in Figure 7-1, the CIBI combination requires approximately 35% more effort (or expenditures or invested capital, respectively) than CVP combination. If during that particular period the cost of capital (e.g., borrowing interest rate, or possible rate of return of the opportunity to invest in another more profitable asset) to finance the project is too high, then it may be better for management to select CVP combination, keeping the short to mid-term capital requirements in low levels and hoping that in a long-term perspective financing conditions would be improved.

Hence, horizon analysis opens a whole spectrum of different aspects and criteria such as investing, economics, and financing, regarding the evaluation and selection among design alternatives for various sub-periods of software lifecycle.

## 7.4 Alternate Computer-Aided Implementation of Formal Models

The derived formal models can be alternatively implemented through computer-aided simulations. Even though such implementations are limited only to a single general problem, it can provide further insight regarding each factor's contribution. In addition, through such simulation techniques, the exploration of the dynamic behavior and evolution patterns of the formal models and their components can be conducted. Furthermore, results from different simulation settings can be stored and compared for



further analysis and interpretation such as sensitivity analysis. In addition, a simulation can be run in live mode, meaning that the user can change the parameter’s values in real-time while numerical and graphical outcomes are simultaneously updated. Since the derived formal models is based on separate integration levels, visual simulation tools like VENSIM®, supporting change rates, and integrated variables (named ‘levels’), are very convenient. Moreover, VENSIM® tool is offered as a mean for conducting both continuous and discrete (event-driven) simulations.

### 7.4.1 Discrete Implementation of Models

The quantitative analysis of the CIBI vs. CVP general problem, presented in chapter 3, implies the discrete application of major maintenance scenarios (i.e., new element and operation) upon design alternatives based on their individual probabilities (i.e.,  $p_{nE}$ ,  $p_{nO}$ ). Next, for each implemented scenario the required effort is computed through the fundamental SMC metrics in terms of number of required (method and class) interventions.

A computer-aided implementation of the discrete application of individual maintenance scenarios based on SMC metrics, referred to CIBI vs. CVP general problem, is presented in Figure 7-2. Each integrated value is presented as “Levels” inside rectangles, where related change rates are presented as double-lined arrows. In addition, the interrelations between variables are indicated through simple arrows. The design of the model provides an understanding of separate integration levels and the implication of SMC metrics. Notice that the integration of the factors N, M, and total CIBI/CVP effort is discrete (event-driven) as analyzed by the introduced quantitative analysis in chapter 3.

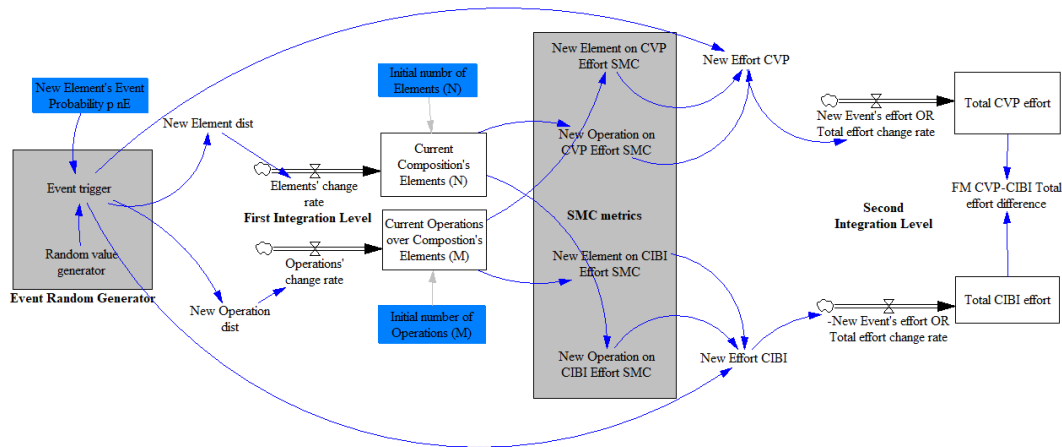


Figure 7-2: Computer-Aided implementation of the discrete Formal Model for the general decision problem CIBI vs CVP, using VENSIM® tool.

By setting initial values in the constant variables N, M, and  $p_{nE}$ , the implementation in Figure 7-2 can simulate and compute all the intermediate and final outcomes for the specific instance of the CVP vs CIBI general problem. The integration of the N, M, and CIBI/CVP total effort values, presented as “Levels” inside rectangles, are calculated through discrete integration in terms of number of scenario applications. Thus, the number of scenario applications has been defined as model’s time or integration parameter. Notice that the probability factor  $p_{nE}$  ( $p_{nO}=1- p_{nE}$ ) affects the event trigger variable which in turn directly and simultaneously affects both integration levels concerning design attribute values N and M and total CVP and CIBI effort values. The code implementation of the model in Figure 7-2 as generated by VENSIM tool is presented in Listing 7-1.

Listing 7-1: Code documentation of CVP vs CIBI general problem in VENSIM implementation

```

115. ===== Definition of model's time representation or integration parameter
116. INITIAL TIME = 0           Units: Day           The initial time for the simulation, 1 day is equal to 1 scenario application
117. TIME STEP = 1             Units: Day [1,*]        The time step for the simulation
118. SAVEPER = TIME STEP      Units: Day [1,*]        The frequency with which output is stored
119. FINAL TIME = 200         Units: Day           The final time for the simulation
120. ===== Definition of model's constant parameters or initial values
121. "Initial number of Elements (N)"= 15           Units: Element [1,200]
122. "Initial number of Operations (M)"= 14         Units: Operation [1,150]
123. New Element's Event Probability p_nE= 0.7     Units: probability [0,1]
124. ===== Discrete / Random event generator
125. Random value generator= RANDOM UNIFORM(0, 1 , 0)   Units: probability
126. Event trigger= IF THEN ELSE( Random value generator<New Element's Event Probability p_nE, 11 , 21 )   Units: probability
127. ===== First level of integration concerning Design attribute values
128. New Element dist= IF THEN ELSE(Event trigger<20, 1, 0)   Units: Element
129. New Operation dist= IF THEN ELSE(Event trigger>20, 1, 0)   Units: Operation
130. Elements' change rate= New Element dist               Units: Element/Day
131. Operations' change rate= New Operation dist           Units: Operation/Day
132. "Current Composition's Elements (N)"= INTEG (Elements' change rate, "Initial number of Elements (N)")   Units: Element
133. "Current Operations over Composition's Elements (M)"= INTEG (Operations' change rate, "Initial number of Operations (M)")
    Units: Operation
134. ===== Definition of SMC metrics
135. New Element on CVP Effort SMC= "Current Operations over Composition's Elements (M)" + "Current Operations over Composition's
    Elements (M)"                                           Units: intervention
136. New Operation on CVP Effort SMC="Current Composition's Elements (N)" + 1   Units: intervention
137. New Element on CIBI Effort SMC= "Current Operations over Composition's Elements (M)" + 1   Units: intervention
138. New Operation on CIBI Effort SMC= "Current Composition's Elements (N)" + "Current Composition's Elements (N)" Units: intervention
139. ===== Second level of integration concerning Total effort values per design alternative
140. New Effort CVP= IF THEN ELSE(Event trigger<20, New Element on CVP Effort SMC, New Operation on CVP Effort SMC)
    Units: intervention
141. New Effort CIBI= IF THEN ELSE(Event trigger<20, New Element on CIBI Effort SMC, New Operation on CIBI Effort SMC)
    Units: intervention
142. New Event's effort OR Total effort change rate = New Effort CVP           Units: interventions/Day
143. "-New Event's effort OR Total effort change rate" = New Effort CIBI       Units: interventions/Day
144. Total CVP effort= INTEG (New Event's effort OR Total effort change rate, 0)   Units: intervention
145. Total CIBI effort= INTEG ("-New Event's effort OR Total effort change rate", 0)   Units: intervention
146. ===== Computation of total effort difference among design alternatives
147. "FM CVP-CIBI Total effort difference" = Total CVP effort - Total CIBI effort   Units: intervention

```

Referring to Listing 7-1, the model has been set to integrate its variables in the time interval of 0 to 200 with a step equal to 1 reflecting by this way the discrete application of maintenance scenarios. The event random generator defines the ‘Event trigger’ variable based on scenarios probabilities, thus synchronizing the discrete event driven application of maintenance scenarios. The ‘Event trigger’ variable simultaneously and uniformly affects both levels of integration and both design alternatives. The randomly generated sequence of events (based on scenario probability) is applied for both design alternatives, thus providing the ideal comparison conditions among design alternatives. It is important that because of the random nature of the generated sequence of events, different simulations provide different effort estimations, thus the model’s outcomes are not deterministic. In practice, this implementation is a non-deterministic simulation model of the formal models in equations (3-19), (3-20), and (3-21) incorporating a single stochastic or random factor related to different sequences of events.

## 7.4.2 Continuous Implementation of Formal Models

An alternate computer-aided implementation of the derived formal models in subsections 4.3 and 4.4, referred to CIBI vs. CVP general problem, is presented in Figure 7-3. Each integrated value is presented as “Levels” inside rectangles, where related change rates are presented as double-lined arrows. In addition, the interrelations between variables are indicated through simple arrows. The design of the model provides an understanding of separate integration levels and the implication of SMC metrics. Notice that integration of the factors N, M, and total CIBI/CVP effort is continuous as proposed by the introduced modeling method in chapter 4.

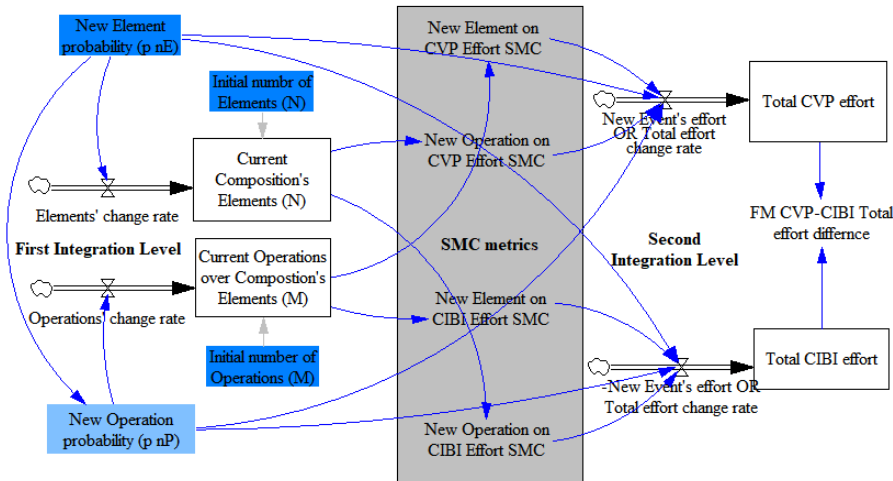


Figure 7-3: Computer-Aided implementation of the continuous Formal Model for the general decision problem CIBI vs CVP, using VENSIM® tool.

By setting initial values in the constant variables  $N$ ,  $M$ , and  $p_{nE}$ , the implementation in Figure 7-3 can simulate and compute all the intermediate and final outcomes for the specific instance of the CVP vs CIBI general problem. The integration of the  $N$ ,  $M$ , and CIBI/CVP total effort values, presented as “Levels” inside rectangles, are calculated through continuous integration in terms of number of scenario applications. Thus, the number of scenario applications has been defined as model’s time or integration parameter. Notice, that the probability factors  $p_{nE}$  and  $p_{nP}$  are directly and simultaneously affect both integration levels concerning design attribute values  $N$  and  $M$  and total CVP and CIBI effort values. The code implementation of the model in Figure 7-3 as generated by VENSIM tool is presented in Listing 7-2.

Listing 7-2: Code documentation of CVP vs CIBI Formal Models in VENSIM implementation

```

148. ===== Definition of model's time representation or integration parameter
149. INITIAL TIME = 0           Units: Day           The initial time for the simulation, 1 day is equal to 1 scenario application
150. TIME STEP = 1             Units: Day [1,*]       The time step for the simulation
151. SAVEPER = TIME STEP      Units: Day [1,*]       The frequency with which output is stored
152. FINAL TIME = 200         Units: Day           The final time for the simulation
153. ===== Definition of model's constant parameters or initial values
154. "Initial number of Elements (N)"= 15           Units: Element [1,200]
155. "Initial number of Operations (M)"= 14         Units: Operation [1,150]
156. "New Element Probability (p_nE)"= 0.7         Units: probability [0,1]
157. "New Operation Probability (p_nP)" = 1 - "New Element Probability (p_nE)"           Units: probability [0,1]
158. ===== First level of integration concerning Design attribute values
159. Elements' change rate= 1 * "New Element Probability (p_nE)"           Units: Element/Day
160. Operations' change rate= 1 * "New Operation Probability (p_nP)"           Units: Operation/Day
161. "Current Composition's Elements (N)"= INTEG (Elements' change rate, "Initial number of Elements (N)")           Units: Element
162. "Current Operations over Composition's Elements (M)"= INTEG (Operations' change rate, "Initial number of Operations (M)")
    Units: Operation
163. ===== Definition of SMC metrics
164. New Element on CVP Effort SMC= "Current Operations over Composition's Elements (M)" + "Current Operations over Composition's
    Elements (M)"           Units: intervention
165. New Operation on CVP Effort SMC="Current Composition's Elements (N)" + 1           Units: intervention
166. New Element on CIBI Effort SMC= "Current Operations over Composition's Elements (M)" + 1           Units: intervention
167. New Operation on CIBI Effort SMC= "Current Composition's Elements (N)" + "Current Composition's Elements (N)" Units: intervention
168. ===== Second level of integration concerning Total effort values per design alternative
169. New Event's effort OR Total effort change rate = ("New Element probability (p_nE)" * New Element on CVP Effort SMC) +
    ("New Operation probability (p_nP)" * New Operation on CVP Effort SMC)           Units: interventions/Day
170. "-New Event's effort OR Total effort change rate" = ("New Element probability (p_nE)" * New Element on CIBI Effort SMC) +
    ("New Operation probability (p_nP)" * New Operation on CIBI Effort SMC)           Units: interventions/Day
171. Total CVP effort= INTEG (New Event's effort OR Total effort change rate, 0)           Units: intervention
172. Total CIBI effort= INTEG ("-New Event's effort OR Total effort change rate", 0)           Units: intervention
173. ===== Computation of total effort difference among design alternatives
174. "FM CVP-CIBI Total effort difference" = Total CVP effort - Total CIBI effort           Units: intervention

```

Referring to Listing 7-2, the model has been set to integrate its variables in the time interval of 0 to 200 with a step equal to 1. It is important that due to continuous integration based on deterministic values, different simulations provide identical effort estimations, thus the model’s outcomes are deterministic. In practice, this implementation is a deterministic simulation model of the formal models in equations (4-11), (4-12), and (4-13) without the incorporation of any stochastic or random factor.

### 7.4.3 Discrete Implementation of Simulation Model

An alternate computer-aided implementation of the event-driven simulation model in chapter 6, referred to CIBI vs. CVP general problem, is presented in Figure 7-4. Again, each integrated value is presented as “Levels” inside rectangles, where related change rates are presented as double-lined arrows. In addition, the interrelations between variables are indicated through simple arrows. The design of the model provides an understanding of separate integration levels and the implication of SMC metrics. Notice that integration of the factors N, M, and total CIBI/CVP effort is discrete or event-driven as proposed by the introduced simulation model chapter 6.

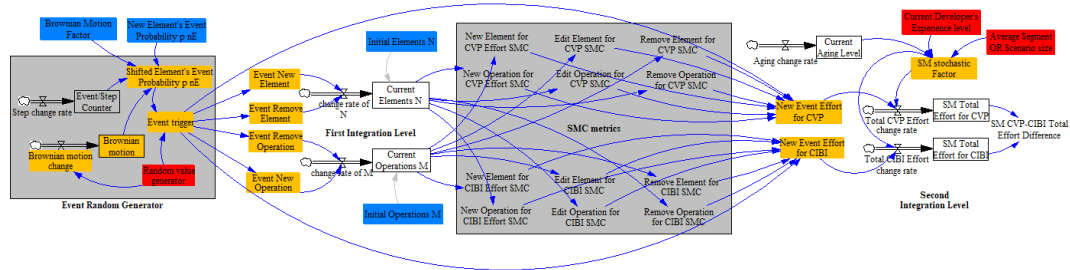


Figure 7-4: Computer-Aided implementation of the event-driven Simulation Model for the general decision problem CIBI vs CVP, using VENSIM® tool.

By setting initial values in the constant variables N, M,  $p_{nE}$ , and Brownian motion factor, the implementation in Figure 7-4 can simulate and compute all the intermediate and final outcomes for the specific instance of the CVP vs CIBI general problem. The integration of the N, M, and CIBI/CVP total effort values, presented as “Levels” inside rectangles, are calculated through discrete integration in terms of number of scenario applications. Thus, the number of scenario applications has been defined as model’s time or integration parameter. Notice that the probability factor  $p_{nE}$  ( $p_{nP}=1- p_{nE}$ ) affects the event trigger variable which in turn directly and simultaneously affects both integration levels concerning design attribute values N and M and total CVP and CIBI effort values. In particular, the model in Figure 7-4 represents the 7<sup>th</sup> fully stochastic simulation state in Table 6-4. The Brownian motion factor defines the uncertainty level or else the impact on Brownian motion level which shifts the values of scenario probability ( $p_{nE}$ ) during maintenance process. Thus, the event random generator takes under considerations the uncertainty regarding the initial estimation of scenarios’ probabilities. Furthermore, the effort change rates in the second integration level are affected by the overall stochastic factor which in turn is affected by the aging level and the randomly generated values of average scenario’s size and developers experience level. In addition, the model incorporates alternate maintenance scenarios such as edit or remove existing elements or operations. For that purpose, the intermediate variables of SMC metrics have been enhanced for each alternate scenario as defined in Table 6-4, while the change rates of the design attributes are increased, or remain unchanged, or decreased accordingly. The code implementation of the model in Figure 7-4 as generated by VENSIM tool is presented in Listing 7-3.

Listing 7-3: Code documentation of Simulation Model for CVP vs CIBI general problem in VENSIM implementation

```

175. ===== Definition of model's time representation or integration parameter
176. INITIAL TIME = 0           Units: Day           The initial time for the simulation, 1 day is equal to 1 scenario application
177. TIME STEP = 1             Units: Day [1,*]        The time step for the simulation
178. SAVERPER = TIME STEP     Units: Day [1,*]        The frequency with which output is stored
179. FINAL TIME = 200         Units: Day           The final time for the simulation
180. ===== Definition of model's constant parameters or initial values
181. "Initial Elements (N)"= 15           Units: Element [1,200]
182. "Initial Operations (M)"= 14         Units: Operation [1,150]
183. New Element's Event Probability p_nE= 0.7   Units: probability [0,1]
184. Brownian Motion Factor= 1           Units: **undefined**
185. ===== Event-driven or Random event generator
186. Random value generator= RANDOM UNIFORM(0, 1, 11)           Units: probability
187. Brownian motion change= IF THEN ELSE (Random value generator<0.5, -1, 1)   Units: **undefined**
188. Brownian motion= INTEG (Brownian motion change, 0)         Units: **undefined**
189. Step change rate= 1                                         Units: Scenario application/Day
190. "Event/Step Counter"= INTEG (Step change rate, 0)          Units: **undefined**
191. Shifted Element's Event Probability p_nE= New Element's Event Probability p_nE + Brownian Motion Factor *
    (Brownian motion / (10 * SQRT("Event/Step Counter" + 1) ) )   Units: probability
192. Event trigger= IF THEN ELSE( Random value generator < Shifted Element's Event Probability p_nE * 0.55, 11 ,
    IF THEN ELSE( Random value generator < Shifted Element's Event Probability p_nE * 0.95, 12 ,
    IF THEN ELSE( Random value generator < Shifted Element's Event Probability p_nE * 1.00, 13 ,
    IF THEN ELSE( Random value generator < Shifted Element's Event Probability p_nE +
    (1- Shifted Element's Event Probability p_nE) * 0.55, 21 ,
    IF THEN ELSE( Random value generator < Shifted Element's Event Probability p_nE +
    (1- Shifted Element's Event Probability p_nE) * 0.95, 22 , 23 ))))   Units: **undefined**
193. ===== First level of integration concerning Design attribute values
194. Event New Element= IF THEN ELSE(Event trigger=11, 1, 0)     Units: Element
195. Event Remove Element= IF THEN ELSE(Event trigger=13, -1, 0)   Units: Element
196. Event Remove Operation= IF THEN ELSE(Event trigger=23, -1, 0)   Units: Operation
197. Event New Operation= IF THEN ELSE(Event trigger=21, 1, 0)     Units: Operation
198. change rate of N= Event New Element + Event Remove Element   Units: Element
199. change rate of M= Event New Operation + Event Remove Operation   Units: Operation
200. Current Elements N= INTEG (change rate of N, Initial Elements N)   Units: Element
201. Current Operations M= INTEG (change rate of M, Initial Operations M)   Units: Operation
202. ===== Definition of SMC metrics for all possible maintenance scenario types per design alternative
203. New Element for CVP Effort SMC= Current Operations M + Current Operations M   Units: Intervention
204. New Operation for CVP Effort SMC= Current Elements N + 1   Units: Intervention
205. New Element for CIBI Effort SMC= Current Operations M + 1   Units: Intervention
206. New Operation for CIBI Effort SMC= Current Elements N + Current Elements N   Units: Intervention
207. Edit Element for CVP SMC= Current Operations M + Current Operations M   Units: Intervention
208. Edit Operation for CVP SMC= Current Elements N + 1   Units: Intervention
209. Edit Element for CIBI SMC= Current Operations M + 1   Units: Intervention
210. Edit Operation for CIBI SMC= Current Elements N + Current Elements N   Units: Intervention
211. Remove Element for CVP SMC= Current Operations M + 1   Units: Intervention
212. Remove Operation for CVP SMC= 1 + 0 * Current Elements N   Units: Intervention
213. Remove Element for CIBI SMC= 1 + 0 * Current Operations M   Units: Intervention
214. Remove Operation for CIBI SMC= Current Elements N + 1   Units: Intervention
215. ===== Definition of stochastic factor for effort estimations
216. Aging change rate= 1/ FINAL TIME           Units: **undefined**
217. Current Aging Level= INTEG (Aging change rate, 1)           Units: **undefined**
218. Current Developer's Experience level= RANDOM NORMAL(0.1, 2, 1.5, 0.33, 100)   Units: **undefined**
219. Average Segment OR Scenario size= RANDOM NORMAL(0, 2, 1, 0.33, 200)   Units: **undefined**
220. SM stochastic Factor= (Average Segment OR Scenario size * Current Aging Level) / Current Developer's Experience level
221. ===== Second level of integration concerning Total effort values per design alternative
222. New Event Effort for CVP= IF THEN ELSE(Event trigger=11, New Element for CVP Effort SMC,
    IF THEN ELSE(Event trigger=12, Edit Element for CVP SMC,
    IF THEN ELSE(Event trigger=13, Remove Element for CVP SMC,
    IF THEN ELSE(Event trigger=21, New Operation for CVP Effort SMC,
    IF THEN ELSE(Event trigger=22, Edit Operation for CVP SMC, Remove Operation for CVP SMC))))   Units: Intervention
223. New Event Effort for CIBI= IF THEN ELSE(Event trigger=11, New Element for CIBI Effort SMC,
    IF THEN ELSE(Event trigger=12, Edit Element for CIBI SMC,
    IF THEN ELSE(Event trigger=13, Remove Element for CIBI SMC,
    IF THEN ELSE(Event trigger=21, New Operation for CIBI Effort SMC,
    IF THEN ELSE(Event trigger=22, Edit Operation for CIBI SMC, Remove Operation for CIBI SMC))))   Units: Intervention
224. Total CVP Effort change rate= SM stochastic Factor * New Event Effort for CVP   Units: Intervention
225. Total CIBI Effort change rate= SM stochastic Factor * New Event Effort for CIBI   Units: Intervention
226. SM Total Effort for CVP= INTEG ( Total CVP Effort change rate, 0)   Units: Intervention
227. SM Total Effort for CIBI= INTEG ( Total CIBI Effort change rate, 0)   Units: Intervention
228. ===== Computation of total effort difference among design alternatives
229. "SM CVP-CIBI Total Effort Difference"= SM Total Effort for CVP - SM Total Effort for CIBI   Units: Intervention

```

Referring to Listing 7-3, the model has been set to integrate its variables in the time interval of 0 to 200 with a step equal to 1 reflecting by this way the discrete application of maintenance scenarios. The event random generator defines the ‘Event trigger’ variable based on scenarios probabilities and Brownian motion factor, thus synchronizing the

discrete event driven application of maintenance scenarios. The ‘Event trigger’ variable simultaneously and uniformly affects both levels of integration and both design alternatives. The randomly generated sequence of events (based on the shifted scenario probability) is applied for both design alternatives. Furthermore, the randomly generated sequence of values of the stochastic factor is applied for both design alternatives (in the second integration level), thus providing the ideal comparison conditions among design alternatives. It is important that because of the random nature of the generated sequence of events and stochastic factor, different simulations provide different effort estimations, thus the model’s outcomes are not deterministic. In practice, this implementation is a non-deterministic representation of the simulation model in chapter 6 incorporating several stochastic or random factors such as different sequences of events, shifted probabilities, alternate maintenance scenarios, average scenario’s size, and developers experience level.

#### 7.4.4 Comparison of Implementations’ Outcomes

In this subsection, a comparison of the outcomes of all previous implementations in VENSIM tool is presented. More specifically, the Discrete Formal Model in subsection 7.4.1 as DFM, the continuous Formal Model in subsection 7.4.2 as FM, and the event-driven simulation model in subsection 7.4.3 as SM have been merged in one single model. The random value generators of DFM and SM have been synchronized, thus providing identical sequences of applied scenarios through the ‘event trigger’ variable for both event-driven models. The number of scenario applications has been defined as model’s time or integration parameter with typical time interval between 0 and 200. All initial constant parameters ( $N$ ,  $M$ ,  $p_{nE}$ ) are commonly defined in all models and referred to the indicative example of GUI implementation ( $N=15$ ,  $M=14$ ,  $p_{nE}=0.7$ ) in Table 1-1 and Table 3-4. The Brownian motion factor has been set to 1.00 reflecting the high uncertainty level of the 7<sup>th</sup> fully stochastic simulation state in Table 6-4. The analysis concentrates on the values of total effort outcomes, change rates, and intermediate variables for all models’ implementations. Keep in mind that DFM and SM are non-deterministic models, thus their results are only an instance of a specific simulation.

The difference of the total effort among CVP and CIBI design alternatives is presented in Figure 7-5. FM outcome demonstrates a steady trend because of the continuous integration and its formal and deterministic behavior. DFM and SM outcomes demonstrate fluctuate trends because of the discrete and event-driven integration and their stochastic and non-deterministic behavior. However, DFM fluctuations follows the SM trend since DFM incorporates only one stochastic factor related to different sequences of applied scenarios. In contrast, SM fluctuations significantly deviates from the SM and DFM trends since SM demonstrates highly stochastic behavior by incorporating several stochastic factors related to different sequences of applied scenarios, shifted scenarios’ probabilities, alternate maintenance scenarios, average scenario’s size, and developers experience level. In the simulation instance presented in Figure 7-5, the SM outcome confirms the design decision based on FM and DFM outcomes. However, in the case of a different simulation instance the SM outcome may reject the design decision based on FM and DFM outcomes, thus indicating a wrong design decision of FM as analyzed in subsection 6.5. To measure the error rate of a possible wrong decision, several repeated or Monte Carlo simulations are required per sample insurance. Even though VENSIM tool provides capabilities for comparison of results among different simulations, the conduction of massive, repeated simulations is out of the scope of this tool.

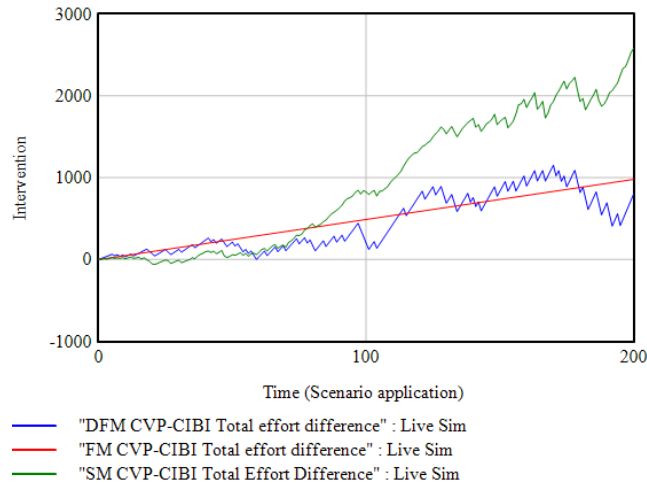


Figure 7-5: Difference of total effort among CVP and CIBI design alternatives for DFM, FM, and SM models, for the GUI implementation ( $N=15$ ,  $M=14$ ,  $p_{nE}=0.7$ )

The total effort of CVP and CIBI design alternatives is presented in Figure 7-6. Again, FM outcome demonstrates a steady trend because of the continuous integration and its formal and deterministic behavior. DFM and SM outcomes demonstrate fluctuate trends because of the discrete and event-driven integration and their stochastic and non-deterministic behavior. Once again, DFM fluctuations follows the SM trend since DFM incorporates only one stochastic factor related to different sequences of applied scenarios. In contrast, SM fluctuations significantly deviates from the SM and DFM trends since SM demonstrates highly stochastic behavior by incorporating several stochastic factors related to different sequences of applied scenarios, shifted scenarios' probabilities, alternate maintenance scenarios, average scenario's size, and developers experience level. In general, SM returns reduced values due to the incorporation of the alternate maintenance scenarios which decelerate the maintenance process, and thus reduce the overall required effort.

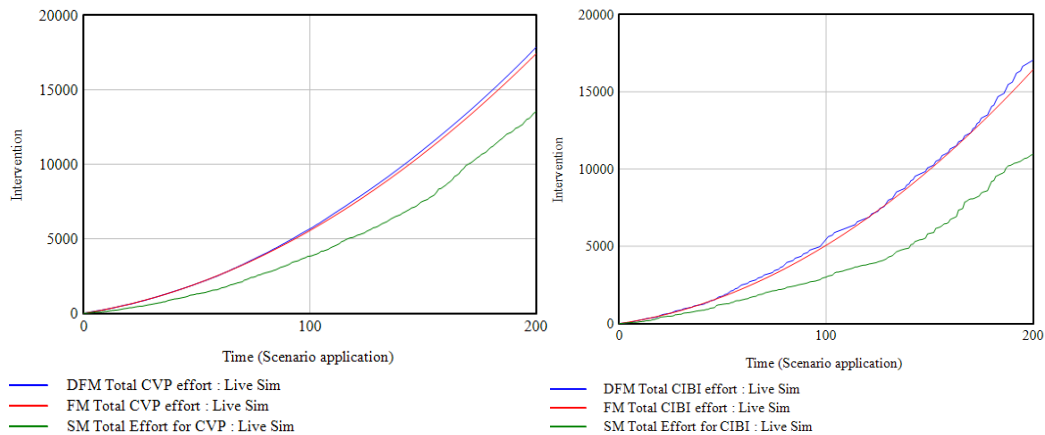


Figure 7-6: Total effort of CVP (left) and CIBI (right) design alternatives for DFM, FM, and SM models, for the GUI implementation ( $N=15$ ,  $M=14$ ,  $p_{nE}=0.7$ )

The stochastic factor of the SM and its components (average scenario size, code aging level, and developers experience level) are presented in Figure 7-7. The fluctuations of the stochastic factor reflect the highly uncertain nature of the SM total effort outcomes. As expected, the most uncertain factors that affect the overall stochastic factor are the average scenario size and developers experience level.

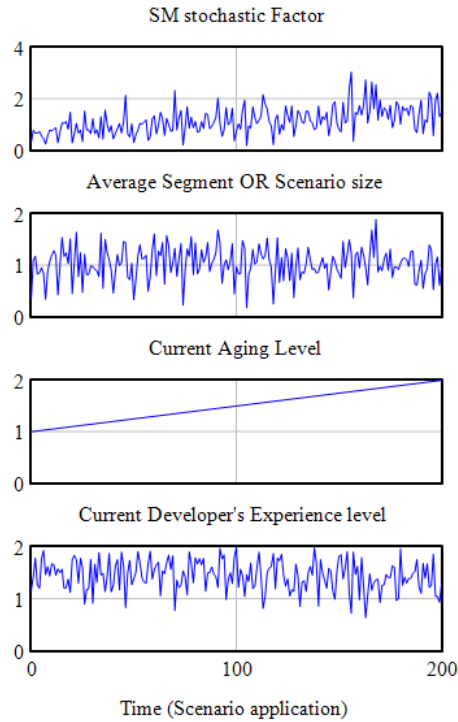


Figure 7-7: Analysis of Stochastic Factor components of SM model, for the GUI implementation ( $N=15$ ,  $M=14$ ,  $p_{nE}=0.7$ )

The change rates of the total effort of CVP and CIBI design alternatives is presented in Figure 7-8. Again, FM change rate demonstrates a steady trend because of the continuous integration and its formal and deterministic behavior. DFM and SM change rates demonstrate fluctuate trends because of the discrete and event-driven integration and their stochastic and non-deterministic behavior. Once again, DFM fluctuations follows the SM trend since DFM incorporates only one stochastic factor related to different sequences of applied scenarios. In contrast, SM fluctuations deviates from the SM and DFM trends since SM demonstrates highly stochastic behavior by incorporating several stochastic factors related to different sequences of applied scenarios, shifted scenarios' probabilities, alternate maintenance scenarios, average scenario's size, and developers experience level. On average, SM returns reduced change rate values due to the incorporation of the alternate maintenance scenarios which decelerate the maintenance process, and thus reduce the required effort per scenario application.

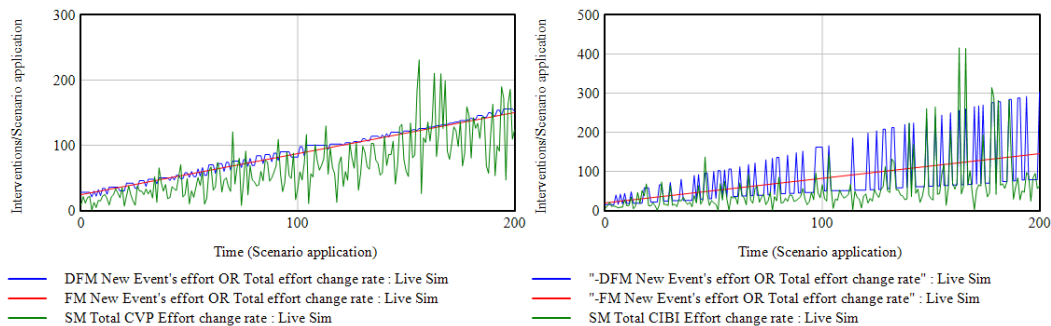


Figure 7-8: Effort change rate of CVP (left) and CIBI (right) design alternatives for DFM, FM, and SM models, for the GUI implementation ( $N=15$ ,  $M=14$ ,  $p_{nE}=0.7$ )

From a different perspective, the change rates of the total effort of the simulation model (SM) in Figure 7-8 represent the sequence of required effort per scenario application or else



the time series of required effort values during maintenance process. These time series are characterized by highly fluctuations or noise and their statistical parameters (i.e., mean, standard deviation, kurtosis, skewness) reflect the statistical characteristics of the required effort values per scenario application. The manipulation of those statistical parameters to approximate the empirical observations from field studies and time series analysis (G. Antoniol et al., 2001; Raja et al., 2009; Shariat Yazdi et al., 2016) offers a reliable mean for calibrating the model’s outcomes to imitate the actual maintenance process, as analyzed in subsection 6.4.9.

The design attributes of elements (N) and operations (M) are presented in Figure 7-9. Again, FM outcome demonstrates a steady trend because of the continuous integration and its formal and deterministic behavior. DFM and SM outcomes demonstrate fluctuate trends because of the discrete and event-driven integration and their stochastic and non-deterministic behavior. Once again, DFM fluctuations follows the SM trend since DFM incorporates only one stochastic factor related to different sequences of applied scenarios. In contrast, SM fluctuations significantly deviates from the SM and DFM trends since SM demonstrates highly stochastic behavior by incorporating several stochastic factors related to different sequences of applied scenarios, shifted scenarios’ probabilities, and alternate maintenance scenarios. In general, SM returns reduced values due to the incorporation of the alternate maintenance scenarios (edit and remove) which decelerate the maintenance process, and thus reduce the increment rate of the design attributes.

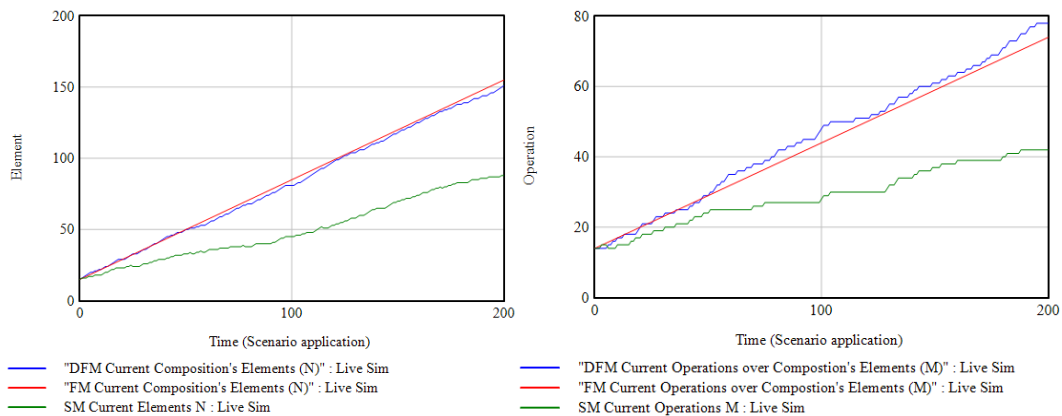


Figure 7-9: Design attributes of Elements N (left) and Operations M (right) for DFM, FM, and SM models, for the GUI implementation (N=15, M=14,  $p_{nE}=0.7$ )

The scenario probability ( $p_{nE}$ ) and shifting scenario probability of the simulation model (SM) are presented in Figure 7-10. The Brownian motion level and factor cause the shifting of scenario probability value during maintenance process. This situation reflects the real circumstances of actual maintenance process and the difficulty to predict in advance the exact value of scenario probability. The shifting value of scenario probability reflects the highly uncertain nature of the SM.

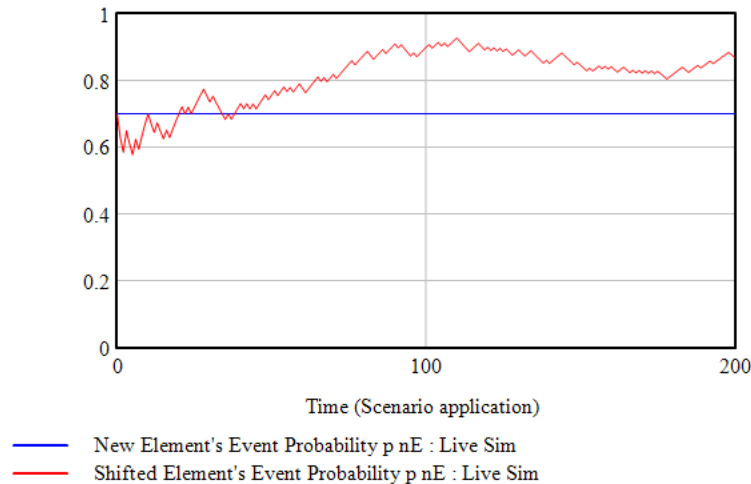


Figure 7-10: New element scenario probability ( $p_{nE}$ ) and shifted scenario probability of SM model, for the GUI implementation ( $N=15$ ,  $M=14$ ,  $p_{nE}=0.7$ )

## 7.5 Seeing Software Design as Investment

Designing high-quality software that satisfies user's requirements is of primary interest among competing businesses in the software market. Most of the novel and widely used software systems in the market are the result of costly projects supported by different experts' teams for long periods until their retirement. Under this perspective, a software system can be perceived by the company's management as a new product in the context of an investment plan. Furthermore, during the design stage of software (product), each design alternative is also an investment alternative with different pros and cons. The evaluation and selection among these alternatives can be made based on different quality criteria such as maintainability, which heavily influences the degree of success and future maintenance cost of the system (product). Thus, more maintainable software is related to lower maintenance costs, which increases the product's useful life and profitability. Each time that a software engineer selects a specific design alternative (investment), he misses the opportunity to use a different – possibly better or more maintainable - design alternative (investment).

### 7.5.1 Cost of Missed Opportunities

The introduced modeling method of this thesis provides reliable formal models that can help software engineers to evaluate different design alternatives (investments) in a formal way regarding their maintainability perspective. The decision-making is based on the minimum estimated (required maintenance) effort among compared design alternatives. For example, selecting CIBI instead of CVP design alternative means that CVP corresponds to either the missed opportunity (maintenance) cost of the taken decision or the best (second most maintainable) available alternative or the next lowest cost that a designer could have achieved if he had selected CVP instead of CIBI. The difference between CVP and CIBI predictions can be interpreted as the (positive) net maintenance cost, reflecting the gained (or avoided wasted) maintenance effort as a consequence of the decision-making process supported by the introduced modeling method.

### 7.5.2 Accounting Perspective

The introduced modeling method and possibly generated formal models help in designing software based on formal quality principles regarding its maintainability degree, hence delaying its expected obsolescence and increasing its useful life. From an accounting perspective, this is defiantly a desirable outcome when the software is intended to be used by a company for internal use. In this case, the development cost is capitalized and

amortized for a longer period with a decreased annual amortization cost, thus improving software usefulness and the company's profitability. However, software that is developed for sale or lease to others is capitalized and amortized (only) after it has reached technological feasibility (K. R. Subramanyam, 2013). Before that stage of development, the software is considered to be Research & Development and it is expensed accordingly. Nevertheless, formal quality principles such as generated formal models can help with accelerating the achievement of software's technological feasibility. Only the one-time cost associated with the initial formal model generation stage (for significant designing problems) is conceived as Research & Development expenses.

## 7.6 Conclusions

The introduced modeling theory and derived formal modes provide a reliable tool for supporting decision-making among design alternatives in respect to their maintainability for significant problems as discussed in previous chapters. Given that the derived formal modes are easily applied and reusable for several instances of each general problem, this chapter discusses major alternate perspectives and possible ways for applying these formal models through other techniques taking advantages of their versatility and mathematical formality.

The representation of design alternatives based on key maintenance scenarios and design attributes through alternate computer-aided simulations extends the analysis of each general and significant problem. Such simulation software and tools help to reveal and explore several aspects concerning the impact of maintenance process on each design alternative. Comparative and intermediate results and real-time sensitivity analysis among different simulation settings and parameters provide further insight about factors' contribution and cause-effect relationships of the problem under study.

The mathematical reformation of the formal models provides a great mean for supporting design decisions under partial or full uncertainty, allowing decision-making when scenarios' probabilities are difficult or even impossible to be estimated in a precise manner. Furthermore, the horizon analysis technique allows the application of the derived formal models across long development and maintenance periods, separated in distinct subperiods with different assumptions and parameters. Both techniques highlight and extend the practicability and the usability of the introduced theory and derived formal modes to even more practical, complicated, and realistic conditions.

Each of the alternate application perspectives discussed in this chapter can be further investigated based on the same or similar principles and techniques concerning similar or other significant design problems in the field of software architecture and engineering.

## 8 Conclusions and Future Work

### 8.1 Contribution of Dissertation

Software architecture design includes several critical decisions with significant impact on the pursued quality attribute of maintainability. Such early design decisions heavily affect related time/effort/cost of software maintenance, updates, and modifications since future maintenance activities should adapt to the initially selected architecture.

The introduced approach in this thesis suggests a documented modeling method that generates probabilistic comparison models that estimate the maintainability degree of design alternatives through effort predictions in a formal and deterministic way. The theoretical foundation of the proposed modeling method and the results of relevant works provide strong evidence that the derived formal models return proportional and reliable effort estimations mainly for comparison purposes among design alternatives. By using the proposed modeling model, specific characteristics of given design problems are considered allowing selection of proper design pattern combinations at an early stage of the design process, before code development. Software quality metrics can be derived directly from design descriptions of well-known object-oriented design pattern combinations. This approach resolves conflicting pro and cons among design alternatives with regards to their maintainability degree. At the same time, it limits the ambiguity imposed by the stochastic nature of the actual maintenance process by relying on a limited set of problem's parameters such as design attributes and probabilities of major maintenance scenarios.

The analysis of the derive formal comparison models, concerning the general design problem of recursive hierarchies of part-whole aggregations, indicates that different design pattern combinations have a significant effect on software quality properties like maintainability. Furthermore, the progressive and probabilistic analysis verify the same significant effect on future modifications during software maintenance. Also, it is indicated that probability analysis over maintenance scenarios has a decisive role on maintenance cost/effort assessment. Models sensitive to design attributes and scenarios probabilities are particularly informative regarding the maintainability degree of each design alternative. In addition, seeing software evolution under the view of differential analysis by integrating change rates of problem's design attributes is an innovation in the pursuing of realistic comparison models.

Evaluation results using extended design problems such as Observer vs. Mediator indicate that the proposed method can be also applied during the high-level architecture design, to handle communication, interfacing, and coordination issues among sub-modules and components of new or even legacy code. These examples of alternate design problems prove the usability and applicability of the proposed modeling method in a wide spectrum of common, difficult, and frequently tackled design problems in the field of software architecture. The application of the proposed approach reveals the usability and extensibility of the suggested method considering different metrics, design implementations, and quality characteristics. Furthermore, the introduced models can be easily implemented in software allowing the assessment of the average beneficial contribution of the method in terms of rate of avoided wasted effort by evaluating the entire design space of each general problem under analysis.

The results of the extensive statistical validation indicate that the evaluated formal models provide reliable estimations of the expected effort, especially for comparison purposes. The reliability of the evaluated probabilistic models increases in a mid-to-long term perspective, and thus, as the maintenance process evolves and decisions' benefits become more significant, the models' decision ability to conclude in the most beneficial design alternative in terms of maintainability is increased. Simulation evidence suggests that simplified modeling approaches such as the introduced modeling method are

particularly reliable able to approximate dynamic system's behavior mostly because of mid-to-long-term statistical convergence. As the maintenance process evolves, and despite the various stochastic and random factors affecting it, the average long-term effect of these factors would be eventually negligible, and thus the predictions of the required effort are increasingly driven by the standard and recurring structural behavior of the used design patterns. Furthermore, decisions concerning design alternatives exhibit very limited selection-risk even under high uncertainty levels regarding the initial estimation of problem's parameters. Thus, the introduced modeling approach and derived formal comparison models manage to eliminate transitory and biased factors to enhance mid-to-long-term decision ability, avoiding significant amounts of wasted maintenance effort and relevant costs.

Methods that yield such reliable, formal, general, and reusable models reduce the long-term uncertainty of design decisions and help developers and engineers to elevate the quality of their decision-making, optimize the design process, and develop more maintainable software. Predicting future levels of maintenance effort is very important for comparison purposes and early structural analysis can significantly improve such assessments, practically reversing the (cone of) uncertainty levels in software project management. This thesis suggests that even if such early and critical design decisions is not the primary concern in software industry, developers, designers, and project/quality manager should turn their attention on them since the cumulative benefits (avoided wasted effort) from the repeating use of formal models significantly overcomes their initial derivation cost. It is expected that the emphasis in maintainability assessment will be shifted from static analysis of code to dynamic analysis of software design structure for future change flows. The proposed approach in this thesis introduces a new perception of software evolution during maintenance, while promotes early decision-making culture in software quality control management.

Finally, this thesis introduces a new conception about the statistical evaluation of formal comparison models and relevant theories regarding their reliability to support design decisions. It relies on massive and homogeneous validation data, sensitive to several design attributes, generated by widely stochastic simulation models which have been thoroughly calibrated to replicate the underlying activities and variability of actual software evolution during maintenance process. Researchers are encouraged and hopefully inspired to possibly apply the introduced concepts in similar or different context.

## 8.2 Future Work

This thesis targets on stimulating and motivating research community toward the evaluation of other general, significant, and frequently tackled design problems in software architecture domain for which proper selection among object-oriented design alternatives could be modeled through the provided theoretical framework and statistically validated by simulated observations. Towards this perspective, future research efforts can engage different or alternate design patterns and aspects under different measurement methods and units, or even for different software quality characteristics. Furthermore, the proposed approach can be combined and analyzed in conjunction with other methods on case studies to determine the relationship between deterministic effort measurements and external quality factors such as reusability, maintainability, testability, and adaptability. In addition, the proposed mathematical approach can be used for evaluating and comparing similar or different design patterns especially in the field of Pattern Languages of Programs.

An interesting research perspective could be the mathematical analysis of the derived formal comparison models. A such analysis could reveal possible similarities, differences, conflicts, requirements, limitations, and patterns regarding the evolution and structural behavior of the used design patterns, thus providing further insight about the maintenance perspective of each design alternative under comparison, or even about the nature the addressed general problem itself.

At the same time, this thesis is a starting point for further research in the domain of software evolution throughout simulation models which could engage other stochastic factors in different frequency distributions and intervals under other measurement methods and units. Furthermore, further research efforts towards a refined calibration of the introduced simulation model through more informative datasets of real-world observations are encouraged. These research perspectives could be assisted by using general or targeted purpose simulation languages and tools. In addition, machine learning or artificial intelligence techniques may be used to assist simulation model reaching even more realistic results. Moreover, the simulation of alternate design problems can be assisted by integrating the proposed simulation model in this thesis to perform jointly as a synthesized simulation or co-simulation. In such cases, existing base models might serve as lumped components of a broader simulation model forming a hierarchical structure.

These research perspectives highlight the possible usability of the proposed theoretical framework and the introduced evaluation method through simulations in a wide spectrum of general and difficult designing problems in the fields of design optimization and software architectural design.

## Appendix A: Mat Lab Modeling Framework

```

%% CODE A.1 ==MODELING FRAMEWORK (DYNAMIC DIFFERENTIAL ANALYSIS AND FORMAL MODEL GENERATION)====
% Data describing (general) comparison problem (CIBI vs CVP) and fundamental (SMC) metric analysis
% In this section, different or alternate problems should be described
D = {'CVP','CIBI'}; % tags of Design combinations under comparison
L_tags = {'N','M'}; % tags of design attributes: N initial elements, M initial operations
A = {'Method aspect', 'Class aspect'}; % tags of Structural aspects
S = {'nE', 'nP'}; % tags of Types of maintenance scenarios: nE new composition element, nP new operation
F = [1 0; 0 1]; % N:+1 and M:+0 for nE, N:+0 and M:+1 for nP (change rates of affected design attributes for each scenario
type |S|x|L|)
% SMC metric factors on design attributes L, for each design combination D, scenario S, and aspect A are stated in K array
K = zeros([size(D,2) size(S,2) size(A,2) size(L_tags,2)+1]); % creates empty matrix with dimensions:
|D|x|S|x|A|x|L|+1
%          method class (structural aspects)
% D S          N M - N M -
K(1,1,:) = [0 1 2; 0 1 2]; % on CVP for a nE : totally 0N+1M+0 method + 0N+1M+0 class interventions = 2(M+2)
K(1,2,:) = [1 0 0; 0 0 1]; % on CVP for a nP : totally 1N+0M+0 method + 0N+0M+1 class interventions = N+1
K(2,1,:) = [0 1 0; 0 0 1]; % on CIBI for a nE : totally 0N+1M+0 method + 0N+0M+1 class interventions = M+1
K(2,2,:) = [1 0 1; 1 0 1]; % on CIBI for a nP : totally 1N+0M+0 method + 1N+0M+0 class interventions = 2(N+1)

%%====DIFFERENTIAL ANALYSIS CODE=====
% Differential analysis is implemented progressively by using symbolic expressions
% Design attributes |L|, individual scenarios probabilities |P|, and number of scenario applications t or λ, are passed as
parameters to the derived equations
syms t; % creates (single) symbolic variable/parameter for time or the number of scenario applications (λ=t)
L_sym = sym(L_tags); % creates (many) symbolic variables/parameters for all design attributes |L|
% creates (many) symbolic variables/parameters for all individual scenarios probabilities |P|
% symbolic parameters are prefixed/named with p_ index for consistency
for i=1:size(S,2)
    P_sym(i) = sym(strcat('p_', S{i}));
end

% First level of integration for each design attribute of L set
for i=1:size(L,2)
    l(i) = 0*t; % sets initial value as symbolic expression
    for j=1:size(S,2)
        l(i) = l(i) + P_sym(j)*F(j,i);
    end
    l(i) = int(l(i), t) + L_sym(i); % differential equation with initial condition l(0)=L(i) returns C = L(i)
end

% Second level of integration for each design combination of D set
for i=1:size(D,2)
    for j=1:size(S,2)
        for q=1:size(A,2)
            for g=1:size(L,2)
                cost_L(g) = K(i,j,q,g)*l(g); % distinct products of current design attributes l(g)
            end
            cost_AL(q) = sum(cost_L(:)) + K(i,j,q,size(L,2)+1); % sub-cost for all design attributes L(g)
        end
        cost_SAL(j) = sum(P_sym(j) * cost_AL(:)); % sub-cost for all structural aspects A(q) [and all attributes L(g)]
    end
    % differential equation with initial condition cost_DSAL(0)=0 returns C=0
    % total-cost for all scenario's types S(j) and [all structural aspects A(q) and design attributes L(g)]
    cost_DSAL(i) = int(sum(cost_SAL(:)),t);
end

% Converts symbolic expressions to an array of anonymous function handles
for i=1:size(D,2)
    FM_cost_D{i} = matlabFunction(cost_DSAL(i)); % function of total-cost for each design combination D
end
% Converts symbolic expressions to a symbolic function returning multiple results for each design combination |D|
FM_cost_D_f(L_sym, P_sym, t)=cost_DSAL;

%%====
% Equations of total progressive cost for each design combination (use these equations for further analysis)
% cost_DSAL(i) is a symbolic expression
% FM_cost_D_f is a symbolic function returning an array of computations for all D combinations (parameters are normally
arranged as L_sym, P_sym, t) - RECOMMENDED for direct use
% FM_cost_D{i} is an anonymous function handle for direct computations for each D combination (attention, parameters are
arbitrary arranged)
%%====

```

```

%% CODE A.2 ===== PLOTTING RESULTS IN A SINGLE GRAPH (PRESENTATION EXAMPLE)=====
% Data (design attributes) derived from specifications of a specific system (instance of general problem)
% In this section, the design attributes of a specific system are placed (multiple attributes can be declared as arrays)
L = [40 10]; % initial values of design attributes N and M
P = [0.5 0.5]; % individual probabilities of each scenario type |S|

%%=====PLOTTING CODE =====
lt = 5:5:100; % declares the array of the interval of interest
merged_parameters_values = {}; % merges L(i) and P(i) values of parameters in a single array of cells
for i=[1:size(L_tags,2)]
    merged_parameters_values{i} = L(i);
end
for i=[1:size(P,2)]
    merged_parameters_values{size(L_tags,2)+i} = P(i);
end
% merged_parameters_values are aligned according to FM_cost_D_f symbolic function's declaration
% Uses the FM_cost_D_f symbolic function (returning an array of computations for all D) and computes total effort/cost for
 $\lambda = [5:5:100]$ 
for i=[1:20]
    G_lines(i,:) = FM_cost_D_f(merged_parameters_values{:}, i*5);
end
% adds different lines for each design combination D
plot (lt, G_lines, 'MarkerSize',4,'Marker','square');
grid on;
title('Total progressive effort predictions per design combination');
xlabel('Number of scenario applications ( $\lambda$ ));
ylabel('Total progressive effort');
legend(D(:), 'Location', 'northwest');
hold off;
savefig('CIBIvsCVP_FigureFile');
%=====

```



```

%% CODE B.1 ===MODELING FRAMEWORK (DYNAMIC DIFFERENTIAL ANALYSIS AND FORMAL MODEL GENERATION)===
% Data describing (general, extended) comparison problem CIBI_DP vs CVP_DP and fundamental (SMC) metric analysis
% In this section, different or alternate problems should be described
D = {'CVP-DP','CIBI-DP'}; % tags of Design comb. under comparison
L_tags = {'N','M','D'}; % tags of design attributes: N initial elements, M initial operations, and V initial decorators
A = {'Method aspect','Class aspect'}; % tags of Structural aspects
S = {'nE','nP','nD'}; % Types of maintenance scenarios: nE new composition element, nP new operation, nDE new decorator element, nDP new decorator operation
F = [1 0 0; 0 1 0; 0 0 1]; % N:+1 M:+0 D:+0 for nE, N:+0 and M:+1 D:+0 for nP, N:+0 M:+0 D:+1 for nD (change rates of affected design attributes for each scenario type |S|x|L|)
% SMC metric factors on design attributes L, for each design combination D, scenario S, and aspect A are stated in K array
K = zeros([size(D,2) size(S,2) size(A,2) size(L_tags,2)+1]);%creates empty matrix dimensions: |D|x|S|x|A|x|L|+1
%          method      class      (structural aspects)
% D S          N M D - N M D -
K(1,1,:,:) = [0 1 0 2; 0 1 0 2]; %CVP-DP for nE: 0N+1M+0D+2 method + 0N+1M+0D+2 class = 2(M+2)
K(1,2,:,:) = [1 0 1 0; 0 0 0 1]; %CVP-DP for nP: 1N+0M+1D+0 method + 0N+0M+1D+1 class = N+D+1
K(1,3,:,:) = [0 1 0 2; 0 1 0 2]; %CVP-DP for nD: 0N+1M+0D+2 method + 0N+1M+0D+2 class = 2(M+2)

K(2,1,:,:) = [0 1 0 0; 0 0 0 1]; %CIBI-DP for nE: 0N+1M+0D+0 method + 0N+0M+0D+1 class = M+1
K(2,2,:,:) = [1 0 1 2; 1 0 1 2]; %CIBI-DP for nP: 1N+0M+1D+2 method + 1N+0M+1D+2 class = 2(N+1)+2(D+1)
K(2,3,:,:) = [0 1 0 0; 0 0 0 1]; %CIBI-DP for nD: 0N+1M+0D+0 method + 0N+0M+0V+1 class = M+1
%% =====
% Differential analysis and formal model generation code has been parameterized based on set matrices and thus, it is common and reusable for any set of parameters

%% CODE B.2 ===== PLOTTING RESULTS IN A SINGLE GRAPH (PRESENTATION EXTENDED EXAMPLE)=====
% Data (design attributes) derived from specifications of a specific system (instance of general extended problem)
% In this section, the design attributes of a specific system are placed (multiple attributes can be declared as arrays)
L = [40 10 30]; % initial values of design attributes N, M, and D
P = [0.25 0.25 0.5]; % individual probabilities of each scenario type |S|
%% =====
% Plotting code has been parameterized based on set matrices and thus, it is common and reusable for any set of parameters

```

```

%% CODE C.1 === MODELING FRAMEWORK (DYNAMIC DIFFERENTIAL ANALYSIS AND FORMAL MODEL GENERATION) ===
% Data describing (general, extended) comparison problem CIBI_MP vs CVP_MP vs CIBI_OP vs CVP_OP and fundamental (SMC)
metric analysis
% In this section, different or alternate problems should be described
D = { 'CIBI and MP', 'CVP and MP', 'CIBI and OP', 'CVP and OP' }; % tags of Design comb. under comparison
L_tags = {'N', 'M', 'C'}; % tags of design attributes: N initial elements, M initial operations, C initial mediators or observers
A = {'Method aspect', 'Class aspect'}; % tags of Structural aspects
S = {'nE', 'nP', 'nM'}; % Types of maintenance scenarios: nE new composition element, nP new operation, nM new mediator
or observer
F = [1 0 0; 0 1 0; 0 0 1]; % N:+1 M:+0 C:+0 for nE, N:+0 M:+1 C:+0 for nP, N:+0 M:+0 C:+1 for nM (change rates of
affected design attributes for each scenario type |S|x|L|)
% SMC metric factors on design attributes L, for each design combination D, scenario S, and aspect A are stated in K array
K = zeros([size(D,2) size(S,2) size(A,2) size(L_tags,2)+1]); % creates empty matrix dimensions: |D|x|S|x|A|x|L|+1
%          method   class   (structural aspects)
% D S      N M C - N M C -
K(1,1,,:) = [0 1 1 1; 0 0 1 2]; %CIBI_MP for nE: 0N+1M+1C+1 method + 0N+0M+1C+1 class = M+2C+3
K(1,2,,:) = [1 0 0 1; 1 0 0 1]; %CIBI_MP for nP: 1N+0M+0C+1 method + 1N+0M+0C+1 class = 2(N+1)
K(1,3,,:) = [0 0 0 2; 0 0 0 1]; %CIBI_MP for nM: 0N+0M+0C+2 method + 0N+0M+0C+1 class = 3

K(2,1,,:) = [0 1 1 3; 0 1 1 3]; %CIBI_MP for nE: 0N+1M+1C+3 method + 0N+1M+1C+3 class = 2M+2C+6
K(2,2,,:) = [1 0 0 0; 0 0 0 1]; %CIBI_MP for nP: 1N+0M+0C+0 method + 0N+0M+0C+1 class = N+1
K(2,3,,:) = [0 0 0 2; 0 0 0 1]; %CIBI_MP for nM: 0N+0M+0C+2 method + 0N+0M+0C+1 class = 3

K(3,1,,:) = [0 1 0 2; 0 0 0 2]; %CVP_OP for nE : 0N+1M+0C+2 method + 0N+0M+0C+2 class = M+4
K(3,2,,:) = [1 0 0 1; 1 0 0 1]; %CVP_OP for nP : 1N+0M+0C+1 method + 1N+0M+0C+1 class = 2(N+1)
K(3,3,,:) = [0 0 0 3; 0 0 0 2]; %CVP_OP for nM : 0N+0M+0C+3 method + 0N+0M+0C+2 class = 5

K(4,1,,:) = [0 1 0 4; 0 1 0 3]; %CVP_OP for nE : 0N+1M+0C+4 method + 0N+1M+0C+3 class = 2M+7
K(4,2,,:) = [1 0 0 0; 0 0 0 1]; %CVP_OP for nP : 1N+0M+0C+0 method + 0N+0M+0C+1 class = N+1
K(4,3,,:) = [0 0 0 3; 0 0 0 2]; %CVP_OP for nM : 0N+0M+0C+3 method + 0N+0M+0C+2 class = 5
%%
% Differential analysis and formal model generation code has been parameterized based on set matrices and thus, it is common
and reusable for any set of parameters

%% CODE C.2 ===== PLOTTING RESULTS IN A SINGLE GRAPH (PRESENTATION EXTENDED EXAMPLE) =====
% Data (design attributes) derived from specifications of a specific system (instance of general extended problem)
% In this section, the design attributes of a specific system are placed (multiple attributes can be declared as arrays)
L = [15 14 10]; % initial values of design attributes N, M, and C
P = [0.1 0.2 0.7]; % individual probabilities of each scenario type |S|
%%
% Plotting code has been parameterized based on set matrices and thus, it is common and reusable for any set of parameters

```

# Appendix B: Sample Data of General Problem

Parameters of Sample Instances of the CVP vs. CIBI General Problem

n.	N	M	p <sub>NE</sub>	e <sub>XP</sub>	n.	N	M	p <sub>NE</sub>	e <sub>XP</sub>	n.	N	M	p <sub>NE</sub>	e <sub>XP</sub>
1.	161	60	0,67	1,84	335.	192	124	0,52	0,46	669.	161	116	0,91	2,00
2.	56	90	0,32	0,26	336.	104	17	0,32	1,99	670.	198	57	0,23	1,35
3.	162	119	0,35	1,08	337.	163	60	0,48	0,85	671.	21	72	0,15	0,42
4.	56	123	0,09	0,99	338.	137	23	0,64	0,77	672.	65	70	0,55	0,29
5.	103	94	0,89	0,24	339.	96	123	0,85	1,07	673.	88	49	0,47	0,87
6.	101	12	0,90	0,65	340.	155	100	0,65	0,55	674.	67	30	0,32	1,30
7.	31	7	0,15	0,26	341.	135	15	0,58	1,04	675.	144	24	0,35	0,32
8.	134	112	0,15	1,20	342.	33	47	0,94	1,49	676.	40	47	0,35	1,51
9.	104	38	0,08	1,69	343.	179	35	0,92	0,53	677.	41	76	0,10	1,40
10.	81	6	0,70	1,76	344.	144	111	0,25	1,52	678.	131	44	0,71	0,71
11.	90	58	0,72	1,80	345.	175	48	0,72	0,63	679.	99	42	0,11	1,59
12.	29	102	0,34	1,51	346.	127	31	0,77	1,43	680.	47	46	0,57	1,75
13.	130	146	0,06	1,21	347.	185	82	0,29	1,22	681.	52	92	0,33	0,85
14.	188	133	0,14	0,24	348.	199	127	0,58	1,36	682.	148	33	0,27	0,64
15.	79	51	0,17	1,67	349.	32	35	0,61	0,59	683.	130	14	0,38	1,77
16.	155	91	0,22	0,32	350.	23	100	0,39	0,47	684.	86	18	0,08	0,45
17.	21	5	0,26	1,66	351.	169	89	0,32	1,93	685.	47	114	0,76	1,83
18.	139	132	0,93	1,73	352.	143	148	0,55	1,47	686.	127	60	0,77	1,89
19.	89	126	0,30	1,86	353.	41	63	0,11	1,47	687.	26	8	0,63	1,06
20.	156	138	0,68	1,94	354.	101	146	0,83	0,43	688.	131	44	0,61	1,60
21.	70	99	0,57	1,44	355.	145	43	0,69	0,41	689.	60	84	0,86	1,47
22.	160	13	0,90	0,53	356.	175	49	0,30	1,95	690.	176	112	0,50	0,42
23.	142	69	0,09	0,90	357.	169	84	0,53	0,41	691.	122	105	0,19	1,51
24.	46	89	0,63	1,33	358.	94	33	0,76	0,72	692.	78	68	0,31	1,82
25.	156	55	0,30	1,98	359.	91	39	0,78	0,75	693.	113	148	0,12	0,90
26.	137	78	0,74	1,61	360.	46	113	0,09	1,50	694.	31	30	0,14	0,98
27.	106	82	0,18	1,13	361.	65	82	0,88	0,72	695.	26	97	0,69	0,35
28.	108	39	0,23	1,61	362.	51	95	0,14	0,31	696.	190	35	0,77	0,42
29.	141	24	0,94	0,86	363.	142	125	0,07	1,04	697.	181	134	0,79	0,93
30.	102	35	0,12	1,90	364.	70	70	0,78	1,52	698.	78	76	0,76	1,80
31.	26	33	0,85	1,74	365.	77	33	0,26	1,00	699.	101	11	0,47	0,36
32.	154	99	0,46	0,68	366.	135	144	0,15	1,59	700.	86	136	0,33	0,29
33.	22	68	0,85	1,68	367.	193	149	0,65	0,47	701.	167	39	0,82	1,38
34.	130	58	0,50	0,70	368.	88	91	0,55	0,23	702.	175	147	0,91	0,55
35.	155	9	0,35	0,36	369.	113	131	0,73	0,30	703.	41	119	0,73	1,19
36.	182	111	0,48	1,97	370.	141	147	0,66	1,86	704.	69	74	0,35	1,89
37.	196	41	0,87	1,95	371.	181	56	0,25	1,95	705.	172	117	0,94	0,48
38.	166	103	0,46	1,40	372.	143	54	0,84	0,84	706.	194	96	0,07	1,07
39.	66	139	0,40	1,83	373.	64	101	0,91	0,33	707.	144	47	0,30	1,46
40.	139	134	0,61	1,58	374.	76	13	0,28	1,16	708.	137	29	0,33	1,05
41.	48	89	0,20	1,60	375.	24	27	0,15	0,32	709.	189	15	0,12	1,65
42.	129	114	0,37	1,25	376.	108	59	0,83	1,99	710.	118	127	0,09	0,42
43.	182	119	0,62	1,89	377.	137	139	0,81	0,71	711.	80	125	0,42	0,26
44.	92	29	0,75	0,92	378.	79	117	0,23	1,15	712.	85	38	0,50	1,24
45.	166	51	0,10	0,69	379.	166	101	0,90	1,21	713.	99	69	0,88	0,51
46.	121	133	0,89	0,86	380.	45	98	0,51	1,18	714.	161	29	0,73	0,29
47.	154	108	0,90	1,97	381.	144	19	0,58	0,62	715.	193	47	0,79	0,52
48.	35	133	0,61	0,52	382.	108	54	0,17	0,45	716.	150	114	0,82	0,22
49.	66	90	0,64	1,15	383.	193	92	0,50	0,77	717.	182	96	0,41	1,89
50.	43	76	0,28	0,99	384.	145	15	0,87	1,29	718.	40	28	0,69	1,06
51.	39	138	0,35	0,74	385.	118	142	0,41	1,01	719.	199	98	0,61	1,81
52.	157	118	0,50	1,59	386.	172	92	0,33	1,20	720.	31	136	0,47	1,85
53.	160	139	0,92	1,22	387.	51	123	0,41	1,95	721.	33	112	0,82	1,14
54.	30	150	0,29	0,44	388.	189	72	0,39	0,46	722.	46	115	0,60	1,51
55.	21	130	0,07	0,83	389.	155	11	0,56	1,29	723.	95	18	0,47	1,67
56.	162	126	0,64	1,58	390.	135	102	0,07	1,38	724.	132	66	0,57	0,55
57.	60	137	0,46	1,45	391.	92	125	0,46	0,89	725.	185	56	0,16	1,91
58.	67	140	0,66	0,58	392.	174	103	0,18	0,77	726.	169	30	0,70	1,76
59.	186	16	0,17	0,55	393.	191	19	0,43	1,06	727.	176	75	0,43	0,50
60.	64	142	0,78	1,80	394.	46	35	0,62	1,13	728.	70	120	0,91	1,30
61.	176	20	0,09	1,85	395.	56	84	0,68	0,57	729.	85	76	0,83	1,33
62.	62	112	0,30	1,34	396.	26	131	0,73	0,93	730.	101	131	0,16	1,62
63.	79	139	0,37	1,72	397.	84	23	0,09	1,04	731.	98	22	0,16	0,86
64.	87	124	0,71	1,39	398.	135	63	0,17	0,62	732.	154	79	0,82	1,09
65.	45	123	0,36	0,80	399.	102	53	0,06	0,47	733.	109	15	0,92	1,50
66.	125	119	0,50	0,66	400.	68	140	0,74	1,87	734.	150	73	0,62	1,61
67.	58	39	0,44	1,08	401.	121	55	0,40	0,45	735.	101	89	0,26	0,96
68.	63	128	0,77	1,89	402.	85	6	0,09	1,79	736.	194	75	0,20	0,91
69.	103	118	0,47	0,55	403.	99	142	0,12	1,86	737.	66	93	0,09	0,74
70.	81	78	0,81	0,50	404.	108	54	0,41	1,09	738.	162	106	0,29	1,11
71.	147	66	0,77	1,92	405.	51	37	0,14	1,66	739.	93	74	0,24	0,31
72.	195	90	0,77	1,49	406.	107	42	0,70	1,69	740.	159	89	0,17	1,88
73.	179	21	0,92	1,20	407.	154	36	0,68	0,79	741.	139	58	0,79	0,97
74.	119	13	0,75	0,83	408.	164	54	0,11	1,88	742.	174	118	0,17	0,77
75.	161	37	0,63	1,40	409.	123	109	0,41	0,55	743.	120	35	0,83	0,61
76.	175	92	0,95	1,18	410.	57	43	0,36	1,05	744.	27	29	0,19	1,23

n.	N	M	p <sub>NE</sub>	Exp	n.	N	M	p <sub>NE</sub>	Exp	n.	N	M	p <sub>NE</sub>	Exp
77.	46	134	0,10	1,19	411.	47	135	0,28	0,95	745.	46	100	0,45	1,03
78.	146	134	0,29	1,65	412.	78	131	0,72	0,57	746.	131	120	0,55	1,01
79.	113	132	0,82	0,30	413.	110	70	0,45	1,36	747.	197	15	0,24	1,63
80.	70	105	0,63	1,02	414.	106	141	0,22	0,51	748.	95	49	0,93	1,95
81.	63	35	0,10	1,17	415.	166	21	0,64	1,97	749.	186	79	0,43	1,20
82.	96	130	0,54	1,73	416.	137	42	0,83	0,20	750.	156	142	0,13	0,64
83.	186	38	0,55	0,76	417.	128	42	0,06	1,94	751.	182	128	0,43	0,59
84.	41	31	0,13	0,62	418.	82	14	0,14	0,49	752.	186	66	0,36	0,79
85.	60	110	0,77	0,86	419.	169	93	0,64	0,44	753.	193	41	0,54	1,33
86.	103	58	0,75	1,34	420.	59	50	0,78	0,82	754.	121	21	0,89	0,65
87.	26	54	0,66	0,38	421.	155	99	0,37	1,09	755.	104	61	0,26	0,44
88.	181	103	0,07	1,97	422.	122	140	0,90	0,69	756.	154	86	0,34	0,62
89.	153	20	0,10	1,07	423.	105	111	0,58	0,46	757.	183	110	0,41	1,10
90.	161	65	0,25	1,92	424.	138	99	0,73	1,79	758.	28	115	0,94	0,52
91.	155	147	0,07	1,23	425.	98	19	0,62	1,22	759.	146	118	0,60	1,41
92.	102	98	0,47	0,63	426.	149	53	0,49	1,53	760.	123	149	0,20	1,07
93.	152	39	0,25	0,96	427.	185	80	0,85	0,50	761.	95	55	0,77	1,08
94.	83	66	0,08	1,97	428.	94	49	0,93	1,10	762.	32	93	0,31	0,40
95.	87	41	0,53	0,90	429.	145	148	0,69	1,00	763.	86	46	0,73	1,58
96.	141	46	0,67	0,31	430.	147	109	0,72	0,92	764.	187	40	0,32	0,85
97.	111	105	0,77	0,59	431.	167	72	0,05	1,90	765.	75	109	0,56	1,58
98.	189	48	0,43	0,95	432.	137	58	0,37	0,27	766.	102	50	0,43	0,25
99.	197	57	0,66	0,36	433.	61	68	0,58	0,54	767.	199	28	0,06	1,73
100.	117	133	0,45	0,25	434.	122	55	0,95	0,43	768.	132	75	0,27	0,64
101.	191	74	0,77	1,21	435.	190	123	0,35	1,22	769.	41	75	0,73	1,83
102.	140	136	0,35	0,74	436.	81	123	0,91	0,62	770.	37	69	0,44	1,90
103.	29	31	0,79	1,45	437.	144	113	0,61	1,00	771.	164	111	0,46	0,84
104.	73	90	0,70	0,51	438.	75	55	0,62	0,62	772.	131	129	0,16	0,48
105.	151	15	0,51	0,58	439.	125	60	0,09	1,49	773.	61	88	0,89	1,96
106.	169	112	0,15	1,10	440.	73	80	0,26	1,34	774.	25	58	0,80	0,98
107.	43	69	0,11	1,52	441.	76	117	0,61	1,38	775.	135	64	0,06	0,71
108.	166	31	0,07	1,57	442.	109	41	0,73	0,35	776.	93	125	0,69	1,20
109.	33	86	0,65	1,72	443.	173	80	0,15	1,20	777.	93	122	0,36	1,78
110.	32	89	0,18	1,54	444.	29	73	0,38	0,51	778.	187	53	0,87	1,01
111.	153	137	0,94	0,58	445.	145	149	0,78	1,12	779.	140	126	0,07	1,57
112.	66	69	0,33	1,76	446.	34	56	0,45	0,93	780.	177	11	0,77	1,27
113.	57	145	0,75	1,59	447.	89	133	0,28	1,17	781.	41	82	0,39	0,52
114.	147	85	0,26	0,25	448.	53	115	0,13	1,81	782.	69	61	0,32	0,60
115.	192	119	0,65	0,41	449.	138	98	0,58	1,18	783.	124	14	0,05	0,83
116.	154	63	0,72	1,17	450.	41	8	0,36	1,74	784.	147	91	0,40	1,20
117.	93	106	0,88	1,22	451.	164	98	0,68	0,49	785.	157	57	0,06	1,86
118.	141	63	0,93	1,60	452.	156	137	0,25	1,41	786.	82	147	0,75	1,12
119.	161	59	0,49	0,55	453.	41	30	0,44	0,55	787.	76	148	0,64	0,85
120.	184	10	0,69	0,80	454.	112	33	0,30	1,46	788.	35	33	0,32	1,62
121.	196	123	0,67	1,68	455.	73	28	0,70	0,46	789.	75	104	0,10	0,88
122.	146	77	0,30	0,55	456.	63	125	0,15	1,40	790.	54	19	0,93	0,92
123.	90	71	0,40	0,25	457.	176	16	0,44	1,50	791.	113	40	0,32	1,97
124.	31	126	0,44	1,45	458.	25	80	0,75	0,29	792.	165	108	0,20	1,56
125.	130	133	0,94	1,51	459.	88	129	0,79	0,80	793.	128	144	0,90	1,62
126.	148	139	0,05	0,68	460.	183	70	0,67	0,37	794.	129	107	0,48	1,11
127.	126	43	0,94	0,45	461.	116	25	0,30	1,20	795.	132	23	0,76	1,58
128.	192	14	0,08	0,51	462.	157	129	0,39	0,28	796.	113	123	0,55	0,89
129.	114	43	0,35	1,96	463.	130	82	0,21	1,64	797.	49	82	0,67	1,12
130.	48	69	0,50	1,07	464.	174	138	0,48	1,51	798.	75	111	0,35	1,32
131.	179	109	0,64	1,06	465.	193	107	0,32	0,31	799.	37	149	0,20	1,58
132.	87	93	0,17	0,31	466.	88	33	0,84	0,42	800.	63	66	0,49	0,60
133.	53	7	0,33	0,34	467.	190	102	0,57	1,10	801.	163	31	0,38	1,93
134.	132	115	0,79	0,45	468.	197	39	0,13	0,81	802.	142	45	0,75	1,14
135.	149	123	0,72	1,73	469.	70	11	0,89	1,14	803.	183	122	0,44	0,81
136.	132	63	0,29	1,32	470.	174	103	0,93	0,74	804.	138	139	0,14	1,71
137.	184	21	0,30	1,85	471.	99	28	0,45	0,38	805.	45	147	0,70	1,25
138.	179	72	0,27	0,83	472.	173	59	0,39	0,24	806.	66	107	0,19	0,37
139.	88	35	0,18	1,28	473.	183	121	0,80	0,50	807.	136	125	0,37	1,68
140.	76	126	0,25	1,06	474.	59	116	0,15	0,96	808.	92	129	0,16	1,12
141.	82	67	0,61	0,24	475.	159	49	0,56	1,50	809.	159	14	0,46	0,26
142.	65	57	0,33	1,27	476.	105	47	0,65	1,97	810.	130	59	0,84	1,79
143.	195	64	0,90	1,42	477.	182	82	0,16	0,92	811.	184	22	0,22	1,65
144.	67	94	0,37	0,45	478.	34	62	0,18	0,29	812.	107	73	0,28	1,90
145.	38	124	0,78	0,90	479.	74	7	0,78	1,91	813.	99	19	0,18	0,21
146.	109	141	0,32	0,80	480.	36	129	0,10	1,43	814.	166	25	0,70	1,63
147.	146	130	0,83	0,21	481.	52	7	0,75	0,94	815.	85	90	0,39	0,67
148.	36	16	0,11	0,33	482.	104	119	0,94	1,68	816.	132	13	0,79	1,79
149.	90	116	0,28	0,23	483.	78	83	0,69	1,83	817.	175	12	0,31	0,82
150.	33	120	0,59	1,32	484.	177	20	0,90	1,62	818.	75	79	0,19	0,38
151.	119	29	0,31	1,93	485.	150	104	0,58	1,81	819.	68	114	0,62	0,96
152.	136	88	0,54	1,68	486.	37	131	0,74	0,49	820.	188	107	0,55	1,33
153.	117	139	0,64	0,68	487.	95	28	0,90	1,09	821.	99	130	0,07	0,80
154.	99	53	0,14	1,15	488.	142	116	0,78	0,37	822.	184	44	0,15	1,85
155.	137	20	0,13	0,32	489.	149	103	0,87	1,25	823.	42	32	0,67	1,19
156.	45	57	0,73	0,30	490.	94	148	0,34	0,91	824.	47	26	0,83	0,67
157.	61	72	0,95	0,32	491.	175	58	0,56	0,77	825.	108	89	0,81	0,65
158.	105	72	0,63	0,97	492.	191	47	0,92	0,75	826.	136	132	0,71	1,48

n.	N	M	DnE	Exp	n.	N	M	DnE	Exp	n.	N	M	DnE	Exp
159.	104	82	0,44	1,76	493.	127	150	0,73	0,47	827.	115	140	0,15	0,78
160.	92	91	0,87	1,97	494.	79	80	0,60	1,91	828.	158	75	0,94	1,78
161.	99	31	0,06	1,40	495.	195	126	0,72	1,25	829.	78	147	0,47	1,91
162.	197	138	0,17	1,81	496.	44	105	0,42	0,98	830.	39	36	0,44	0,81
163.	114	118	0,17	0,45	497.	189	40	0,09	0,80	831.	188	142	0,28	1,40
164.	92	11	0,47	0,70	498.	113	51	0,90	1,81	832.	115	33	0,14	0,34
165.	169	147	0,39	1,90	499.	165	92	0,79	0,99	833.	169	27	0,86	1,91
166.	26	37	0,95	1,71	500.	124	7	0,22	0,86	834.	105	116	0,36	1,37
167.	78	92	0,23	0,69	501.	85	140	0,86	0,90	835.	62	92	0,48	1,09
168.	145	39	0,29	1,72	502.	109	110	0,57	1,87	836.	112	35	0,86	1,84
169.	97	16	0,35	1,70	503.	45	64	0,11	0,55	837.	91	62	0,84	1,83
170.	103	46	0,89	1,54	504.	105	22	0,47	0,38	838.	73	128	0,76	0,92
171.	58	36	0,94	1,18	505.	49	73	0,85	0,87	839.	63	69	0,10	0,81
172.	30	63	0,90	0,92	506.	175	97	0,78	1,47	840.	178	54	0,29	0,71
173.	51	136	0,90	0,53	507.	169	92	0,38	0,60	841.	46	118	0,25	1,92
174.	77	143	0,05	1,65	508.	167	91	0,49	0,41	842.	41	127	0,36	0,33
175.	129	40	0,26	0,55	509.	179	143	0,79	1,25	843.	92	115	0,68	1,00
176.	177	37	0,26	1,60	510.	141	72	0,30	0,26	844.	196	57	0,30	1,54
177.	34	124	0,45	1,52	511.	52	106	0,21	1,07	845.	98	71	0,07	1,94
178.	196	98	0,90	0,95	512.	27	89	0,07	0,82	846.	33	42	0,74	0,37
179.	122	112	0,50	0,46	513.	146	50	0,18	1,86	847.	170	74	0,23	1,13
180.	94	67	0,24	0,68	514.	60	97	0,85	0,56	848.	65	66	0,42	0,29
181.	33	42	0,46	0,88	515.	127	68	0,12	0,69	849.	134	87	0,73	0,89
182.	165	142	0,40	1,56	516.	134	109	0,91	0,34	850.	77	85	0,12	1,39
183.	26	45	0,19	1,41	517.	87	9	0,85	0,68	851.	131	28	0,07	1,02
184.	82	11	0,10	1,75	518.	132	83	0,06	1,35	852.	142	73	0,53	0,93
185.	118	92	0,81	0,24	519.	122	99	0,47	1,09	853.	86	73	0,21	1,63
186.	183	83	0,45	1,27	520.	177	9	0,73	0,94	854.	62	65	0,29	1,80
187.	36	20	0,80	0,40	521.	58	78	0,14	1,58	855.	50	18	0,62	1,08
188.	122	150	0,76	1,41	522.	130	128	0,79	0,27	856.	48	146	0,47	1,31
189.	146	58	0,48	1,55	523.	131	29	0,45	0,58	857.	183	113	0,55	1,84
190.	115	52	0,64	1,56	524.	98	22	0,13	0,34	858.	105	89	0,56	1,95
191.	59	120	0,16	0,69	525.	162	64	0,31	1,93	859.	110	80	0,91	1,49
192.	90	97	0,94	1,32	526.	109	36	0,26	0,25	860.	191	15	0,35	0,31
193.	144	115	0,86	1,06	527.	124	117	0,81	0,54	861.	68	43	0,77	0,31
194.	193	83	0,89	0,88	528.	186	43	0,33	1,70	862.	82	52	0,71	1,18
195.	66	110	0,57	1,08	529.	97	150	0,11	1,12	863.	193	52	0,85	1,64
196.	40	15	0,22	0,70	530.	27	66	0,71	1,24	864.	64	99	0,09	0,46
197.	131	65	0,90	1,23	531.	27	36	0,88	1,30	865.	120	72	0,92	1,51
198.	30	32	0,72	0,72	532.	68	66	0,36	1,06	866.	170	59	0,24	0,75
199.	83	110	0,68	0,54	533.	69	14	0,81	1,64	867.	112	48	0,46	0,52
200.	120	6	0,54	1,82	534.	123	20	0,24	0,60	868.	87	82	0,59	1,10
201.	181	141	0,56	1,63	535.	133	79	0,30	1,64	869.	113	55	0,76	0,91
202.	125	106	0,67	1,89	536.	131	83	0,22	1,65	870.	162	114	0,66	1,32
203.	105	95	0,42	0,89	537.	163	66	0,59	1,48	871.	179	91	0,66	0,32
204.	161	124	0,92	0,83	538.	62	32	0,34	1,77	872.	178	139	0,30	1,07
205.	170	84	0,56	0,42	539.	105	53	0,15	1,53	873.	121	139	0,91	1,42
206.	20	109	0,86	1,84	540.	147	14	0,73	1,49	874.	137	28	0,26	0,23
207.	174	14	0,38	0,21	541.	176	149	0,11	0,31	875.	22	17	0,68	1,73
208.	72	70	0,63	1,13	542.	162	19	0,35	1,90	876.	32	115	0,46	1,62
209.	58	60	0,63	1,49	543.	24	21	0,50	0,60	877.	191	148	0,18	1,29
210.	102	56	0,21	0,96	544.	47	116	0,81	1,58	878.	63	48	0,08	0,26
211.	165	107	0,21	1,64	545.	153	89	0,84	1,56	879.	151	79	0,54	0,75
212.	140	131	0,84	0,79	546.	106	97	0,61	0,25	880.	159	46	0,40	0,86
213.	46	106	0,21	0,43	547.	52	87	0,25	2,00	881.	159	117	0,06	1,66
214.	137	68	0,25	0,93	548.	117	114	0,60	1,01	882.	108	141	0,91	1,30
215.	116	55	0,47	1,51	549.	174	70	0,06	0,56	883.	192	93	0,24	0,76
216.	61	117	0,15	1,96	550.	36	52	0,52	0,60	884.	115	111	0,44	1,19
217.	31	124	0,08	0,95	551.	70	76	0,82	1,72	885.	159	62	0,21	0,81
218.	119	34	0,06	0,37	552.	45	118	0,26	1,20	886.	22	105	0,39	1,30
219.	35	46	0,28	0,65	553.	147	5	0,45	1,56	887.	58	147	0,67	1,91
220.	53	93	0,56	0,73	554.	26	50	0,48	0,60	888.	189	110	0,31	0,49
221.	143	18	0,81	0,71	555.	61	116	0,45	0,41	889.	191	66	0,89	0,99
222.	106	85	0,83	1,27	556.	109	73	0,66	1,59	890.	55	19	0,71	0,52
223.	34	46	0,06	1,04	557.	125	130	0,78	0,71	891.	61	101	0,59	1,11
224.	171	126	0,91	1,45	558.	144	123	0,41	0,81	892.	190	26	0,24	1,04
225.	66	12	0,71	1,14	559.	83	17	0,43	1,58	893.	106	112	0,47	1,44
226.	178	42	0,74	1,15	560.	80	40	0,45	0,82	894.	178	141	0,38	1,09
227.	97	80	0,09	0,62	561.	146	91	0,33	1,20	895.	116	79	0,53	0,35
228.	118	79	0,43	1,32	562.	91	91	0,69	0,53	896.	130	14	0,11	1,96
229.	174	146	0,49	0,25	563.	112	75	0,87	0,87	897.	166	10	0,18	0,54
230.	151	73	0,59	1,06	564.	115	102	0,25	0,29	898.	148	147	0,11	0,88
231.	181	118	0,46	0,32	565.	33	134	0,07	0,60	899.	167	75	0,52	1,62
232.	161	20	0,34	1,84	566.	80	89	0,71	1,33	900.	114	139	0,92	1,73
233.	105	137	0,60	0,83	567.	65	39	0,43	1,91	901.	89	9	0,49	1,31
234.	184	16	0,82	1,93	568.	193	77	0,60	1,72	902.	88	143	0,83	1,82
235.	72	149	0,86	1,47	569.	98	55	0,18	0,90	903.	26	54	0,87	0,70
236.	58	129	0,16	1,06	570.	87	135	0,94	1,69	904.	85	18	0,19	0,32
237.	195	64	0,90	1,46	571.	102	84	0,36	1,50	905.	112	123	0,86	1,10
238.	188	46	0,69	0,57	572.	131	57	0,57	1,77	906.	141	59	0,43	0,98
239.	28	112	0,57	0,58	573.	84	109	0,12	0,93	907.	168	88	0,44	1,88
240.	48	64	0,34	0,28	574.	181	64	0,75	0,78	908.	199	96	0,32	1,12

n.	N	M	p <sub>NE</sub>	ε <sub>P</sub>	n.	N	M	p <sub>NE</sub>	ε <sub>P</sub>	n.	N	M	p <sub>NE</sub>	ε <sub>P</sub>
241.	115	111	0,39	1,98	575.	81	141	0,08	1,32	909.	147	21	0,23	1,00
242.	90	7	0,28	0,27	576.	134	97	0,95	0,80	910.	148	49	0,68	0,73
243.	172	139	0,07	0,69	577.	134	65	0,87	1,71	911.	106	82	0,31	0,47
244.	109	78	0,59	0,44	578.	165	91	0,42	0,20	912.	159	132	0,14	1,94
245.	148	56	0,36	1,04	579.	95	10	0,67	1,54	913.	80	85	0,21	1,97
246.	167	91	0,43	1,37	580.	27	52	0,72	1,74	914.	79	71	0,57	0,96
247.	157	67	0,69	0,20	581.	91	14	0,56	1,52	915.	172	136	0,62	1,68
248.	106	134	0,83	1,34	582.	174	128	0,14	1,48	916.	77	51	0,18	0,23
249.	191	29	0,09	0,40	583.	49	84	0,37	1,08	917.	121	38	0,94	1,00
250.	166	76	0,25	0,75	584.	24	108	0,68	1,96	918.	161	95	0,15	1,06
251.	89	115	0,74	0,51	585.	165	53	0,31	1,01	919.	149	36	0,58	0,50
252.	94	35	0,79	1,01	586.	100	124	0,40	1,34	920.	24	40	0,60	0,89
253.	108	98	0,84	1,17	587.	156	62	0,23	1,13	921.	114	51	0,15	1,27
254.	189	10	0,86	1,89	588.	134	37	0,58	1,07	922.	100	65	0,28	0,57
255.	177	118	0,69	1,37	589.	192	71	0,47	1,28	923.	185	126	0,36	0,43
256.	148	19	0,16	0,25	590.	200	139	0,86	1,85	924.	92	59	0,62	1,61
257.	103	37	0,18	1,79	591.	97	29	0,67	1,46	925.	63	31	0,29	1,25
258.	198	42	0,59	0,85	592.	45	99	0,20	1,79	926.	60	143	0,08	0,49
259.	111	57	0,62	0,56	593.	177	8	0,14	0,59	927.	92	56	0,33	1,97
260.	45	94	0,34	1,50	594.	188	111	0,26	0,71	928.	44	57	0,17	0,82
261.	41	98	0,20	0,81	595.	187	30	0,50	1,40	929.	108	137	0,72	1,76
262.	163	39	0,07	1,55	596.	137	29	0,57	0,88	930.	162	64	0,16	1,35
263.	117	122	0,10	1,64	597.	174	113	0,25	1,54	931.	180	121	0,19	1,84
264.	108	9	0,69	1,72	598.	61	144	0,84	0,38	932.	144	100	0,60	1,45
265.	47	57	0,86	0,69	599.	66	102	0,65	1,69	933.	77	109	0,17	1,48
266.	120	14	0,28	0,53	600.	151	101	0,82	1,48	934.	183	149	0,88	0,21
267.	27	9	0,95	0,24	601.	25	142	0,65	1,36	935.	96	82	0,51	0,33
268.	172	26	0,16	1,21	602.	132	142	0,45	1,26	936.	95	130	0,43	0,73
269.	185	56	0,14	0,33	603.	169	100	0,89	1,75	937.	144	85	0,93	0,86
270.	170	46	0,12	1,04	604.	169	129	0,48	1,76	938.	119	121	0,87	1,03
271.	21	43	0,66	1,83	605.	42	56	0,29	1,35	939.	62	18	0,88	0,90
272.	141	137	0,91	0,33	606.	169	117	0,71	1,37	940.	49	95	0,59	1,77
273.	197	77	0,71	1,72	607.	82	70	0,39	0,23	941.	166	49	0,79	1,19
274.	145	86	0,24	1,17	608.	146	19	0,14	0,94	942.	33	112	0,26	0,30
275.	67	117	0,36	1,96	609.	67	88	0,95	1,75	943.	137	104	0,25	0,87
276.	164	27	0,41	1,37	610.	193	132	0,34	1,05	944.	94	34	0,37	1,09
277.	96	76	0,75	0,85	611.	141	141	0,43	0,30	945.	147	110	0,19	1,69
278.	74	114	0,09	1,06	612.	132	39	0,37	1,47	946.	131	18	0,18	0,56
279.	62	138	0,25	0,40	613.	125	8	0,84	1,98	947.	187	19	0,81	1,22
280.	186	112	0,87	1,69	614.	177	101	0,24	1,79	948.	46	62	0,77	1,81
281.	110	144	0,22	1,20	615.	120	97	0,95	1,91	949.	196	68	0,29	0,40
282.	33	44	0,25	1,30	616.	136	30	0,59	0,70	950.	146	95	0,85	1,57
283.	20	139	0,92	0,73	617.	81	44	0,35	0,30	951.	154	144	0,85	0,52
284.	162	90	0,55	0,76	618.	124	77	0,21	1,31	952.	103	41	0,82	1,61
285.	167	115	0,65	1,21	619.	135	116	0,13	0,83	953.	47	30	0,73	1,51
286.	153	39	0,80	1,79	620.	22	33	0,66	1,75	954.	38	42	0,85	1,89
287.	197	129	0,08	0,62	621.	110	47	0,16	1,44	955.	58	33	0,89	1,78
288.	173	46	0,13	1,95	622.	171	51	0,87	1,81	956.	200	128	0,42	1,64
289.	37	121	0,18	1,78	623.	118	30	0,45	0,32	957.	197	27	0,93	0,49
290.	143	54	0,65	1,92	624.	121	117	0,11	1,42	958.	95	25	0,94	0,51
291.	45	105	0,10	0,33	625.	156	122	0,53	0,30	959.	56	85	0,24	0,60
292.	155	65	0,35	1,72	626.	98	100	0,19	0,49	960.	117	17	0,37	0,23
293.	97	72	0,84	1,82	627.	166	98	0,61	0,33	961.	40	68	0,14	1,78
294.	149	126	0,50	1,22	628.	160	44	0,12	0,44	962.	102	133	0,17	1,86
295.	143	122	0,24	1,97	629.	50	124	0,12	0,55	963.	30	77	0,64	0,92
296.	89	9	0,52	0,98	630.	59	117	0,60	1,42	964.	59	45	0,71	0,53
297.	177	142	0,94	0,38	631.	130	147	0,57	0,85	965.	32	131	0,68	1,19
298.	190	81	0,70	0,45	632.	114	113	0,91	1,19	966.	140	84	0,32	1,94
299.	60	21	0,91	0,94	633.	69	137	0,92	1,86	967.	111	44	0,10	0,92
300.	74	15	0,52	1,47	634.	93	43	0,68	0,91	968.	103	58	0,82	0,50
301.	147	56	0,42	1,79	635.	172	62	0,84	1,49	969.	58	69	0,59	1,21
302.	105	113	0,44	1,42	636.	55	36	0,27	1,02	970.	38	82	0,26	0,85
303.	154	47	0,43	1,78	637.	59	29	0,86	1,42	971.	85	122	0,23	0,57
304.	151	105	0,92	1,84	638.	41	85	0,53	0,61	972.	150	53	0,50	0,27
305.	30	133	0,44	0,53	639.	35	147	0,54	1,37	973.	100	114	0,55	1,68
306.	140	76	0,64	1,32	640.	103	27	0,05	0,74	974.	92	57	0,52	1,81
307.	37	46	0,42	0,83	641.	120	94	0,21	1,61	975.	36	37	0,65	0,46
308.	34	75	0,10	1,88	642.	119	101	0,08	0,47	976.	170	91	0,93	1,52
309.	33	146	0,16	1,48	643.	97	38	0,07	0,94	977.	180	65	0,94	1,61
310.	151	51	0,78	0,80	644.	181	101	0,25	0,78	978.	79	100	0,63	0,73
311.	48	76	0,77	0,44	645.	128	113	0,25	1,29	979.	42	136	0,22	0,57
312.	171	101	0,85	0,72	646.	113	54	0,82	0,34	980.	151	83	0,72	0,60
313.	72	13	0,48	0,81	647.	101	134	0,80	1,71	981.	149	69	0,88	0,81
314.	164	10	0,42	0,58	648.	176	96	0,23	1,10	982.	169	8	0,19	1,62
315.	79	32	0,36	1,25	649.	140	19	0,70	0,54	983.	95	14	0,68	0,62
316.	94	99	0,53	1,47	650.	90	47	0,42	1,60	984.	43	38	0,76	1,57
317.	45	26	0,47	1,86	651.	181	12	0,34	1,03	985.	23	46	0,14	1,76
318.	46	41	0,63	1,68	652.	55	128	0,25	1,31	986.	55	131	0,61	1,16
319.	133	110	0,57	0,60	653.	170	145	0,87	0,33	987.	167	113	0,15	0,65
320.	182	9	0,25	1,18	654.	161	117	0,72	0,43	988.	147	53	0,18	0,26
321.	38	106	0,49	1,17	655.	150	7	0,40	1,07	989.	119	32	0,10	1,58
322.	81	120	0,29	1,20	656.	124	60	0,09	0,93	990.	173	127	0,27	0,99

n.	N	M	p <sub>nE</sub>	e <sub>xP</sub>	n.	N	M	p <sub>nE</sub>	e <sub>xP</sub>	n.	N	M	p <sub>nE</sub>	e <sub>xP</sub>
323.	70	26	0,61	0,44	657.	49	100	0,32	0,70	991.	21	62	0,61	1,22
324.	56	36	0,77	1,80	658.	179	38	0,71	1,26	992.	176	71	0,67	1,14
325.	163	94	0,56	1,24	659.	50	109	0,82	0,60	993.	165	24	0,06	0,61
326.	92	63	0,78	0,22	660.	49	35	0,68	0,50	994.	136	147	0,34	0,43
327.	104	56	0,78	1,63	661.	30	13	0,69	1,63	995.	160	141	0,85	1,22
328.	179	53	0,73	1,76	662.	151	144	0,07	1,51	996.	189	119	0,36	1,77
329.	128	32	0,54	1,67	663.	28	10	0,65	1,48	997.	188	16	0,42	0,95
330.	151	67	0,70	0,48	664.	132	139	0,91	0,82	998.	194	104	0,38	0,32
331.	59	76	0,80	1,09	665.	98	35	0,19	0,53	999.	87	27	0,06	0,60
332.	99	46	0,75	0,79	666.	49	97	0,78	0,60	1000.	84	87	0,20	0,60
333.	106	89	0,39	1,39	667.	134	136	0,88	1,00	1001.	---	---	---	---
334.	174	32	0,46	1,22	668.	27	110	0,84	0,30	1002.	---	---	---	---

## Appendix C: Variability of Sample Instances

This appendix presents the statistical parameters of the frequency distributions generated by single and repeated (Monte Carlo) simulations per sample instance of the CVP vs CIBI general design problem. The parameters of each sample instance have been pooled by the randomly generated sample in Appendix B. The conducted simulations are referred to 7<sup>th</sup> fully stochastic simulation state in Table 6-4. The focus is on the variability of the simulations per sample insurance as expressed by the dimensionless parameter of coefficient of variation (CV). In general, a normal distribution has skewness 0 and kurtosis 3. Negative skewness represents left skewed distributions and positive skewness represents right skewed distributions. An overall (graphical) assessment of the statistical parameters of the  $\Sigma_{\lambda}Y_i^{\text{repeated}}$  and  $Y_i$  variables concerning all the sample instances ( $i:[1, \dots, 1000]$ ) is presented in subsection 6.4.9.4.

### Frequency distributions of the total effort assessments ( $\Sigma_{\lambda}Y_i^{\text{repeated}}$ ) per sample instance

In the flowing table, each row of the table represents the corresponding sample instance in Appendix B. In each row of the table (for each sample instance  $i$ ), several statistical parameters of the frequency distribution of the total effort assessments  $\Sigma_{\lambda}Y_i$  (per design alternative) for several repeated (Monte Carlo) simulations are presented. Parameter  $\mu$  represents the mean value,  $\sigma$  is the standard deviation,  $CV=\sigma/\mu$  is the coefficient of variation, ‘skew’ is the skewness, and ‘kurt’ is the kurtosis of each frequency distribution. Since the total effort outcome ( $\Sigma_{\lambda}Y_i$ ) is the sum of several ( $\lambda=200$ ) intermediate effort assessments ( $Y_i$ ) per applied scenario, its frequency distribution follows (or resembles) a normal distribution pattern for all the sample instances. Thus, its skewness for all sample instances lies around 0 [-1.09, ..., 1.26] and the kurtosis around 3 [1.90, ..., 6.06], as discussed in subsection 6.4.9.3. However, the CV parameter considerably variates per sample instance between 0.03 and 0.51, while  $\approx 60\%$  of instances have a CV less than 0.10,  $\approx 90\%$  of instances have a CV less than 0.20, and  $\approx 3\%$  of instances have a CV more than 0.30.

Statistical Parameters of Total Effort of Repeated (Monte Carlo) Simulations per Sample Instance

inst. n.	Distribution of CVP total effort ( $\lambda=200$ )					Distribution of CIBI total effort ( $\lambda=200$ )				
	$\mu$	$\sigma$	CV	skew	kurt	$\mu$	$\sigma$	CV	skew	kurt
1.	32361	2275	0,07	-0,07	2,97	34659	6280	0,18	-0,06	2,66
2.	25635	3623	0,14	-0,46	3,01	27037	1655	0,06	-0,29	2,67
3.	42955	3308	0,08	-0,05	3,00	55596	4024	0,07	-0,41	2,89
4.	17914	4713	0,26	0,39	2,27	25048	2079	0,08	0,23	2,58
5.	37234	1422	0,04	0,92	3,87	22943	3424	0,15	0,36	2,27
6.	9004	2695	0,30	0,33	2,40	9193	4433	0,48	0,49	2,61
7.	9404	1756	0,19	0,04	2,51	14275	1254	0,09	0,19	2,87
8.	32613	3953	0,12	0,07	2,54	51115	2397	0,05	0,43	3,37
9.	21800	1549	0,07	0,46	2,74	40026	1666	0,04	-0,26	3,32
10.	12591	2793	0,22	-0,31	2,63	16638	4394	0,26	-0,09	2,72
11.	27277	1371	0,05	-0,31	3,27	24459	3561	0,15	0,03	2,64
12.	23992	3958	0,16	-0,16	2,65	20953	1966	0,09	-0,28	2,80
13.	29405	3849	0,13	0,69	2,60	50824	1785	0,04	0,01	2,74
14.	41375	3528	0,09	0,04	2,07	70142	3383	0,05	-0,38	3,13
15.	20200	2225	0,11	0,04	2,35	31465	1460	0,05	0,73	3,44
16.	35972	2440	0,07	0,41	2,77	55764	3151	0,06	-0,64	3,81
17.	9453	1426	0,15	-0,10	2,33	12153	1230	0,10	0,11	3,35
18.	51209	1639	0,03	0,35	2,67	29554	3663	0,12	0,62	3,07
19.	33606	4164	0,12	-0,41	3,06	37773	1626	0,04	-0,09	2,50
20.	52616	2543	0,05	-0,13	3,78	43784	4167	0,10	-0,07	3,00
21.	34982	2483	0,07	-0,53	3,01	29929	1974	0,07	0,00	3,46
22.	10933	3793	0,35	0,40	2,41	12354	6255	0,51	0,35	2,32
23.	29734	2161	0,07	0,16	2,64	53387	2583	0,05	-0,36	3,58
24.	32478	2204	0,07	0,38	2,74	24561	1728	0,07	-0,19	2,64
25.	33986	1712	0,05	-0,04	2,45	51640	3939	0,08	-0,18	2,86
26.	35445	2093	0,06	-0,06	2,76	31470	5727	0,18	-0,15	2,63
27.	27338	3245	0,12	-0,24	2,30	41147	1745	0,04	0,27	2,73
28.	24733	1556	0,06	-0,38	2,93	39601	2501	0,06	-0,66	3,32
29.	12718	2838	0,22	0,54	2,35	10632	5040	0,47	0,70	2,59



inst. n.	Distribution of CVP total effort ( $\lambda=200$ )					Distribution of CIBI total effort ( $\lambda=200$ )				
	$\mu$	$\sigma$	CV	skew	kurt	$\mu$	$\sigma$	CV	skew	kurt
30.	21953	1598	0,07	-0,05	2,43	38655	1891	0,05	-0,46	3,03
31.	15288	1338	0,09	-0,14	2,63	10545	2155	0,20	-0,01	2,13
32.	41150	2301	0,06	0,18	2,70	48720	3743	0,08	-0,27	3,41
33.	27034	1431	0,05	0,10	2,39	16393	1713	0,10	0,01	2,99
34.	31353	1592	0,05	-0,37	4,00	38949	3364	0,09	-0,12	2,57
35.	27633	2030	0,07	-0,18	3,09	47384	5386	0,11	-0,06	2,14
36.	46412	2127	0,05	-0,14	3,03	54402	5187	0,10	0,13	2,53
37.	22366	3526	0,16	0,13	2,64	21021	7112	0,34	0,28	2,62
38.	43294	2163	0,05	-0,12	2,95	52005	4353	0,08	-0,66	4,28
39.	36763	4633	0,13	-0,22	3,38	33430	1865	0,06	0,04	2,90
40.	48923	2648	0,05	0,09	2,19	43135	4315	0,10	-0,04	2,51
41.	18439	4099	0,22	0,08	2,62	22901	1931	0,08	0,26	2,53
42.	38863	3352	0,09	0,10	2,37	46329	2651	0,06	-0,08	2,51
43.	49004	1988	0,04	0,26	4,13	48099	5874	0,12	0,12	2,98
44.	18491	1961	0,11	-0,21	3,00	18924	3900	0,21	-0,11	2,86
45.	33370	1622	0,05	0,44	2,94	61000	3507	0,06	-0,38	3,14
46.	51410	2089	0,04	0,33	3,28	30720	3252	0,11	0,56	3,12
47.	43046	1662	0,04	0,38	3,29	27088	4506	0,17	0,51	3,14
48.	41513	4074	0,10	-0,03	2,78	28116	1461	0,05	0,03	2,86
49.	34209	1959	0,06	0,14	2,98	27476	2310	0,08	-0,22	2,69
50.	21054	3254	0,15	-0,24	2,72	22556	1553	0,07	-0,59	3,17
51.	30920	5155	0,17	-0,14	2,57	26262	2211	0,08	-0,49	2,97
52.	45649	2652	0,06	-0,19	2,97	49084	4029	0,08	0,14	2,42
53.	54263	1949	0,04	-0,19	2,99	31418	4120	0,13	0,55	2,26
54.	27243	6452	0,24	0,21	2,42	22711	3109	0,14	-0,04	2,54
55.	11152	5094	0,46	0,92	4,58	12269	2923	0,24	0,85	4,37
56.	49186	2342	0,05	0,05	3,13	45330	4372	0,10	0,07	2,77
57.	37124	4304	0,12	0,15	2,35	31934	1615	0,05	-0,13	2,94
58.	46847	3761	0,08	0,02	2,56	33666	1575	0,05	0,28	2,75
59.	34474	1858	0,05	0,03	2,96	63657	5232	0,08	-0,19	2,27
60.	50735	3276	0,06	-0,17	2,84	31495	2093	0,07	0,18	2,89
61.	33502	1335	0,04	-0,19	2,75	64136	3859	0,06	-0,19	2,57
62.	27902	4005	0,14	-0,42	3,22	29819	1461	0,05	0,06	3,75
63.	35418	4857	0,14	-0,01	3,10	35738	1514	0,04	0,33	2,63
64.	45714	2366	0,05	-0,32	2,90	32589	2635	0,08	-0,28	3,11
65.	29790	4765	0,16	0,20	2,78	26557	1885	0,07	-0,17	2,34
66.	42815	2618	0,06	-0,11	2,75	43437	3223	0,07	-0,06	2,75
67.	20495	1377	0,07	-0,09	3,10	23288	1816	0,08	-0,60	3,87
68.	46455	2514	0,05	0,10	2,32	29568	2156	0,07	-0,15	3,13
69.	39887	3373	0,08	-0,11	4,08	39734	2434	0,06	-0,06	2,81
70.	32338	1668	0,05	0,22	2,69	22953	3301	0,14	-0,25	3,09
71.	31367	2031	0,06	0,10	2,88	28126	5091	0,18	-0,30	2,92
72.	40867	2175	0,05	-0,23	2,76	35490	6581	0,19	-0,06	2,53
73.	13244	3724	0,28	0,67	2,69	12740	6442	0,51	0,80	3,00
74.	15260	2940	0,19	0,02	2,68	19553	5529	0,28	0,28	2,82
75.	27549	2746	0,10	0,01	2,32	35037	6444	0,18	0,07	2,66
76.	37476	2226	0,06	0,18	2,10	23063	5270	0,23	0,63	2,31
77.	15595	4839	0,31	0,59	2,97	21056	2258	0,11	0,41	3,21
78.	41184	3733	0,09	0,43	3,53	53489	2420	0,05	0,09	2,48
79.	50475	2422	0,05	0,07	3,06	33119	3474	0,10	-0,01	2,65
80.	37363	2462	0,07	0,04	3,63	29708	2151	0,07	-0,05	2,86
81.	15053	2166	0,14	0,44	2,36	25374	1321	0,05	0,43	3,24
82.	43752	3205	0,07	-0,39	3,41	38030	2393	0,06	0,24	2,34
83.	31551	2603	0,08	-0,13	3,03	43987	6739	0,15	-0,22	2,99
84.	11813	2168	0,18	0,15	2,33	17876	1302	0,07	0,28	2,78
85.	40807	2256	0,06	0,20	2,86	27135	2196	0,08	-0,50	3,27
86.	27619	1556	0,06	0,02	2,77	24641	4243	0,17	0,22	2,97
87.	22622	1391	0,06	-0,37	3,25	17381	1480	0,09	-0,15	3,19
88.	37074	2374	0,06	0,18	2,20	67728	3195	0,05	-0,18	3,06
89.	29476	1343	0,05	0,43	2,90	56240	3442	0,06	0,07	2,82
90.	35135	1855	0,05	0,01	2,71	55513	4263	0,08	-0,09	2,80
91.	34540	3824	0,11	0,33	2,31	59684	2249	0,04	0,16	2,90
92.	35672	2655	0,07	0,67	6,39	37734	2503	0,07	-0,13	2,92
93.	31227	1544	0,05	-0,32	2,50	51782	4373	0,08	-0,16	3,13
94.	18885	2394	0,13	0,46	2,69	32812	1477	0,05	0,75	3,78
95.	23736	1449	0,06	0,31	3,11	27774	2933	0,11	-0,48	3,66
96.	27770	2163	0,08	-0,66	4,24	30743	5277	0,17	-0,52	3,75
97.	41751	1883	0,05	0,14	2,72	31106	4068	0,13	-0,22	2,69
98.	36028	2034	0,06	0,21	2,61	53293	6028	0,11	-0,06	2,60
99.	33925	2385	0,07	0,01	3,25	39204	6030	0,15	0,10	2,86
100.	43452	3650	0,08	-0,29	2,62	43135	2630	0,06	0,16	2,73
101.	35769	2506	0,07	0,07	2,59	33612	6886	0,20	0,17	2,55
102.	42749	3596	0,08	-0,10	2,66	51484	2557	0,05	-0,29	3,26
103.	15930	1488	0,09	0,30	2,98	12288	2174	0,18	0,00	3,07
104.	34590	1757	0,05	0,30	2,88	26978	3049	0,11	-0,05	2,77
105.	24756	2687	0,11	-1,04	5,80	38000	5736	0,15	-0,70	3,93
106.	37715	3249	0,09	0,10	2,34	63179	3099	0,05	0,01	2,99
107.	13648	3529	0,26	0,28	2,28	19439	1916	0,10	0,32	2,30
108.	32136	1343	0,04	0,11	3,59	61578	3179	0,05	-0,32	3,31
109.	31157	2141	0,07	0,12	3,32	22233	1327	0,06	-0,53	3,80
110.	16040	4119	0,26	0,19	2,19	17918	2168	0,12	-0,15	2,79

inst. n.	Distribution of CVP total effort ( $\lambda=200$ )					Distribution of CIBI total effort ( $\lambda=200$ )				
	$\mu$	$\sigma$	CV	skew	kurt	$\mu$	$\sigma$	CV	skew	kurt
111.	53126	1771	0,03	0,36	3,64	30429	3950	0,13	0,79	3,53
112.	23904	2912	0,12	-0,07	3,13	28287	1569	0,06	-0,02	3,00
113.	50782	3345	0,07	-0,36	3,10	31891	1631	0,05	0,34	3,23
114.	34631	2231	0,06	-0,26	3,05	52112	2896	0,06	-0,29	3,32
115.	49593	1959	0,04	0,16	2,59	48284	5806	0,12	0,07	3,22
116.	32191	2557	0,08	-0,18	2,66	31941	6446	0,20	-0,17	2,81
117.	41067	1782	0,04	0,47	3,09	25387	3482	0,14	0,42	2,74
118.	27265	2291	0,08	0,53	2,58	18542	5480	0,30	0,72	2,63
119.	34437	1841	0,05	0,03	2,92	44430	5124	0,12	-0,15	3,02
120.	20315	4290	0,21	-0,31	2,84	30398	8134	0,27	-0,07	2,96
121.	50915	1879	0,04	0,21	3,30	47962	6017	0,13	0,01	2,35
122.	34824	1879	0,05	-0,10	2,88	49960	3515	0,07	0,34	2,65
123.	28647	2146	0,07	0,09	2,77	33759	1895	0,06	0,08	3,11
124.	32857	4698	0,14	-0,48	3,47	25013	1938	0,08	-0,61	4,18
125.	51778	2030	0,04	0,19	2,78	29799	3653	0,12	0,46	2,78
126.	32220	3447	0,11	0,65	2,80	57029	2020	0,04	0,08	2,39
127.	19516	2076	0,11	0,47	2,44	13667	4328	0,32	0,86	3,07
128.	36126	1534	0,04	-0,19	2,69	69668	4640	0,07	-0,22	2,14
129.	26850	1334	0,05	0,02	3,08	38443	2901	0,08	-0,32	2,62
130.	26091	2225	0,09	0,11	2,87	23829	1522	0,06	-0,19	2,94
131.	46534	2015	0,04	0,15	2,38	45589	6012	0,13	-0,35	2,97
132.	23949	3701	0,15	0,32	2,73	35098	1558	0,04	0,18	2,56
133.	14341	1127	0,08	-0,19	2,92	20840	1619	0,08	-0,48	2,72
134.	45343	1684	0,04	0,04	2,73	32873	4976	0,15	-0,02	2,77
135.	48574	2040	0,04	-0,07	2,91	39189	4656	0,12	0,14	2,44
136.	31188	1904	0,06	-0,32	3,08	45765	3115	0,07	-0,12	3,24
137.	33499	2292	0,07	-0,41	3,36	58455	5928	0,10	-0,39	3,32
138.	38320	2104	0,05	0,29	3,26	59987	5163	0,09	-0,47	2,59
139.	21014	1822	0,09	-0,27	2,65	33532	1495	0,04	-0,25	2,96
140.	28708	5278	0,18	-0,18	2,77	33645	1977	0,06	0,04	2,92
141.	29392	1561	0,05	0,44	3,50	28040	2863	0,10	-0,30	2,72
142.	22450	2259	0,10	0,00	3,17	27348	1454	0,05	-0,04	2,95
143.	29545	2806	0,09	0,59	2,89	23065	6538	0,28	0,71	3,37
144.	28083	3658	0,13	-0,18	2,87	30103	1443	0,05	0,40	3,74
145.	44089	2953	0,07	-0,19	2,57	26921	1664	0,06	0,23	2,28
146.	38518	4373	0,11	0,07	2,77	43963	1963	0,04	0,19	2,82
147.	51180	2066	0,04	-0,02	2,32	34821	4532	0,13	0,17	2,07
148.	10225	1924	0,19	0,34	2,39	15937	1308	0,08	0,32	2,57
149.	30808	4014	0,13	-0,43	3,37	37462	1711	0,05	-0,10	2,45
150.	37505	3580	0,10	-0,09	2,77	26105	1431	0,05	-0,10	4,71
151.	25702	1433	0,06	0,35	2,76	39839	3245	0,08	-0,06	3,20
152.	37759	1659	0,04	0,03	2,94	40890	3898	0,10	-0,32	2,95
153.	49704	2758	0,06	-0,25	2,99	40178	3265	0,08	-0,19	2,61
154.	22699	2081	0,09	0,31	2,44	38012	1680	0,04	0,13	3,31
155.	27231	1182	0,04	-0,08	2,99	50379	2736	0,05	0,00	3,21
156.	24538	1454	0,06	-0,04	2,72	18903	2321	0,12	-0,21	2,49
157.	28814	1181	0,04	0,17	2,38	16390	2335	0,14	0,83	2,94
158.	32141	1406	0,04	0,51	3,95	31179	3745	0,12	0,02	2,88
159.	32256	2300	0,07	-0,65	4,04	36677	2530	0,07	-0,08	2,91
160.	36196	1680	0,05	0,50	3,54	23165	3604	0,16	0,51	2,68
161.	20235	1406	0,07	0,67	3,21	38146	1605	0,04	-0,10	3,46
162.	44185	3849	0,09	0,04	2,40	72040	3448	0,05	-0,32	2,95
163.	30046	4280	0,14	0,21	2,45	44795	1781	0,04	0,04	2,63
164.	19179	1553	0,08	-0,51	3,60	28096	3370	0,12	-0,09	3,37
165.	49555	3790	0,08	-0,09	2,59	58174	4184	0,07	-0,16	2,25
166.	15766	1112	0,07	0,43	2,44	9431	1783	0,19	0,46	1,99
167.	25475	3447	0,14	0,09	2,91	33043	1654	0,05	0,35	3,19
168.	30457	1348	0,04	0,08	3,00	48354	4126	0,09	-0,38	3,29
169.	21455	1376	0,06	0,55	5,59	32997	3017	0,09	-0,26	3,23
170.	21404	2232	0,10	0,24	2,92	16264	4675	0,29	0,45	2,79
171.	15957	1823	0,11	0,67	2,89	10451	2983	0,29	0,74	2,66
172.	25277	1251	0,05	0,30	2,84	15127	2075	0,14	0,23	2,16
173.	50934	2230	0,04	-0,20	2,81	28238	1813	0,06	-0,16	2,27
174.	19311	4229	0,22	0,91	3,51	31368	1981	0,06	0,70	3,54
175.	28141	1403	0,05	0,48	3,13	45182	3093	0,07	0,04	2,84
176.	34871	1525	0,04	0,35	2,85	58583	4050	0,07	-0,32	2,85
177.	33356	3844	0,12	-0,41	2,84	25611	1709	0,07	-0,43	3,94
178.	40746	2288	0,06	0,15	2,12	28634	6641	0,23	0,22	2,01
179.	41377	2503	0,06	-0,31	2,18	41948	2801	0,07	0,12	2,44
180.	25417	2789	0,11	-0,06	2,38	36227	1599	0,04	-0,15	2,78
181.	18328	1824	0,10	-0,25	2,90	18258	1184	0,06	-0,12	2,76
182.	47978	3105	0,06	-0,27	3,72	56555	4207	0,07	0,10	2,15
183.	12414	3002	0,24	-0,18	2,75	14664	1775	0,12	-0,40	2,66
184.	17399	1280	0,07	0,62	4,49	31545	1365	0,04	-0,22	2,42
185.	38127	1795	0,05	-0,24	2,39	27953	3801	0,14	0,04	2,73
186.	41291	1985	0,05	0,19	2,85	53689	5444	0,10	-0,37	3,20
187.	12978	1679	0,13	-0,35	2,84	11439	2744	0,24	-0,22	2,42
188.	54952	3150	0,06	0,05	3,11	38233	3429	0,09	0,10	3,14
189.	33210	1247	0,04	-0,13	2,99	42762	3803	0,09	-0,26	2,95
190.	27929	1869	0,07	-0,37	3,53	29488	4677	0,16	-0,28	3,95
191.	22021	5270	0,24	-0,01	2,51	27164	2192	0,08	0,01	2,18

inst. n.	Distribution of CVP total effort ( $\lambda=200$ )					Distribution of CIBI total effort ( $\lambda=200$ )				
	$\mu$	$\sigma$	CV	skew	kurt	$\mu$	$\sigma$	CV	skew	kurt
192.	38027	1464	0,04	-0,03	2,67	21629	2904	0,13	0,64	2,70
193.	45504	1719	0,04	-0,18	2,25	29755	4705	0,16	0,28	2,43
194.	35747	2478	0,07	0,27	2,34	25907	6675	0,26	0,53	2,82
195.	37646	3203	0,09	-0,24	2,92	30354	1782	0,06	0,11	2,54
196.	12380	1769	0,14	-0,09	2,91	17953	1179	0,07	0,33	3,71
197.	28002	2057	0,07	0,29	2,48	19513	4629	0,24	0,36	2,56
198.	16934	1198	0,07	0,03	4,04	14437	1858	0,13	-0,56	4,30
199.	40609	2245	0,06	0,12	3,53	31257	2713	0,09	-0,42	2,90
200.	18837	2637	0,14	-0,55	3,11	28688	4909	0,17	-0,29	2,55
201.	53020	2789	0,05	-0,09	3,64	53086	4631	0,09	-0,05	2,93
202.	42274	2001	0,05	0,31	2,68	36969	4005	0,11	0,02	2,26
203.	34279	2817	0,08	-0,43	3,28	38678	2174	0,06	-0,34	2,71
204.	48784	1890	0,04	0,29	2,54	29656	4234	0,14	0,28	2,41
205.	40068	1827	0,05	-0,03	2,34	45826	5408	0,12	-0,31	3,55
206.	40146	2162	0,05	0,11	2,40	22555	1195	0,05	0,36	3,49
207.	29838	2230	0,07	-0,65	3,97	50063	5546	0,11	-0,47	3,58
208.	29387	1531	0,05	0,10	2,96	26202	2608	0,10	-0,25	2,72
209.	25993	1383	0,05	0,08	3,50	22655	2479	0,11	-0,36	3,37
210.	25106	2302	0,09	-0,25	3,04	38749	1980	0,05	0,07	3,67
211.	38675	3080	0,08	-0,44	2,60	59234	3017	0,05	0,28	2,84
212.	50923	2157	0,04	-0,26	2,91	34516	4821	0,14	0,20	2,71
213.	20576	4861	0,24	-0,06	2,24	23459	2307	0,10	0,00	2,68
214.	31900	2014	0,06	-0,04	3,48	49123	2747	0,06	-0,28	2,76
215.	29251	1439	0,05	0,03	2,23	36305	3353	0,09	0,03	2,61
216.	21042	5076	0,24	0,15	2,01	27514	2049	0,07	0,12	2,27
217.	12734	4825	0,38	0,50	2,18	15776	2557	0,16	0,34	2,04
218.	23876	1347	0,06	0,43	2,22	45187	1825	0,04	-0,30	2,84
219.	16198	2377	0,15	-0,36	2,83	18537	1394	0,08	-0,31	2,94
220.	32144	2399	0,07	-0,20	2,37	26594	1698	0,06	-0,06	3,09
221.	15804	3684	0,23	0,01	2,48	18549	6832	0,37	0,20	2,51
222.	35161	1619	0,05	0,32	3,38	25168	3848	0,15	-0,17	2,65
223.	9596	2589	0,27	0,87	3,11	14984	1516	0,10	0,65	3,01
224.	50034	1940	0,04	0,30	2,85	31205	5430	0,17	0,79	3,91
225.	13454	2251	0,17	-0,39	3,00	15510	3460	0,22	-0,32	2,98
226.	26754	2757	0,10	-0,12	2,38	29893	5915	0,20	-0,15	2,93
227.	22325	3080	0,14	0,70	2,50	38087	1575	0,04	-0,03	2,97
228.	33612	2298	0,07	0,38	3,08	40293	2915	0,07	0,12	2,61
229.	52631	3121	0,06	-0,34	2,84	55696	4762	0,09	-0,25	2,56
230.	36116	1590	0,04	-0,32	2,82	39744	4588	0,12	-0,12	3,45
231.	47100	2391	0,05	-0,15	3,02	56172	4738	0,08	-0,25	3,59
232.	29916	1701	0,06	-0,44	2,69	49807	4459	0,09	-0,27	2,59
233.	47348	3406	0,07	-0,24	2,92	39381	2840	0,07	-0,06	2,59
234.	15787	3884	0,25	-0,21	2,67	19672	7178	0,36	-0,04	2,85
235.	54687	2854	0,05	-0,30	3,10	31803	2126	0,07	-0,14	2,53
236.	22274	4735	0,21	0,04	2,50	27231	2020	0,07	-0,02	2,66
237.	29088	3317	0,11	0,39	2,30	22211	7272	0,33	0,43	2,22
238.	30202	2723	0,09	0,05	2,34	35611	6781	0,19	0,27	2,69
239.	35073	3217	0,09	-0,09	2,85	24633	1162	0,05	-0,25	3,14
240.	21819	2681	0,12	-0,05	2,78	23474	1213	0,05	0,15	2,79
241.	37920	3348	0,09	-0,43	3,23	42797	2595	0,06	-0,25	3,23
242.	19391	998	0,05	-0,14	2,42	31854	2482	0,08	-0,44	3,22
243.	37038	3571	0,10	1,26	5,66	65584	2572	0,04	-0,06	2,75
244.	33656	1613	0,05	-0,13	3,15	34197	3739	0,11	0,01	2,65
245.	32870	1454	0,04	0,18	3,27	47812	4093	0,09	-0,25	3,44
246.	41239	1862	0,05	0,20	2,71	52257	4484	0,09	-0,71	3,69
247.	33557	2302	0,07	-0,47	2,77	33925	5751	0,17	-0,50	2,97
248.	50693	2505	0,05	0,21	3,13	32538	3191	0,10	0,20	2,69
249.	36327	1365	0,04	0,19	2,56	69827	3996	0,06	-0,59	3,27
250.	36717	1947	0,05	0,03	2,68	57323	3664	0,06	-0,17	2,81
251.	43159	2544	0,06	0,03	3,40	30903	3216	0,10	-0,20	2,62
252.	19989	2058	0,10	-0,12	2,98	18956	4241	0,22	0,05	2,75
253.	39428	1613	0,04	0,41	3,15	26553	3193	0,12	0,17	3,22
254.	12535	4427	0,35	-0,03	2,35	16615	7739	0,47	0,01	2,33
255.	48652	1986	0,04	-0,13	3,32	42890	5718	0,13	0,07	2,59
256.	28852	1425	0,05	-0,26	3,13	52484	3815	0,07	-0,19	2,91
257.	23031	1758	0,08	0,24	3,06	38490	2083	0,05	-0,21	2,59
258.	32754	3185	0,10	-0,52	3,74	44246	8146	0,18	-0,28	3,56
259.	29307	1598	0,05	0,39	3,95	30500	4163	0,14	0,11	2,62
260.	25623	3564	0,14	-0,20	2,57	24757	1599	0,06	-0,49	3,78
261.	18673	4291	0,23	0,20	2,75	21049	2070	0,10	0,13	2,42
262.	31876	1509	0,05	0,14	2,90	60112	3330	0,06	0,11	2,39
263.	28031	3687	0,13	0,68	3,76	45755	1769	0,04	0,25	3,28
264.	15209	3016	0,20	-0,40	3,06	20442	5048	0,25	-0,28	2,87
265.	23917	1300	0,05	0,15	3,00	15405	2362	0,15	0,22	2,76
266.	24119	1411	0,06	0,03	4,16	40097	3299	0,08	-0,26	3,02
267.	5239	1530	0,29	0,83	2,90	4020	1989	0,49	0,90	2,80
268.	33173	1386	0,04	0,59	3,40	61364	4372	0,07	-0,61	3,92
269.	36974	1324	0,04	-0,04	2,97	66154	4461	0,07	-0,10	3,03
270.	33899	1428	0,04	-0,14	2,93	62375	3203	0,05	-0,09	2,66
271.	19563	1198	0,06	0,49	3,40	15031	1340	0,09	-0,38	2,94
272.	53392	2012	0,04	-0,03	2,57	31807	4178	0,13	1,03	5,02

inst. n.	Distribution of CVP total effort ( $\lambda=200$ )					Distribution of CIBI total effort ( $\lambda=200$ )				
	$\mu$	$\sigma$	CV	skew	kurt	$\mu$	$\sigma$	CV	skew	kurt
273.	38540	2712	0,07	0,22	3,52	39029	6613	0,17	-0,03	3,23
274.	34543	2982	0,09	0,00	2,42	52718	3124	0,06	-0,47	2,97
275.	31848	4327	0,14	-0,28	2,58	31438	1576	0,05	0,46	3,56
276.	30049	1948	0,06	-0,19	2,45	46667	5050	0,11	-0,11	2,76
277.	32657	1650	0,05	0,41	3,03	26074	3693	0,14	0,00	2,87
278.	20532	4486	0,22	0,28	2,27	30767	1831	0,06	0,54	3,24
279.	28077	5064	0,18	-0,31	2,83	30462	2121	0,07	-0,38	2,61
280.	45853	2040	0,04	0,04	3,15	31231	5309	0,17	0,22	2,46
281.	34567	4907	0,14	0,45	2,96	44880	1876	0,04	-0,14	2,98
282.	15420	2837	0,18	-0,14	2,39	17597	1530	0,09	-0,18	2,71
283.	51866	2615	0,05	-0,35	2,62	27027	1198	0,04	0,69	3,60
284.	40670	1977	0,05	-0,14	2,72	45077	4705	0,10	0,07	2,33
285.	47102	2180	0,05	-0,28	2,98	44529	5900	0,13	-0,14	3,80
286.	22265	2601	0,12	-0,13	2,88	21756	5574	0,26	0,03	2,71
287.	40959	3018	0,07	0,67	3,12	73848	3515	0,05	0,10	2,61
288.	34471	1485	0,04	0,09	2,75	63058	4091	0,06	-0,27	2,37
289.	18221	6073	0,33	0,19	2,68	20071	3050	0,15	0,07	2,79
290.	30000	2050	0,07	-0,95	4,05	33140	5555	0,17	-0,48	3,31
291.	15609	4141	0,27	0,34	2,91	20855	2141	0,10	0,38	3,00
292.	34748	1809	0,05	0,02	2,42	50706	4470	0,09	-0,40	3,00
293.	30382	1672	0,06	-0,10	3,00	21826	3743	0,17	0,05	3,02
294.	46500	2918	0,06	0,30	2,58	48747	4022	0,08	-0,04	2,49
295.	37544	3554	0,09	0,01	2,67	53602	2753	0,05	-0,13	3,02
296.	17976	1488	0,08	0,00	3,23	25533	2898	0,11	0,07	2,15
297.	55686	2084	0,04	0,26	3,24	32751	4954	0,15	0,67	2,86
298.	39255	2440	0,06	-0,16	2,60	39441	6850	0,17	-0,16	2,60
299.	10978	1950	0,18	0,33	2,38	8407	2959	0,35	0,47	2,67
300.	17420	1326	0,08	-0,20	2,82	22950	3028	0,13	-0,15	2,96
301.	32925	1565	0,05	0,05	2,62	44895	4286	0,10	0,02	2,48
302.	38221	2865	0,07	-0,14	2,78	39951	2306	0,06	-0,98	5,79
303.	31813	1679	0,05	-0,07	2,57	45144	4714	0,10	0,21	2,34
304.	41978	1849	0,04	0,17	2,56	25890	4436	0,17	0,62	2,74
305.	34785	4637	0,13	-0,42	2,96	25737	2193	0,09	-0,21	3,06
306.	35705	1853	0,05	0,15	2,84	36058	4725	0,13	-0,30	3,00
307.	19223	1818	0,09	0,02	2,76	19345	1229	0,06	-0,38	3,12
308.	12103	3675	0,30	0,62	3,31	16297	2032	0,12	0,31	2,62
309.	18758	5650	0,30	0,05	2,40	19418	2834	0,15	-0,05	2,10
310.	27234	2928	0,11	0,11	2,73	26568	6454	0,24	0,28	2,91
311.	30445	1661	0,05	-0,42	3,71	20809	2211	0,11	0,02	2,40
312.	42307	2129	0,05	0,42	2,96	30897	6997	0,23	0,64	3,26
313.	17205	1281	0,07	-0,13	2,90	23438	2641	0,11	-0,39	3,08
314.	27664	2352	0,09	-0,31	2,38	46146	5443	0,12	-0,09	2,34
315.	21113	1380	0,07	-0,31	4,26	29091	1925	0,07	0,06	3,62
316.	36454	2721	0,07	-0,06	2,92	34812	2462	0,07	-0,31	2,58
317.	16995	1196	0,07	0,83	4,14	19500	1785	0,09	-0,22	2,56
318.	20661	1238	0,06	0,03	2,98	18701	1888	0,10	-0,66	3,36
319.	42966	2225	0,05	-0,17	2,85	41480	3893	0,09	-0,42	3,27
320.	32609	2064	0,06	-0,40	3,14	59322	5534	0,09	-0,47	3,27
321.	31524	3128	0,10	-0,24	2,67	25085	1365	0,05	0,07	3,28
322.	31336	4394	0,14	-0,19	2,64	35276	1837	0,05	0,32	3,50
323.	18926	1587	0,08	-0,16	2,97	21224	3177	0,15	-0,29	3,56
324.	18665	1621	0,09	-0,26	3,02	15838	3220	0,20	0,17	3,23
325.	41576	1798	0,04	-0,19	2,96	45466	5111	0,11	-0,39	2,63
326.	28233	1718	0,06	0,13	2,70	22191	3621	0,16	-0,25	2,71
327.	26583	1900	0,07	-0,22	2,49	22091	4253	0,19	0,06	2,42
328.	30860	2772	0,09	0,13	2,38	33237	6643	0,20	0,09	2,52
329.	25665	1763	0,07	0,07	2,41	34611	4311	0,12	-0,19	2,32
330.	33051	2057	0,06	0,17	2,82	32805	5355	0,16	0,12	2,38
331.	30718	1340	0,04	-0,09	2,58	21145	2824	0,13	0,18	2,79
332.	23912	1818	0,08	-0,66	3,81	22149	3975	0,18	-0,31	3,19
333.	33256	2387	0,07	-0,43	3,08	39089	2320	0,06	-0,03	2,54
334.	31515	2091	0,07	-0,21	3,01	47766	5652	0,12	-0,33	3,37
335.	50726	2638	0,05	0,23	2,65	55901	4960	0,09	-0,31	2,93
336.	22149	1357	0,06	-0,20	3,64	34931	3318	0,09	-0,21	3,69
337.	34845	1723	0,05	-0,08	3,41	45622	5143	0,11	-0,07	2,61
338.	22242	2489	0,11	-0,10	2,55	29149	5246	0,18	0,13	2,69
339.	47593	2266	0,05	-0,27	3,56	29572	3304	0,11	0,44	3,13
340.	42441	1836	0,04	0,33	2,55	39954	5014	0,13	-0,23	2,52
341.	21155	2462	0,12	-0,38	2,99	30116	4868	0,16	-0,15	2,90
342.	19513	1190	0,06	0,58	3,26	11560	2000	0,17	0,76	3,07
343.	17330	3173	0,18	0,95	3,53	13720	6091	0,44	0,94	3,13
344.	37341	3298	0,09	-0,29	3,07	52405	2758	0,05	-0,48	3,93
345.	29206	2940	0,10	-0,43	3,52	32278	6500	0,20	-0,29	3,58
346.	19833	2984	0,15	0,14	3,02	20859	6215	0,30	0,46	2,85
347.	40520	1742	0,04	-0,03	2,79	62322	4452	0,07	-0,06	2,66
348.	51904	2399	0,05	0,16	3,53	52765	6287	0,12	-0,26	3,25
349.	17993	1227	0,07	0,05	2,38	16750	1438	0,09	-0,08	2,91
350.	24897	4209	0,17	-0,17	3,46	20044	1878	0,09	-1,07	4,46
351.	39420	2285	0,06	0,00	2,68	56626	3757	0,07	0,15	3,27
352.	52214	2932	0,06	0,26	2,96	47964	3198	0,07	-0,29	3,37
353.	13390	3291	0,25	0,24	2,13	18760	1759	0,09	0,44	2,71

inst. n.	Distribution of CVP total effort ( $\lambda=200$ )					Distribution of CIBI total effort ( $\lambda=200$ )				
	$\mu$	$\sigma$	CV	skew	kurt	$\mu$	$\sigma$	CV	skew	kurt
354.	54432	2683	0,05	-0,12	2,42	33586	2872	0,09	-0,10	2,79
355.	26647	2311	0,09	-0,26	3,25	29696	5716	0,19	0,12	3,19
356.	35569	1499	0,04	0,29	2,96	56928	4370	0,08	-0,19	3,35
357.	40034	1837	0,05	0,01	2,59	46901	5847	0,12	-0,61	3,42
358.	19633	2080	0,11	-0,42	3,27	19440	4267	0,22	-0,16	3,08
359.	20696	2220	0,11	0,28	3,30	18313	4307	0,24	0,05	2,68
360.	14962	4220	0,28	0,62	2,92	20946	2227	0,11	0,40	2,23
361.	32979	1613	0,05	0,12	2,86	20252	2730	0,13	-0,06	2,12
362.	18075	3777	0,21	0,14	2,34	23523	1833	0,08	0,22	2,49
363.	30759	3025	0,10	0,35	2,47	54573	2043	0,04	-0,36	2,98
364.	29671	1350	0,05	-0,32	2,68	22185	2832	0,13	0,41	2,92
365.	19997	1503	0,08	-0,49	2,83	29650	1688	0,06	0,05	3,30
366.	33628	4293	0,13	0,09	2,37	52987	2062	0,04	-0,32	2,40
367.	57364	2565	0,04	0,20	2,60	50978	5339	0,10	-0,30	3,02
368.	34732	2352	0,07	0,10	2,88	32979	2531	0,08	-0,44	3,86
369.	48939	2542	0,05	-0,02	2,83	35649	3600	0,10	0,04	2,88
370.	54076	2758	0,05	-0,14	3,03	44148	4036	0,09	0,08	3,63
371.	36936	1636	0,04	0,01	3,72	60799	4455	0,07	0,00	2,31
372.	25841	2380	0,09	0,29	2,82	20972	5291	0,25	0,41	2,95
373.	39117	1665	0,04	0,50	3,60	22166	2406	0,11	0,45	2,70
374.	18160	1283	0,07	-0,18	2,70	28015	1985	0,07	0,00	2,61
375.	9541	2369	0,25	-0,03	2,10	12567	1611	0,13	0,02	2,31
376.	26758	2139	0,08	0,37	3,08	20576	4947	0,24	0,03	2,07
377.	52930	2441	0,05	-0,66	3,79	36077	3948	0,11	0,12	3,38
378.	27469	4537	0,17	-0,21	2,28	34159	1826	0,05	0,03	2,91
379.	41186	1657	0,04	0,68	3,66	27553	5284	0,19	0,47	2,48
380.	31053	2610	0,08	-0,27	3,12	25795	1172	0,05	0,09	2,60
381.	23806	2539	0,11	-0,36	2,74	34018	5530	0,16	-0,05	2,87
382.	24976	2271	0,09	0,00	2,89	40957	1924	0,05	0,05	3,55
383.	43942	1749	0,04	-0,04	2,52	53729	5568	0,10	-0,06	4,40
384.	12689	3440	0,27	-0,22	2,12	14357	5811	0,40	-0,14	2,20
385.	43420	3884	0,09	-0,13	2,59	45221	2383	0,05	0,42	3,25
386.	40323	2241	0,06	-0,14	3,76	57876	4178	0,07	0,20	3,08
387.	33431	4230	0,13	-0,55	3,07	29116	1836	0,06	-0,15	2,62
388.	40270	1925	0,05	0,35	3,48	57389	5198	0,09	-0,28	3,57
389.	23179	2981	0,13	0,13	3,21	35393	5864	0,17	0,16	3,09
390.	29044	3124	0,11	0,62	2,55	51901	2039	0,04	-0,27	3,24
391.	39211	3878	0,10	-0,09	2,82	37997	1769	0,05	0,22	3,22
392.	38704	2915	0,08	-0,14	2,74	62828	3962	0,06	-0,20	2,47
393.	31508	2667	0,08	-0,16	2,45	51693	6157	0,12	0,04	2,58
394.	19179	1344	0,07	-0,10	2,41	18366	1962	0,11	0,02	3,33
395.	32326	1737	0,05	0,43	3,27	24752	2091	0,08	-0,31	2,96
396.	44369	3128	0,07	-0,16	3,47	27075	1230	0,05	0,47	3,68
397.	18323	1608	0,09	0,35	2,72	32389	1436	0,04	0,21	2,91
398.	29996	2059	0,07	0,29	3,35	49361	2984	0,06	-0,07	3,20
399.	21878	2033	0,09	0,75	2,98	39401	1593	0,04	0,10	2,43
400.	49681	3076	0,06	-0,11	2,20	32633	2045	0,06	0,48	3,97
401.	29988	1649	0,06	0,13	2,95	39759	3509	0,09	-0,45	2,67
402.	17583	937	0,05	0,27	2,68	32312	1496	0,05	-0,37	2,79
403.	26248	4650	0,18	0,37	2,48	40269	1885	0,05	0,34	3,15
404.	28398	1610	0,06	-0,21	2,53	36779	2842	0,08	-0,26	2,96
405.	14553	2206	0,15	-0,02	2,53	21883	1154	0,05	-0,25	2,48
406.	24307	2139	0,09	-0,27	3,46	25080	4485	0,18	-0,43	3,48
407.	25353	2554	0,10	-0,08	2,23	29960	6188	0,21	-0,14	2,42
408.	33072	1754	0,05	0,44	3,30	60234	3590	0,06	-0,31	3,56
409.	38320	2886	0,08	-0,24	2,52	44106	3077	0,07	-0,24	3,28
410.	20544	1594	0,08	-0,35	3,18	24370	1496	0,06	-0,25	3,05
411.	28190	5296	0,19	0,14	2,79	26760	2063	0,08	-0,03	2,94
412.	46916	2993	0,06	0,22	3,09	32798	2232	0,07	-0,01	2,80
413.	31412	1872	0,06	0,31	2,73	37116	3238	0,09	-0,20	2,74
414.	33489	4493	0,13	0,04	3,25	43348	1948	0,04	0,53	3,70
415.	23696	2737	0,12	-0,04	3,84	33147	5974	0,18	-0,03	3,85
416.	22390	3004	0,13	0,00	2,51	20191	6080	0,30	0,03	2,29
417.	25732	1832	0,07	0,96	4,98	48319	2027	0,04	-0,28	3,04
418.	17974	1321	0,07	-0,21	2,82	31115	1534	0,05	-0,06	3,50
419.	41911	1968	0,05	0,59	3,53	41756	5310	0,13	-0,07	2,28
420.	23017	1311	0,06	0,05	2,70	17865	2868	0,16	-0,07	2,33
421.	39661	2563	0,06	-0,05	3,06	52662	3661	0,07	-0,58	3,41
422.	53741	2020	0,04	0,12	2,67	31140	3461	0,11	0,60	3,12
423.	40990	2458	0,06	0,29	3,83	36963	2742	0,07	-0,20	2,78
424.	41372	1702	0,04	0,08	3,04	35258	4818	0,14	-0,13	2,87
425.	19282	1828	0,09	-0,42	2,51	24577	3941	0,16	-0,29	2,30
426.	31823	1981	0,06	0,50	3,63	41443	4672	0,11	0,44	3,73
427.	35408	2112	0,06	-0,50	3,48	27194	5205	0,19	-0,27	2,58
428.	21288	2024	0,10	0,56	2,41	14385	4071	0,28	0,76	2,92
429.	54847	3032	0,06	0,19	2,48	43081	4215	0,10	-0,07	2,40
430.	44552	1969	0,04	0,15	2,22	37359	4604	0,12	-0,06	2,70
431.	33416	1950	0,06	0,62	4,22	62581	2777	0,04	-0,15	3,06
432.	31739	1685	0,05	0,16	2,71	44881	3888	0,09	-0,21	2,72
433.	27772	1593	0,06	-0,17	3,29	24923	2222	0,09	-0,16	3,82
434.	23412	1943	0,08	0,61	2,49	15000	3929	0,26	0,64	2,47

inst. n.	Distribution of CVP total effort ( $\lambda=200$ )					Distribution of CIBI total effort ( $\lambda=200$ )				
	$\mu$	$\sigma$	CV	skew	kurt	$\mu$	$\sigma$	CV	skew	kurt
435.	47435	2928	0,06	-0,31	3,26	62925	4781	0,08	0,22	2,42
436.	47069	1737	0,04	0,20	2,93	26817	2487	0,09	0,27	2,05
437.	44690	2499	0,06	-0,18	3,07	42416	3891	0,09	-0,03	3,40
438.	26231	1526	0,06	0,12	3,12	25024	2771	0,11	-0,29	2,67
439.	26541	1882	0,07	0,09	2,40	47548	2177	0,05	-0,11	2,86
440.	24321	3378	0,14	-0,25	2,56	30917	1676	0,05	-0,18	2,95
441.	40501	3166	0,08	-0,34	2,81	32372	1948	0,06	-0,29	3,00
442.	23256	2204	0,09	-0,24	2,81	23347	4998	0,21	-0,20	2,82
443.	36598	1823	0,05	0,05	2,93	63120	3487	0,06	-0,16	2,80
444.	21617	2850	0,13	0,12	3,57	19527	1361	0,07	0,05	3,19
445.	56265	2473	0,04	-0,27	3,67	39965	4161	0,10	0,15	3,14
446.	20817	2244	0,11	-0,37	3,07	19634	1311	0,07	0,29	2,86
447.	34459	4360	0,13	0,02	2,85	38477	1828	0,05	0,22	2,73
448.	18389	4436	0,24	0,20	2,30	24254	2007	0,08	-0,04	2,39
449.	40581	2145	0,05	0,94	6,21	40570	4316	0,11	-0,40	3,05
450.	13125	1165	0,09	-0,14	2,74	17859	1332	0,07	0,06	2,84
451.	42809	1797	0,04	-0,07	3,35	41279	5294	0,13	-0,44	3,19
452.	40893	4326	0,11	0,10	2,52	58190	2898	0,05	-0,07	2,33
453.	17487	1450	0,08	-0,04	3,11	19253	1354	0,07	-0,06	2,80
454.	25160	1340	0,05	0,27	2,54	38592	2869	0,07	0,20	3,64
455.	17906	1718	0,10	-0,01	3,88	18166	3611	0,20	-0,20	2,71
456.	21742	4815	0,22	0,58	4,12	28157	1934	0,07	0,28	2,94
457.	29088	2685	0,09	-0,15	2,75	47476	6317	0,13	-0,01	2,64
458.	30402	1716	0,06	-0,56	4,48	19607	1317	0,07	-0,25	2,76
459.	48193	2417	0,05	-0,32	3,45	31365	2839	0,09	-0,28	2,89
460.	36245	2136	0,06	-0,07	2,54	38815	6022	0,16	0,07	3,10
461.	24783	1225	0,05	0,65	3,66	39301	2978	0,08	0,06	2,36
462.	44670	3620	0,08	0,02	2,49	54358	3370	0,06	0,16	2,66
463.	31400	2542	0,08	-0,35	2,60	48312	2159	0,04	-0,29	3,06
464.	50486	2388	0,05	0,12	4,21	55322	4155	0,08	-0,37	3,65
465.	44929	2206	0,05	0,17	3,02	63787	5313	0,08	-0,10	3,91
466.	17460	2086	0,12	-0,07	2,55	14810	4024	0,27	-0,10	2,28
467.	45219	1945	0,04	-0,51	4,19	49796	6399	0,13	-0,17	3,06
468.	38028	1773	0,05	0,14	2,22	70083	5565	0,08	-0,32	3,13
469.	8448	2665	0,32	0,40	2,29	8366	4071	0,49	0,42	2,31
470.	41673	2087	0,05	0,40	2,55	26147	5247	0,20	0,64	2,77
471.	23055	1406	0,06	0,30	2,77	31953	3228	0,10	-0,46	2,75
472.	36534	1414	0,04	-0,03	3,79	53030	4479	0,08	-0,09	2,94
473.	49282	1864	0,04	0,14	3,06	37997	6441	0,17	-0,25	2,55
474.	20485	4370	0,21	-0,01	2,46	26761	1862	0,07	-0,14	2,25
475.	31436	2011	0,06	-0,37	3,08	39728	5455	0,14	-0,03	2,97
476.	26163	1686	0,06	0,00	3,42	27692	3924	0,14	-0,48	3,39
477.	38031	2142	0,06	0,23	2,98	65856	3340	0,05	-0,41	2,79
478.	14781	3056	0,21	0,09	2,76	17762	1592	0,09	-0,08	3,20
479.	10495	2670	0,25	-0,25	2,48	12707	4038	0,32	-0,19	2,37
480.	14471	4653	0,32	0,27	1,90	17964	2401	0,13	0,12	1,97
481.	10345	2150	0,21	-0,27	2,15	11834	3137	0,27	-0,21	2,35
482.	46041	1509	0,03	0,06	2,79	26221	3057	0,12	0,73	3,51
483.	33371	1722	0,05	-0,64	4,36	26539	2618	0,10	0,14	2,88
484.	13630	3523	0,26	0,20	2,24	14165	6245	0,44	0,23	2,40
485.	42605	2221	0,05	0,05	2,79	43473	4229	0,10	0,03	3,29
486.	45835	2816	0,06	-0,68	4,90	28165	1616	0,06	0,14	2,85
487.	14545	2537	0,17	0,37	2,28	11791	4511	0,38	0,38	2,06
488.	46257	1923	0,04	-0,03	2,50	35828	4852	0,14	-0,18	2,86
489.	41570	1736	0,04	-0,21	2,54	27692	5138	0,19	0,13	2,08
490.	39436	4053	0,10	-0,05	2,79	40867	1845	0,05	0,48	3,29
491.	34810	2221	0,06	0,01	3,47	44111	6184	0,14	0,14	3,70
492.	22449	3190	0,14	0,54	2,86	17938	6822	0,38	0,72	3,32
493.	54843	2971	0,05	-0,17	2,56	40100	3621	0,09	0,21	2,72
494.	32016	1846	0,06	-0,03	2,99	28633	2821	0,10	-0,35	3,15
495.	51971	2080	0,04	0,18	2,98	43972	6121	0,14	-0,24	2,32
496.	29309	3648	0,12	-0,12	2,60	25673	1508	0,06	-0,31	2,77
497.	36524	1429	0,04	0,65	3,84	68804	4216	0,06	-0,05	3,09
498.	22920	2187	0,10	0,42	2,94	16641	4367	0,26	0,21	2,16
499.	39464	1938	0,05	0,15	2,65	31137	6009	0,19	-0,01	2,59
500.	24465	1022	0,04	-0,41	3,62	43159	3215	0,07	-0,78	3,24
501.	52362	2409	0,05	-0,17	2,77	30747	2738	0,09	0,54	3,48
502.	40757	2248	0,06	0,01	3,01	37958	3260	0,09	-0,04	3,01
503.	13399	3198	0,24	0,13	1,94	19735	1659	0,08	0,08	2,23
504.	22762	1263	0,06	0,18	2,91	31798	3056	0,10	0,34	2,78
505.	29441	1763	0,06	1,02	5,39	18844	2312	0,12	0,11	2,54
506.	42130	2128	0,05	0,14	2,28	35349	6695	0,19	-0,06	2,48
507.	40594	2224	0,05	-0,01	2,55	54419	4276	0,08	-0,38	2,77
508.	40956	1857	0,05	0,37	2,70	48462	3971	0,08	-0,28	3,33
509.	55852	2292	0,04	0,13	2,60	40160	5452	0,14	-0,05	2,77
510.	33653	1905	0,06	0,18	3,56	48893	3175	0,06	-0,13	2,87
511.	22684	4278	0,19	-0,44	2,62	25568	1912	0,07	-0,41	2,77
512.	9633	3824	0,40	0,62	2,14	13139	2227	0,17	0,47	2,13
513.	30683	1646	0,05	-0,27	2,84	52163	3210	0,06	-0,15	2,44
514.	37420	1621	0,04	0,24	2,64	23041	2484	0,11	0,14	2,55
515.	27613	2451	0,09	0,36	2,87	47833	2292	0,05	0,07	2,46

inst. n.	Distribution of CVP total effort ( $\lambda=200$ )					Distribution of CIBI total effort ( $\lambda=200$ )				
	$\mu$	$\sigma$	CV	skew	kurt	$\mu$	$\sigma$	CV	skew	kurt
516.	43277	1890	0,04	0,44	3,42	26243	4073	0,16	0,43	2,36
517.	9535	3158	0,33	0,10	3,27	10983	5096	0,46	0,33	3,48
518.	27717	2403	0,09	0,87	3,08	50859	1899	0,04	-0,19	3,53
519.	38091	2317	0,06	0,15	2,91	41580	3373	0,08	-0,30	2,72
520.	18466	4047	0,22	-0,25	2,54	27198	7358	0,27	-0,20	2,50
521.	17707	3485	0,20	0,49	3,56	25302	1701	0,07	0,21	2,54
522.	49519	2121	0,04	0,15	3,12	34935	4491	0,13	0,25	2,53
523.	26394	1714	0,06	-0,63	3,66	38230	4433	0,12	-0,23	3,05
524.	20599	1238	0,06	0,19	2,51	36717	2314	0,06	-0,95	3,85
525.	35375	1753	0,05	-0,08	2,48	53499	3947	0,07	-0,20	2,66
526.	24873	1561	0,06	-0,10	2,18	39055	2772	0,07	-0,01	2,41
527.	45702	1915	0,04	0,52	4,22	32438	4130	0,13	0,10	2,74
528.	36065	1669	0,05	0,08	2,87	58666	5448	0,09	-0,17	2,39
529.	26043	5036	0,19	0,48	2,42	39610	1943	0,05	0,10	2,55
530.	26203	1529	0,06	-0,09	3,24	18417	1569	0,09	-0,59	3,35
531.	16173	1300	0,08	-0,22	2,47	10730	2045	0,19	-0,10	2,22
532.	24530	2313	0,09	-0,56	3,88	28613	1628	0,06	0,08	3,26
533.	11723	2146	0,18	-0,18	2,66	12455	3556	0,29	0,07	3,11
534.	25220	1248	0,05	0,06	3,14	42599	3589	0,08	-0,39	3,00
535.	33487	2254	0,07	0,17	2,88	47323	3139	0,07	-0,34	3,28
536.	31631	2912	0,09	-0,33	2,34	48717	2582	0,05	-0,39	4,37
537.	35248	1875	0,05	0,61	3,11	41265	5597	0,14	-0,14	3,02
538.	18734	1558	0,08	-0,07	2,61	24930	1814	0,07	-0,09	3,08
539.	23840	2302	0,10	0,26	2,60	39884	1807	0,05	-0,08	2,99
540.	17004	3580	0,21	-0,08	3,10	22633	6278	0,28	0,03	2,99
541.	39631	4370	0,11	0,30	2,03	66917	2703	0,04	-0,08	3,06
542.	29655	1783	0,06	-0,31	3,35	49501	4576	0,09	-0,25	2,76
543.	14216	1233	0,09	0,35	2,67	14232	1368	0,10	-0,19	3,21
544.	42885	2301	0,05	-0,24	4,07	25761	2017	0,08	0,13	2,82
545.	37838	1985	0,05	0,13	2,21	27966	5839	0,21	0,28	2,26
546.	37907	1979	0,05	0,18	3,39	34898	3587	0,10	-0,23	3,05
547.	21949	3372	0,15	-0,11	2,53	24992	1521	0,06	0,41	2,91
548.	42661	2221	0,05	-0,19	3,00	38353	3076	0,08	-0,32	2,69
549.	34740	1942	0,06	0,34	2,49	65375	2795	0,04	-0,22	2,75
550.	21438	1714	0,08	-0,34	2,77	19548	1369	0,07	0,22	2,32
551.	30848	1516	0,05	0,22	2,61	21496	3172	0,15	-0,02	2,44
552.	24706	5023	0,20	0,21	2,66	25058	2124	0,08	0,01	2,43
553.	24064	2647	0,11	-0,16	2,90	39898	5736	0,14	0,08	2,66
554.	19726	2027	0,10	-0,11	3,08	17647	1202	0,07	0,24	3,02
555.	34475	3813	0,11	-0,27	2,77	30650	1428	0,05	0,14	3,19
556.	32407	1767	0,05	0,27	2,65	30127	4156	0,14	-0,21	2,43
557.	49648	2031	0,04	-0,02	3,51	35042	3856	0,11	-0,36	2,80
558.	43589	2920	0,07	-0,27	3,65	49351	3199	0,06	0,12	3,12
559.	19463	1227	0,06	0,16	2,57	27223	2704	0,10	-0,22	3,43
560.	22944	1221	0,05	0,28	3,27	27826	2719	0,10	0,10	3,16
561.	36985	2309	0,06	0,10	2,32	51039	2968	0,06	-0,32	3,47
562.	35837	1845	0,05	-0,50	3,80	29476	3284	0,11	-0,19	2,67
563.	31668	1899	0,06	-0,18	2,56	22461	4327	0,19	0,08	2,66
564.	31890	3203	0,10	-0,07	2,67	44055	2151	0,05	0,44	3,21
565.	12547	5034	0,40	0,86	2,99	16098	2599	0,16	0,73	2,92
566.	35462	1747	0,05	0,04	3,12	27132	2391	0,09	-0,15	2,68
567.	21091	1548	0,07	-0,03	3,55	25044	1952	0,08	0,00	3,45
568.	40263	2018	0,05	-0,10	2,47	46872	6020	0,13	0,27	2,80
569.	23661	2153	0,09	-0,08	2,45	37624	1562	0,04	0,09	2,88
570.	51757	2003	0,04	-0,28	3,09	28279	2493	0,09	0,82	3,08
571.	31233	2233	0,07	0,16	3,06	38182	2184	0,06	-0,23	2,63
572.	30945	1647	0,05	0,16	3,01	35760	4378	0,12	0,04	2,79
573.	22999	4011	0,17	0,42	2,33	34488	1715	0,05	0,44	3,19
574.	33097	2587	0,08	-0,56	3,65	32697	7037	0,22	-0,30	2,59
575.	21882	4588	0,21	0,50	2,65	33482	1788	0,05	0,57	2,83
576.	38709	1400	0,04	0,38	3,00	22506	3427	0,15	0,69	2,54
577.	28627	2222	0,08	-0,08	2,09	20809	5037	0,24	0,04	1,95
578.	40406	1970	0,05	-0,15	2,31	52236	4827	0,09	-0,19	2,55
579.	15057	2655	0,18	-0,63	2,95	19721	4571	0,23	-0,36	2,50
580.	22267	1272	0,06	0,15	2,59	16220	1862	0,11	-0,55	3,50
581.	18134	1835	0,10	-0,80	4,09	24842	3965	0,16	-0,39	3,31
582.	39236	3414	0,09	0,29	3,18	65049	3182	0,05	0,11	3,29
583.	25605	3413	0,13	-0,49	3,29	25155	1317	0,05	-0,09	2,80
584.	36554	2822	0,08	-0,24	2,90	23653	1330	0,06	0,50	3,20
585.	34690	1687	0,05	0,13	2,99	53819	4541	0,08	-0,52	3,69
586.	38021	3814	0,10	0,12	3,29	40250	1960	0,05	-0,01	3,09
587.	33683	2062	0,06	-0,15	2,42	54779	3175	0,06	-0,35	2,73
588.	26556	1966	0,07	0,10	2,23	33497	4747	0,14	0,04	2,22
589.	40056	1956	0,05	0,01	2,82	54212	4902	0,09	-0,41	3,35
590.	54939	1900	0,03	-0,17	2,76	37713	6200	0,16	0,03	2,31
591.	20298	2118	0,10	-0,58	3,44	22856	4298	0,19	-0,63	3,18
592.	19398	4346	0,22	-0,01	3,03	22596	2061	0,09	-0,34	2,87
593.	32986	1413	0,04	0,01	2,52	62739	4127	0,07	-0,41	2,72
594.	43512	2965	0,07	0,63	4,43	65049	4581	0,07	-0,14	4,24
595.	30842	3070	0,10	-0,13	2,53	46144	7555	0,16	0,33	3,01
596.	25025	2155	0,09	-0,18	2,64	33570	5347	0,16	-0,03	2,65

inst.	Distribution of CVP total effort ( $\lambda=200$ )					Distribution of CIBI total effort ( $\lambda=200$ )				
	n.	$\mu$	$\sigma$	CV	skew	kurt	$\mu$	$\sigma$	CV	skew
597.	41791	2621	0,06	0,27	3,11	61798	3559	0,06	-0,39	3,11
598.	52413	2858	0,05	-0,15	2,81	30661	1772	0,06	-0,16	2,77
599.	37568	2614	0,07	-0,92	4,68	28536	2073	0,07	-0,12	2,33
600.	41878	1788	0,04	0,09	2,91	30806	4304	0,14	-0,13	3,02
601.	44597	4356	0,10	-0,42	2,95	28222	1420	0,05	-0,59	3,72
602.	45826	3557	0,08	-0,30	2,44	48546	2544	0,05	-0,51	2,82
603.	40910	1819	0,04	0,54	4,15	27299	6159	0,23	0,85	3,31
604.	48064	2596	0,05	0,16	2,77	53624	4362	0,08	-0,11	3,28
605.	18301	2865	0,16	-0,10	2,52	20910	1285	0,06	0,40	2,81
606.	47660	2085	0,04	0,33	3,04	40571	5826	0,14	-0,58	3,21
607.	27692	2152	0,08	-0,92	5,97	31924	2060	0,06	-0,27	2,82
608.	28492	1154	0,04	-0,42	3,65	52521	3355	0,06	-0,43	3,34
609.	34747	1394	0,04	0,26	2,86	19809	2439	0,12	0,34	2,12
610.	48177	3137	0,07	-0,43	3,59	65697	3893	0,06	-0,29	4,28
611.	46832	3679	0,08	0,15	2,57	49928	3214	0,06	0,37	2,86
612.	28608	1404	0,05	0,25	3,11	42548	3229	0,08	0,25	3,34
613.	10659	3435	0,32	-0,12	2,79	13642	5578	0,41	-0,06	2,91
614.	40292	2871	0,07	0,12	2,65	62574	4381	0,07	-0,32	3,02
615.	38677	1260	0,03	0,40	3,17	23008	3454	0,15	0,50	2,27
616.	24886	2498	0,10	-0,13	2,80	32128	5282	0,16	-0,17	3,05
617.	23050	1384	0,06	0,15	3,09	30322	2046	0,07	-0,02	2,92
618.	30581	2232	0,07	-0,06	2,44	45930	2399	0,05	-0,22	2,96
619.	31604	3556	0,11	0,22	2,34	51850	1912	0,04	0,08	3,50
620.	16913	1266	0,07	-0,07	2,54	13925	1785	0,13	-0,57	2,95
621.	24469	1827	0,07	-0,03	2,63	41425	2355	0,06	-0,18	2,33
622.	24907	3000	0,12	0,21	2,35	20749	6405	0,31	0,16	2,04
623.	25114	1540	0,06	-0,78	5,28	35104	4110	0,12	-0,42	3,17
624.	29977	3601	0,12	0,01	2,29	47091	1847	0,04	0,21	2,53
625.	46782	2565	0,05	0,11	2,89	48993	4082	0,08	0,11	2,79
626.	27803	3273	0,12	-0,28	2,85	39279	1785	0,05	0,08	2,49
627.	43122	2131	0,05	0,01	3,03	44155	5104	0,12	0,08	2,51
628.	32100	1482	0,05	0,44	3,17	58361	4199	0,07	-0,49	2,97
629.	17506	4850	0,28	0,82	3,61	23088	2335	0,10	0,74	3,62
630.	39059	3437	0,09	-0,64	3,54	30046	1729	0,06	-0,02	3,26
631.	50739	3271	0,06	-0,64	4,07	45035	2825	0,06	-0,12	2,56
632.	44140	1536	0,03	-0,01	2,81	25800	3224	0,12	0,77	3,35
633.	52040	2138	0,04	0,44	3,53	28371	1913	0,07	0,43	2,95
634.	23722	1373	0,06	0,04	2,50	24083	3079	0,13	-0,17	2,63
635.	30118	2770	0,09	-0,08	2,87	26112	6755	0,26	0,18	2,86
636.	17686	1698	0,10	-0,04	2,80	23344	1281	0,05	-0,09	3,24
637.	15020	2100	0,14	0,05	2,38	12070	3609	0,30	0,17	2,35
638.	29294	2449	0,08	-0,45	4,41	23780	1372	0,06	0,13	2,80
639.	41303	4769	0,12	-0,37	2,86	29046	1734	0,06	-0,58	3,47
640.	21093	1481	0,07	0,32	2,39	39603	1694	0,04	-0,27	3,48
641.	30962	2935	0,09	-0,26	3,13	45959	2202	0,05	-0,62	3,41
642.	26198	2932	0,11	0,87	4,24	46124	1660	0,04	-0,34	2,92
643.	20596	1909	0,09	0,46	2,84	37132	1483	0,04	0,22	2,96
644.	41448	2768	0,07	-0,04	3,37	63318	4057	0,06	-0,40	4,61
645.	35213	3318	0,09	-0,16	3,11	48513	2194	0,05	-0,62	4,99
646.	25453	2032	0,08	0,23	2,73	21087	4854	0,23	0,28	3,46
647.	50008	2047	0,04	0,01	2,79	33573	3181	0,09	-0,17	2,82
648.	39632	3002	0,08	-0,11	2,14	61913	3757	0,06	-0,34	3,39
649.	19685	3144	0,16	-0,52	3,90	25459	6224	0,24	-0,08	3,92
650.	25227	1460	0,06	0,28	2,60	31829	2388	0,08	-0,39	2,75
651.	31183	2332	0,07	-0,91	3,99	54228	5553	0,10	-0,47	2,88
652.	25972	4944	0,19	-0,06	2,87	27757	2058	0,07	0,03	2,63
653.	56331	2321	0,04	-0,21	2,74	36276	5236	0,14	0,46	2,67
654.	47580	1943	0,04	0,23	3,35	39957	4940	0,12	-0,18	3,06
655.	25717	2080	0,08	-0,41	3,33	42914	4775	0,11	-0,10	2,91
656.	26482	2103	0,08	0,12	2,65	46962	1975	0,04	0,01	2,54
657.	25023	4264	0,17	-0,49	2,83	25415	1794	0,07	-0,32	3,39
658.	26955	3110	0,12	0,00	2,54	32444	6959	0,21	0,08	2,77
659.	40621	1843	0,05	-0,13	3,17	25211	1814	0,07	-0,02	2,68
660.	18788	1320	0,07	-0,17	3,81	17135	2491	0,15	-0,47	3,08
661.	11653	1572	0,13	0,11	3,42	11721	2373	0,20	0,10	3,82
662.	32961	3724	0,11	0,82	2,97	58042	2155	0,04	-0,11	2,75
663.	11506	1403	0,12	-0,75	4,37	12206	1962	0,16	-0,60	4,24
664.	53664	1912	0,04	0,14	2,72	31435	3628	0,12	0,48	2,62
665.	22347	1761	0,08	0,10	3,60	36424	1991	0,05	-0,25	3,19
666.	36019	1807	0,05	-0,21	2,50	24107	1855	0,08	-0,07	2,61
667.	52696	1946	0,04	-0,11	3,18	32500	3498	0,11	0,19	2,64
668.	40679	2578	0,06	-0,34	2,83	23373	1377	0,06	-0,22	2,32
669.	46026	1730	0,04	0,31	3,31	28206	4763	0,17	0,77	2,90
670.	39284	1712	0,04	0,37	2,94	67667	5541	0,08	-0,10	2,89
671.	11270	3418	0,30	0,25	2,30	12703	2084	0,16	0,16	2,32
672.	28237	1734	0,06	-0,07	2,37	26153	1908	0,07	0,08	2,51
673.	25581	1467	0,06	0,46	3,27	30441	2735	0,09	0,07	1,99
674.	19382	1479	0,08	-0,05	2,90	26345	1656	0,06	0,08	2,60
675.	28084	1594	0,06	0,05	3,67	45152	4274	0,09	-0,32	3,14
676.	18548	2083	0,11	-0,21	2,58	20261	1168	0,06	-0,05	2,72
677.	12626	2946	0,23	0,50	2,60	18498	1629	0,09	0,25	2,39



inst. n.	Distribution of CVP total effort ( $\lambda=200$ )					Distribution of CIBI total effort ( $\lambda=200$ )				
	$\mu$	$\sigma$	CV	skew	kurt	$\mu$	$\sigma$	CV	skew	kurt
678.	25490	2135	0,08	-0,43	3,08	26216	4939	0,19	-0,23	2,93
679.	21918	2240	0,10	0,59	2,81	38326	1427	0,04	-0,27	2,80
680.	21819	1242	0,06	-0,01	3,26	20653	1613	0,08	-0,26	3,47
681.	25660	3688	0,14	-0,28	3,41	26286	1548	0,06	0,32	3,54
682.	30087	1320	0,04	0,32	3,87	49737	3562	0,07	0,61	3,45
683.	24540	2010	0,08	-1,21	6,34	39191	4674	0,12	-1,09	6,06
684.	18139	1361	0,08	0,51	3,42	33204	1462	0,04	0,08	2,51
685.	41136	2804	0,07	-0,25	5,17	26563	1819	0,07	-0,08	2,90
686.	29116	2048	0,07	0,12	2,32	25817	5112	0,20	0,01	2,50
687.	11096	1275	0,11	-0,71	3,91	12078	1663	0,14	-0,52	3,71
688.	27724	1986	0,07	-0,20	3,31	32903	4383	0,13	-0,10	2,54
689.	33316	1367	0,04	-0,09	3,24	20901	2655	0,13	0,09	2,38
690.	46055	2468	0,05	-0,06	2,96	51898	4905	0,09	0,05	2,67
691.	30748	3566	0,12	-0,12	2,45	46908	2265	0,05	-0,07	2,79
692.	25283	2533	0,10	-0,29	2,66	31585	1732	0,05	0,04	3,32
693.	28463	4284	0,15	0,47	2,47	45043	1554	0,03	-0,08	2,44
694.	10306	2500	0,24	-0,06	2,10	14562	1452	0,10	0,01	2,61
695.	33948	2789	0,08	-0,25	2,47	22353	1468	0,07	0,00	2,52
696.	23957	3770	0,16	-0,29	2,64	27031	7872	0,29	0,01	2,71
697.	53230	1906	0,04	0,31	3,31	39807	5743	0,14	-0,39	2,71
698.	31376	1483	0,05	0,18	3,29	23226	3246	0,14	-0,27	2,84
699.	20239	1523	0,08	-0,30	3,00	30142	3289	0,11	-0,16	3,10
700.	34931	4575	0,13	-0,01	2,90	37877	1442	0,04	-0,03	2,74
701.	22493	3496	0,16	0,13	2,45	22109	7201	0,33	0,27	2,71
702.	57428	2313	0,04	0,38	3,10	34146	5117	0,15	0,92	3,81
703.	41917	2663	0,06	0,31	3,29	27025	1638	0,06	-0,45	3,04
704.	25489	2651	0,10	-0,14	3,31	29013	1467	0,05	0,44	3,27
705.	46827	2111	0,05	0,56	3,42	28646	5539	0,19	1,00	3,54
706.	39012	2421	0,06	0,34	3,14	72356	3752	0,05	0,02	2,77
707.	31175	1442	0,05	0,43	3,14	48132	3739	0,08	-0,20	2,91
708.	27983	1493	0,05	-0,02	3,47	44206	3296	0,07	-0,23	2,88
709.	35420	1520	0,04	-0,24	3,04	67477	4694	0,07	-0,47	2,71
710.	27744	3934	0,14	0,69	2,90	45960	1753	0,04	0,06	2,59
711.	37124	4131	0,11	-0,15	2,96	35672	1829	0,05	0,05	3,00
712.	23158	1366	0,06	0,44	3,75	27091	3157	0,12	-0,81	4,16
713.	28911	1820	0,06	0,42	2,48	19365	3899	0,20	0,24	2,20
714.	22518	3155	0,14	-0,09	2,58	27193	6362	0,23	0,20	2,94
715.	27066	3174	0,12	-0,23	2,56	27663	7236	0,26	0,02	2,45
716.	45957	1865	0,04	0,33	3,12	32706	4429	0,14	0,03	2,78
717.	43371	1730	0,04	-0,33	3,44	56557	4419	0,08	0,04	2,71
718.	16531	1256	0,08	0,68	4,90	15193	2167	0,14	-0,13	2,63
719.	45417	2163	0,05	-0,01	2,75	49221	6397	0,13	-0,30	2,53
720.	35853	5619	0,16	-0,24	3,06	26236	2058	0,08	-0,51	3,46
721.	40858	2464	0,06	0,17	2,80	24441	1450	0,06	0,29	3,28
722.	36557	3286	0,09	-0,32	2,91	27606	1313	0,05	0,46	3,88
723.	20393	1492	0,07	-0,49	2,91	28569	3373	0,12	-0,18	2,93
724.	32963	1571	0,05	0,39	3,62	36788	4399	0,12	0,00	2,72
725.	37217	1601	0,04	0,06	2,44	65856	4273	0,06	-0,63	4,00
726.	24714	3103	0,13	-0,19	3,02	31294	6501	0,21	-0,08	2,76
727.	39209	1697	0,04	0,23	2,45	52239	5094	0,10	-0,02	2,78
728.	45985	1779	0,04	0,49	3,70	26231	2341	0,09	0,38	2,37
729.	31385	1674	0,05	0,51	2,91	21724	3474	0,16	0,26	3,11
730.	27743	4907	0,18	0,90	3,30	40688	1813	0,04	0,31	2,92
731.	21257	1324	0,06	-0,12	2,89	36283	1962	0,05	-0,42	2,61
732.	34868	2161	0,06	0,17	2,60	27018	5444	0,20	0,00	2,52
733.	9843	2831	0,29	0,55	2,53	9270	4617	0,50	0,68	2,67
734.	35739	1755	0,05	0,42	3,22	38422	4866	0,13	-0,08	2,32
735.	28805	3277	0,11	-0,41	2,98	39124	1775	0,05	-0,06	2,99
736.	40346	1832	0,05	-0,03	3,08	67996	4222	0,06	0,01	2,63
737.	17326	3608	0,21	0,83	3,68	27254	1549	0,06	0,70	3,45
738.	39287	2930	0,07	-0,21	2,79	57076	3574	0,06	-0,36	3,41
739.	26365	2917	0,11	-0,17	2,89	36292	1838	0,05	-0,53	2,63
740.	35064	2566	0,07	0,26	3,16	58248	2784	0,05	-0,44	3,53
741.	28680	1990	0,07	-0,49	3,02	26148	5458	0,21	0,08	2,83
742.	39113	3327	0,09	0,20	2,19	64412	3390	0,05	-0,16	3,00
743.	19257	2376	0,12	0,16	2,74	17504	4709	0,27	0,32	3,01
744.	11231	2247	0,20	-0,78	3,01	14498	1520	0,10	-0,53	2,94
745.	30210	3549	0,12	-0,45	2,55	26111	1634	0,06	0,29	3,21
746.	44096	2526	0,06	0,23	3,37	43276	3658	0,08	-0,16	3,70
747.	34951	2248	0,06	-0,27	2,98	63369	6036	0,10	-0,10	2,97
748.	21479	1906	0,09	0,50	2,58	14247	3943	0,28	0,67	2,50
749.	40597	1917	0,05	0,09	2,46	55373	5902	0,11	-0,42	4,12
750.	36533	4152	0,11	0,11	2,40	59612	2599	0,04	0,20	3,52
751.	48809	2806	0,06	-0,31	2,66	58203	5015	0,09	-0,24	2,69
752.	38775	1678	0,04	-0,02	2,73	57742	5136	0,09	-0,25	2,40
753.	32887	2509	0,08	-0,20	2,61	45861	6771	0,15	-0,09	2,66
754.	12547	3096	0,25	0,55	2,80	11393	5229	0,46	0,62	2,88
755.	26778	2119	0,08	-0,19	2,85	38768	2168	0,06	-0,16	3,05
756.	38165	2004	0,05	-0,26	2,86	51717	3472	0,07	-0,34	2,64
757.	45038	2532	0,06	-0,03	2,92	58109	4607	0,08	-0,19	2,80
758.	43711	2030	0,05	-0,06	3,31	23342	1540	0,07	0,55	2,95

inst. n.	Distribution of CVP total effort ( $\lambda=200$ )					Distribution of CIBI total effort ( $\lambda=200$ )				
	$\mu$	$\sigma$	CV	skew	kurt	$\mu$	$\sigma$	CV	skew	kurt
759.	46337	2385	0,05	-0,25	2,89	43927	4090	0,09	-0,11	3,00
760.	34938	4341	0,12	-0,17	2,89	48327	1833	0,04	-0,14	2,95
761.	25904	1934	0,07	-0,09	2,80	21470	4166	0,19	-0,13	2,42
762.	21441	3650	0,17	-0,25	2,99	20472	1785	0,09	-0,32	2,98
763.	23500	1695	0,07	0,06	3,45	21840	3743	0,17	-0,15	2,98
764.	35782	1692	0,05	-0,42	3,64	58738	5291	0,09	-0,11	2,80
765.	37482	3013	0,08	-0,35	2,96	32049	2003	0,06	-0,14	2,54
766.	26969	1503	0,06	-0,17	2,97	34221	2957	0,09	-0,08	2,61
767.	37711	1420	0,04	-0,17	2,55	73519	3873	0,05	-0,11	2,24
768.	32201	2337	0,07	-0,06	2,72	47164	2830	0,06	0,04	3,20
769.	29561	1665	0,06	0,10	2,90	20792	1720	0,08	-0,15	2,87
770.	23192	2174	0,09	-0,18	3,03	21324	1202	0,06	0,07	2,69
771.	44657	2472	0,06	-0,55	3,38	51724	4703	0,09	-0,28	3,33
772.	33536	4300	0,13	-0,22	2,33	50579	2381	0,05	0,00	2,84
773.	34739	1544	0,04	0,39	3,02	20954	2701	0,13	0,11	2,64
774.	23740	1403	0,06	0,06	2,77	15741	1899	0,12	0,20	2,58
775.	27878	2236	0,08	1,22	5,67	51506	2188	0,04	-0,44	2,71
776.	45244	2879	0,06	-0,07	2,21	34137	3037	0,09	0,09	2,74
777.	35491	4044	0,11	-0,07	2,50	38491	1816	0,05	0,32	2,98
778.	26171	2794	0,11	0,00	2,37	22126	6307	0,29	0,33	2,45
779.	30831	3214	0,10	0,66	2,91	54111	1634	0,03	0,14	3,23
780.	16396	4630	0,28	-0,13	2,57	23045	8505	0,37	0,01	2,43
781.	24880	3350	0,13	-0,07	2,39	23185	1507	0,06	0,09	2,97
782.	23248	2244	0,10	-0,48	3,17	28683	1444	0,05	-0,09	3,06
783.	24343	1029	0,04	-0,06	2,78	46985	2058	0,04	-0,02	2,52
784.	38402	2044	0,05	0,10	2,58	48618	4129	0,08	0,35	2,58
785.	31276	1637	0,05	0,42	2,77	59523	3254	0,05	-0,23	3,13
786.	52032	3379	0,06	-0,08	2,54	34880	2496	0,07	0,28	3,01
787.	49554	3817	0,08	-0,42	2,84	35764	1799	0,05	0,34	3,02
788.	15434	1702	0,11	-0,17	2,72	18058	1220	0,07	-0,09	2,90
789.	20295	3819	0,19	0,31	2,38	30959	1613	0,05	0,47	2,84
790.	10223	1812	0,18	0,24	2,59	7830	2696	0,34	0,44	2,57
791.	26292	1362	0,05	-0,02	2,57	38997	2846	0,07	-0,41	3,18
792.	38721	2943	0,08	-0,31	2,39	59944	3642	0,06	-0,06	3,18
793.	55698	2007	0,04	-0,35	3,51	32615	3639	0,11	0,40	2,25
794.	40020	2796	0,07	-1,05	5,35	43843	3268	0,07	-0,07	2,36
795.	18401	2769	0,15	-0,22	2,92	21364	5068	0,24	-0,35	2,88
796.	43443	3352	0,08	-0,01	2,71	40196	3094	0,08	0,08	3,38
797.	31517	1614	0,05	0,16	3,94	22999	1949	0,08	-0,55	3,56
798.	31679	4086	0,13	-0,17	2,54	33591	1557	0,05	0,43	3,47
799.	22335	6102	0,27	0,15	2,62	22136	3015	0,14	-0,16	2,65
800.	26508	1906	0,07	-0,10	3,16	26425	1923	0,07	-0,28	2,41
801.	31027	1759	0,06	-0,10	2,70	48438	4864	0,10	0,08	3,12
802.	25885	2464	0,10	-0,09	3,01	26837	5932	0,22	0,34	3,19
803.	47845	2579	0,05	-0,11	2,41	56877	4637	0,08	0,38	3,23
804.	33630	4339	0,13	0,52	3,47	53053	2338	0,04	0,36	3,40
805.	48731	4259	0,09	-0,23	2,60	31256	1569	0,05	0,28	2,95
806.	22006	4910	0,22	0,25	2,46	29014	1814	0,06	-0,01	2,68
807.	41232	3208	0,08	-0,14	2,80	49392	2701	0,05	0,26	2,73
808.	27209	4214	0,15	0,17	3,12	37956	1565	0,04	-0,07	3,17
809.	26274	2492	0,09	-0,56	3,37	42027	5519	0,13	-0,39	2,77
810.	27510	2327	0,08	0,52	3,32	22286	5474	0,25	0,32	3,08
811.	33989	1722	0,05	-0,43	3,44	60854	5076	0,08	-0,27	3,24
812.	28581	2299	0,08	-0,37	3,13	39943	2002	0,05	-0,28	3,25
813.	21519	1249	0,06	0,28	3,53	36766	1918	0,05	0,35	3,01
814.	22938	3118	0,14	-0,75	3,86	29336	6627	0,23	-0,45	3,56
815.	30622	2745	0,09	0,01	2,97	34014	1899	0,06	-0,48	3,46
816.	14192	3314	0,23	0,04	2,53	17692	5588	0,32	0,16	2,68
817.	30556	2088	0,07	-0,14	2,72	53277	6267	0,12	-0,07	2,45
818.	22204	3466	0,16	-0,33	2,55	31230	1601	0,05	0,06	2,36
819.	39609	2801	0,07	0,35	2,95	30739	1823	0,06	-0,33	3,70
820.	46394	2203	0,05	0,43	4,13	52032	6235	0,12	-0,15	2,85
821.	24022	4653	0,19	0,99	3,70	39600	1642	0,04	0,71	3,73
822.	36085	1419	0,04	0,35	2,98	66042	4706	0,07	-0,58	2,74
823.	17651	1090	0,06	-0,28	3,94	16131	2110	0,13	-0,29	3,03
824.	13968	1900	0,14	-0,01	2,46	11274	3032	0,27	0,08	2,44
825.	36493	1727	0,05	0,32	2,65	26481	4237	0,16	-0,06	2,58
826.	50069	2271	0,05	-0,65	4,53	38804	4180	0,11	-0,37	2,91
827.	31228	5133	0,16	0,15	2,58	45727	1970	0,04	0,09	3,06
828.	31593	2159	0,07	0,98	3,96	20723	4895	0,24	0,58	2,55
829.	41582	3697	0,09	0,24	2,90	36891	1663	0,05	0,36	2,92
830.	17733	1561	0,09	0,02	3,11	19014	1401	0,07	-0,11	2,60
831.	47577	4048	0,09	0,31	2,51	65846	4226	0,06	-0,03	2,22
832.	24265	1507	0,06	0,47	3,13	43265	2173	0,05	0,35	3,08
833.	17544	3559	0,20	0,10	2,30	18063	6970	0,39	0,36	2,58
834.	35941	3450	0,10	-0,13	2,95	41019	2244	0,05	-0,23	2,65
835.	31359	2553	0,08	0,21	2,64	28999	1666	0,06	0,27	4,31
836.	18391	2147	0,12	0,22	3,25	15768	4123	0,26	0,15	2,87
837.	27222	1737	0,06	0,32	3,09	19929	3567	0,18	0,03	2,56
838.	46913	2742	0,06	-0,11	2,32	31019	2215	0,07	0,22	2,44
839.	16776	2947	0,18	0,70	3,28	26220	1334	0,05	0,34	2,52

inst.	Distribution of CVP total effort ( $\lambda=200$ )					Distribution of CIBI total effort ( $\lambda=200$ )				
	n.	$\mu$	$\sigma$	CV	skew	kurt	$\mu$	$\sigma$	CV	skew
840.	36701	1408	0,04	-0,17	2,60	58659	4988	0,09	-0,25	2,49
841.	23990	4464	0,19	-0,17	2,91	24784	1879	0,08	-0,31	2,96
842.	30107	4977	0,17	-0,03	2,96	26061	2045	0,08	-0,15	2,85
843.	42604	2361	0,06	0,26	2,74	32483	2698	0,08	-0,17	3,45
844.	39083	1790	0,05	0,91	5,26	63278	5330	0,08	-0,48	3,26
845.	22295	2631	0,12	0,48	2,47	38342	1627	0,04	-0,30	4,15
846.	19816	1272	0,06	-0,04	3,25	15266	2187	0,14	-0,01	3,16
847.	36951	2010	0,05	0,48	5,47	59290	4062	0,07	-0,11	2,62
848.	25591	1838	0,07	-0,44	4,05	27552	1419	0,05	-0,09	2,85
849.	37362	1728	0,05	-0,19	2,89	32008	5247	0,16	-0,05	3,03
850.	20349	3580	0,18	0,41	2,38	31414	1470	0,05	0,19	2,66
851.	26120	1348	0,05	0,60	3,39	49594	2222	0,04	0,03	3,27
852.	35530	1686	0,05	-0,10	2,72	41536	4546	0,11	-0,07	2,78
853.	24017	2637	0,11	-0,31	2,51	34262	1448	0,04	-0,37	3,78
854.	22487	2673	0,12	0,17	2,38	27079	1485	0,05	0,18	3,01
855.	15294	1267	0,08	0,00	2,66	16962	2149	0,13	-0,21	2,12
856.	39212	4637	0,12	0,00	2,67	30448	1678	0,06	0,02	2,85
857.	47474	2278	0,05	0,19	2,75	51784	5999	0,12	-0,34	2,98
858.	35634	1936	0,05	-0,36	2,57	35372	2863	0,08	-0,26	2,12
859.	32808	1466	0,04	0,33	2,45	20653	3602	0,17	0,44	2,37
860.	32432	2225	0,07	-0,35	2,94	55796	5739	0,10	-0,13	2,28
861.	21307	1668	0,08	-0,54	3,38	18008	3325	0,18	-0,29	2,99
862.	24982	1498	0,06	0,64	3,28	22120	3471	0,16	-0,25	2,70
863.	26559	3355	0,13	0,07	2,47	23918	7632	0,32	0,23	2,48
864.	17733	3770	0,21	0,52	2,64	27011	1652	0,06	0,35	2,83
865.	29623	1805	0,06	0,35	2,52	18951	4296	0,23	0,77	3,07
866.	35785	1696	0,05	-0,01	2,75	58214	4275	0,07	-0,20	2,60
867.	27939	1418	0,05	0,26	2,90	35471	3067	0,09	-0,76	3,66
868.	33556	1899	0,06	-0,17	2,80	30195	2879	0,10	-0,36	3,90
869.	27349	2015	0,07	0,18	2,92	25110	4741	0,19	-0,09	2,61
870.	46387	2305	0,05	-0,22	3,69	42629	4670	0,11	-0,23	2,49
871.	41704	2158	0,05	-0,28	3,14	41889	6504	0,16	-0,02	2,96
872.	46316	3245	0,07	-0,08	3,46	62592	3724	0,06	-0,10	2,70
873.	53664	2063	0,04	0,24	2,80	30843	3458	0,11	0,43	2,28
874.	28016	1321	0,05	0,00	3,28	46756	3748	0,08	-0,33	3,52
875.	12704	1210	0,10	-0,21	2,55	12031	1738	0,14	-0,48	3,37
876.	31324	4020	0,13	0,07	3,17	24385	1848	0,08	-0,43	3,05
877.	44347	3743	0,08	0,52	3,07	70409	3427	0,05	-0,29	2,80
878.	15104	2254	0,15	0,63	2,69	25471	1049	0,04	0,19	3,00
879.	37531	1608	0,04	0,01	2,49	42676	4118	0,10	-0,01	2,88
880.	32538	1822	0,06	-0,02	3,46	47711	4715	0,10	-0,12	3,20
881.	33137	2980	0,09	0,68	2,93	60413	2212	0,04	-0,26	2,70
882.	54121	2192	0,04	-0,18	2,99	30535	3014	0,10	0,48	2,52
883.	42318	2148	0,05	0,11	3,80	66931	4848	0,07	-0,04	2,99
884.	39418	3092	0,08	-0,18	3,02	41921	3038	0,07	0,05	4,14
885.	34112	1879	0,06	0,02	3,36	56869	3936	0,07	-0,28	2,20
886.	25479	4714	0,19	-0,36	2,76	20178	2313	0,11	-0,44	2,97
887.	47889	3948	0,08	-0,34	3,13	33355	1669	0,05	-0,13	2,94
888.	44265	2507	0,06	-0,12	3,32	63463	4477	0,07	-0,66	2,89
889.	29988	3212	0,11	0,52	2,96	22929	7136	0,31	0,58	2,48
890.	14424	2250	0,16	-0,49	3,29	14982	3721	0,25	-0,40	2,98
891.	35178	2291	0,07	-0,35	3,31	28884	1616	0,06	-0,44	2,83
892.	35251	1820	0,05	0,17	3,20	62641	5424	0,09	-0,03	2,67
893.	38691	2947	0,08	-0,17	2,84	39769	2066	0,05	-0,17	2,67
894.	49180	3465	0,07	-0,14	2,29	59690	4324	0,07	-0,47	3,26
895.	34159	1812	0,05	-0,69	3,74	36846	3480	0,09	-0,13	2,48
896.	25690	1082	0,04	0,14	2,54	48190	2824	0,06	-0,47	2,80
897.	31017	1354	0,04	-0,37	3,08	57484	3535	0,06	-0,17	3,24
898.	34468	4479	0,13	0,31	2,39	57165	2215	0,04	-0,30	2,92
899.	38089	1899	0,05	-0,64	3,50	46332	5395	0,12	-0,05	2,93
900.	53417	1968	0,04	0,09	2,72	30243	3211	0,11	0,71	2,95
901.	18041	1633	0,09	-0,44	3,54	26158	3534	0,14	-0,04	3,06
902.	53181	2782	0,05	-0,36	3,02	33254	2645	0,08	-0,15	2,52
903.	22304	1187	0,05	0,07	2,36	14134	2159	0,15	0,22	2,49
904.	19255	1312	0,07	0,09	2,63	32216	1868	0,06	-0,63	4,07
905.	47358	1976	0,04	0,05	2,67	30110	3753	0,12	0,51	3,46
906.	32740	1471	0,04	0,03	2,56	44156	4376	0,10	0,00	2,32
907.	40415	1787	0,04	-0,14	2,91	51451	4587	0,09	-0,11	2,62
908.	44473	2125	0,05	-0,07	3,05	65960	5045	0,08	-0,09	2,64
909.	28934	1355	0,05	0,14	3,59	50220	3883	0,08	0,19	2,49
910.	28421	2291	0,08	-0,33	2,93	31124	5574	0,18	-0,11	2,98
911.	30775	2248	0,07	0,16	3,31	39692	2462	0,06	-0,38	3,15
912.	37065	4244	0,11	0,08	2,22	59525	2879	0,05	-0,48	3,98
913.	23711	3364	0,14	0,08	2,26	32887	1422	0,04	-0,10	2,70
914.	30030	1654	0,06	0,18	2,98	28038	2570	0,09	0,24	2,94
915.	52075	2811	0,05	0,44	2,43	49109	4377	0,09	-0,40	2,96
916.	19869	2303	0,12	0,06	2,89	30666	1389	0,05	0,50	2,99
917.	17970	2451	0,14	0,45	2,30	13246	4602	0,35	0,59	2,57
918.	35469	2462	0,07	0,02	2,97	59986	3264	0,05	0,03	2,54
919.	27382	2388	0,09	-0,44	3,78	35070	5560	0,16	-0,62	4,47
920.	18728	1293	0,07	-0,18	4,64	15644	1610	0,10	-0,77	3,77

inst. n.	Distribution of CVP total effort ( $\lambda=200$ )					Distribution of CIBI total effort ( $\lambda=200$ )				
	$\mu$	$\sigma$	CV	skew	kurt	$\mu$	$\sigma$	CV	skew	kurt
921.	25374	1913	0,08	0,14	3,23	43040	1986	0,05	0,14	3,41
922.	27213	2452	0,09	-0,14	3,06	37524	2347	0,06	-0,12	3,23
923.	46712	2588	0,06	-0,16	4,47	62419	4072	0,07	-0,70	3,77
924.	28118	1570	0,06	-0,13	2,89	27941	3481	0,12	-0,30	2,84
925.	18631	1625	0,09	-0,24	3,34	25338	1504	0,06	-0,29	3,17
926.	18094	4769	0,26	0,80	3,44	26125	2043	0,08	0,34	2,55
927.	25795	1850	0,07	-0,22	2,85	34568	1821	0,05	0,05	2,59
928.	15319	3017	0,20	-0,03	2,52	20456	1535	0,08	-0,11	2,57
929.	50468	2771	0,05	-0,12	2,86	37100	3236	0,09	-0,03	3,40
930.	33818	1805	0,05	-0,21	2,39	58578	2957	0,05	0,01	2,72
931.	40731	3531	0,09	0,28	2,62	65772	3257	0,05	-0,64	3,83
932.	41676	1915	0,05	-0,36	2,77	40915	4032	0,10	0,11	2,70
933.	23236	4122	0,18	0,48	3,23	32646	1538	0,05	0,01	2,73
934.	58136	2025	0,03	0,30	3,01	37038	5669	0,15	0,27	2,21
935.	33052	1812	0,05	0,04	2,53	33715	2658	0,08	-0,26	2,49
936.	39454	3822	0,10	-0,08	2,39	39385	2100	0,05	0,46	3,96
937.	34727	1814	0,05	0,50	3,46	21760	4544	0,21	0,54	2,58
938.	47322	2029	0,04	0,39	3,40	30188	4105	0,14	0,15	2,13
939.	11148	2481	0,22	0,09	2,13	9761	3925	0,40	0,19	1,96
940.	33313	2654	0,08	-0,18	3,40	25974	1439	0,06	0,30	3,83
941.	26937	2978	0,11	0,23	2,73	26294	6577	0,25	0,16	2,25
942.	21236	5153	0,24	-0,08	2,51	20620	2493	0,12	-0,13	2,74
943.	34942	2903	0,08	0,12	2,39	50703	2634	0,05	-0,07	2,80
944.	23114	1098	0,05	0,20	2,37	32960	2469	0,07	-0,50	3,43
945.	35022	3302	0,09	0,07	3,03	54539	2510	0,05	-0,54	3,74
946.	26025	1225	0,05	0,40	2,75	46607	2930	0,06	0,10	2,24
947.	18049	4382	0,24	-0,25	2,28	22643	8222	0,36	-0,17	2,18
948.	25890	1195	0,05	0,22	2,43	18913	2189	0,12	-0,05	2,65
949.	40297	1687	0,04	0,44	3,65	62856	5615	0,09	-0,44	3,00
950.	39468	1821	0,05	0,12	2,91	27905	5129	0,18	0,07	2,15
951.	55630	2209	0,04	0,19	2,82	35942	4609	0,13	0,19	2,22
952.	21522	2265	0,11	-0,20	2,42	19133	4753	0,25	-0,08	2,25
953.	16889	1720	0,10	-0,23	3,71	15087	2971	0,20	-0,25	2,75
954.	18925	1380	0,07	-0,20	2,47	13243	2625	0,20	-0,01	2,46
955.	15607	1895	0,12	0,40	2,58	11199	3293	0,29	0,51	2,49
956.	50730	2767	0,05	-0,22	3,01	63624	5303	0,08	-0,20	3,46
957.	15608	3858	0,25	0,51	2,52	14720	7239	0,49	0,51	2,38
958.	12868	2528	0,20	0,48	2,27	10124	4251	0,42	0,69	2,64
959.	21557	3682	0,17	-0,18	2,45	25882	1588	0,06	-0,21	2,88
960.	23808	1376	0,06	0,00	2,66	36978	2830	0,08	-0,28	2,90
961.	13935	3226	0,23	0,17	2,09	18873	1821	0,10	0,08	2,17
962.	28842	4662	0,16	0,10	2,39	41240	1887	0,05	0,31	2,58
963.	28139	2490	0,09	-0,40	3,66	20795	1334	0,06	0,17	2,52
964.	22349	1346	0,06	0,17	2,55	19739	2747	0,14	-0,05	2,28
965.	43514	3646	0,08	-0,42	3,18	27501	1339	0,05	-0,04	3,42
966.	35449	2540	0,07	0,54	4,20	48691	3355	0,07	0,05	3,35
967.	23618	1970	0,08	0,32	2,09	42217	1892	0,04	-0,45	3,49
968.	26228	1877	0,07	0,18	2,64	21147	4397	0,21	0,18	2,72
969.	27957	1965	0,07	-0,31	2,84	24713	1735	0,07	-0,15	2,74
970.	19702	3699	0,19	-0,17	2,54	21031	1826	0,09	-0,45	2,68
971.	28733	5121	0,18	0,01	2,51	36168	1801	0,05	0,13	2,71
972.	32370	1596	0,05	-0,36	2,44	42657	4143	0,10	-0,23	3,21
973.	40424	2640	0,07	-0,18	2,94	37457	2576	0,07	0,07	3,30
974.	27688	1605	0,06	-0,14	3,13	31168	3078	0,10	0,02	3,21
975.	18825	1201	0,06	-0,31	3,24	16970	1756	0,10	-0,30	2,81
976.	37171	2290	0,06	1,03	4,46	23472	5065	0,22	0,63	2,79
977.	27809	2250	0,08	0,73	2,85	18320	5127	0,28	0,88	3,22
978.	37523	2483	0,07	-0,39	3,58	30657	2280	0,07	-0,14	3,70
979.	24022	5669	0,24	0,36	3,30	24158	2741	0,11	0,37	3,35
980.	37049	1859	0,05	0,20	2,95	33503	5362	0,16	0,20	2,51
981.	30323	2143	0,07	0,07	2,67	21763	5218	0,24	0,34	2,64
982.	31313	1442	0,05	-0,02	2,54	58025	4349	0,07	0,11	2,48
983.	16064	2380	0,15	-0,32	3,36	20300	4216	0,21	-0,46	3,34
984.	18754	1473	0,08	0,06	2,61	14732	2710	0,18	0,02	2,64
985.	10806	3018	0,28	-0,05	2,82	12936	1901	0,15	0,26	3,47
986.	42723	3918	0,09	-0,13	2,34	30991	1454	0,05	-0,33	4,05
987.	37441	3070	0,08	0,48	2,95	61551	2895	0,05	-0,35	3,10
988.	31216	1841	0,06	0,03	3,14	52573	3641	0,07	-0,55	3,38
989.	24478	1491	0,06	0,31	2,90	44738	2101	0,05	-0,42	3,06
990.	43847	3282	0,07	-0,11	2,80	61090	4099	0,07	0,10	3,35
991.	23630	2091	0,09	-0,31	4,44	17778	1145	0,06	0,02	2,98
992.	36270	2010	0,06	-0,61	3,62	38124	6017	0,16	-0,21	2,92
993.	31723	1285	0,04	0,31	3,16	61711	3393	0,05	-0,28	3,17
994.	43333	4021	0,09	0,15	2,89	51142	2539	0,05	-0,09	3,43
995.	54510	2260	0,04	0,05	2,39	36027	4613	0,13	-0,06	2,31
996.	46740	2366	0,05	0,18	2,98	61996	4563	0,07	-0,13	2,73
997.	30407	2901	0,10	-0,02	2,51	50067	6958	0,14	0,27	3,00
998.	45818	2295	0,05	-0,30	3,31	61136	5160	0,08	0,15	2,54
999.	18382	1433	0,08	0,46	2,63	33888	1409	0,04	0,17	2,56
1000.	24691	3583	0,15	0,09	2,65	34303	1364	0,04	0,90	5,02



### Frequency distributions of the intermediate effort assessments ( $Y_i$ ) per sample instance

In the flowing table, each row of the table represents the corresponding sample instance in Appendix B. In each row of the table (for each sample instance  $i$ ), several statistical parameters of the frequency distribution of the intermediate effort assessments  $Y_i$  (per design alternative) of several applied scenarios ( $\lambda=200$ ) for a single (indicative) simulation are presented. Parameter  $\mu$  represents the mean value,  $\sigma$  is the standard deviation,  $CV=\sigma/\mu$  is the coefficient of variation, ‘skew’ is the skewness, and ‘kurt’ is the kurtosis of each frequency distribution. Since the intermediate effort outcome ( $Y_i$ ) is a heavily stochastic variable expressing the required effort per applied scenario, all the parameters of its frequency distribution variate significantly among different sample instances. CV lies between 0.43 and 2.88, skewness between 0.16 and 11.43, and kurtosis between 2.31 and 149.62, as discussed in subsection 6.4.9.4.

Statistical Parameters of Intermediate Effort of Repeated Applied Scenarios (of a Single Simulation) per Sample Instance

inst. n.	Distribution of CVP intermediate effort					Distribution of CIBI intermediate effort				
	$\mu$	$\sigma$	CV	skew	kurt	$\mu$	$\sigma$	CV	skew	kurt
1.	153,69	94,57	0,62	1,24	5,01	177,59	210,43	1,18	2,00	6,63
2.	112,58	115,54	1,03	2,55	9,79	138,48	68,24	0,49	0,67	2,88
3.	208,13	141,72	0,68	2,02	10,97	309,35	195,72	0,63	1,09	3,81
4.	117,76	135,20	1,15	2,91	13,89	139,15	86,68	0,62	1,84	8,11
5.	183,75	106,94	0,58	1,13	4,85	102,54	60,60	0,59	1,03	4,38
6.	59,03	54,34	0,92	3,04	19,01	69,08	116,24	1,68	3,42	19,32
7.	46,32	40,24	0,87	3,38	19,68	75,73	46,24	0,61	1,10	5,33
8.	161,29	109,67	0,68	1,92	8,29	260,51	127,16	0,49	0,93	3,86
9.	106,92	56,40	0,53	1,07	5,46	208,65	103,53	0,50	1,55	6,69
10.	84,64	74,97	0,89	2,79	17,38	116,01	156,30	1,35	3,07	18,07
11.	140,12	85,41	0,61	2,88	20,70	122,35	119,05	0,97	2,06	7,61
12.	161,19	195,08	1,21	4,87	43,01	121,73	92,85	0,76	4,76	43,70
13.	130,25	72,68	0,56	1,22	6,98	257,04	131,88	0,51	1,68	9,03
14.	195,20	112,98	0,58	2,08	11,44	357,50	160,68	0,45	0,81	3,48
15.	103,17	79,12	0,77	2,04	7,99	154,52	81,81	0,53	0,99	4,40
16.	172,40	112,26	0,65	1,81	7,70	288,63	177,97	0,62	2,11	10,34
17.	48,30	55,15	1,14	2,79	11,37	59,17	35,95	0,61	1,09	4,65
18.	258,87	135,95	0,53	1,19	4,99	152,94	93,84	0,61	1,81	8,09
19.	171,48	165,24	0,96	1,87	6,63	178,12	109,96	0,62	1,35	6,03
20.	269,02	143,04	0,53	0,87	4,16	199,94	127,48	0,64	1,52	5,85
21.	168,06	134,14	0,80	1,95	8,60	158,07	103,02	0,65	1,56	7,54
22.	48,59	71,34	1,47	3,42	14,68	58,18	149,67	2,57	3,61	15,37
23.	133,45	78,70	0,59	1,36	8,75	258,66	122,96	0,48	0,72	3,21
24.	159,40	135,75	0,85	1,99	10,12	122,75	73,55	0,60	1,51	7,55
25.	173,37	112,54	0,65	2,08	11,84	214,56	202,00	0,94	2,00	9,38
26.	181,19	94,61	0,52	1,04	4,13	207,82	176,06	0,85	1,45	5,52
27.	126,51	85,54	0,68	2,25	11,76	207,00	107,47	0,52	0,82	3,70
28.	137,28	107,02	0,78	2,99	18,00	205,43	139,49	0,68	0,88	4,18
29.	50,35	36,96	0,73	3,61	20,81	31,78	62,96	1,98	7,58	61,75
30.	110,77	64,82	0,59	1,30	6,89	205,68	117,58	0,57	1,23	6,40
31.	69,52	39,59	0,57	1,79	8,49	48,01	65,67	1,37	4,64	29,16
32.	191,03	110,15	0,58	0,95	5,02	234,69	189,95	0,81	1,37	5,00
33.	137,44	157,67	1,15	7,38	81,79	82,95	81,10	0,98	6,53	69,03
34.	153,07	87,43	0,57	0,97	4,77	184,70	181,37	0,98	1,75	6,70
35.	128,59	89,18	0,69	1,05	4,53	221,27	197,81	0,89	0,98	3,84
36.	219,47	108,83	0,50	0,49	3,34	245,95	186,60	0,76	1,28	3,99
37.	161,47	105,74	0,65	1,67	7,45	199,33	243,43	1,22	1,98	7,32
38.	214,18	138,73	0,65	1,34	5,34	267,07	193,47	0,72	1,13	4,83
39.	212,45	216,70	1,02	1,93	8,03	169,16	104,37	0,62	1,14	5,83
40.	225,20	141,72	0,63	1,62	7,40	212,13	143,92	0,68	1,07	4,05
41.	66,96	67,67	1,01	3,72	19,80	106,82	56,89	0,53	1,63	7,18
42.	211,49	149,42	0,71	2,68	15,49	242,79	216,10	0,89	5,21	48,94
43.	248,17	149,21	0,60	1,47	6,56	211,74	183,44	0,87	2,30	10,38
44.	92,35	55,50	0,60	1,14	5,02	99,88	106,41	1,07	1,75	6,07
45.	165,18	92,71	0,56	1,32	6,97	296,19	163,93	0,55	0,77	3,49
46.	247,98	137,30	0,55	1,79	8,36	141,96	106,91	0,75	2,60	13,11
47.	238,15	143,28	0,60	2,24	10,83	129,57	91,29	0,70	1,75	7,20
48.	222,81	185,14	0,83	3,12	24,38	133,89	89,72	0,67	3,05	24,63
49.	167,32	105,00	0,63	1,31	5,59	124,97	92,65	0,74	2,18	9,69
50.	110,80	109,05	0,98	2,18	8,92	112,33	60,84	0,54	1,22	6,55
51.	154,15	188,44	1,22	2,25	9,52	126,35	86,42	0,68	1,78	8,47
52.	225,62	135,67	0,60	1,37	5,84	272,03	195,62	0,72	1,27	5,05
53.	251,50	122,38	0,49	0,65	3,60	169,21	137,81	0,81	2,46	10,36
54.	93,18	155,79	1,67	2,72	10,06	91,68	69,95	0,76	2,09	8,23
55.	25,47	36,08	1,42	7,19	62,07	41,37	24,51	0,59	2,04	10,66
56.	243,13	154,08	0,63	1,27	5,57	258,54	198,83	0,77	1,31	4,40
57.	189,65	163,35	0,86	1,83	7,49	166,54	103,16	0,62	1,15	4,52

inst. n.	Distribution of CVP intermediate effort					Distribution of CIBI intermediate effort				
	$\mu$	$\sigma$	CV	skew	kurt	$\mu$	$\sigma$	CV	skew	kurt
58.	249,03	206,57	0,83	1,52	6,28	171,54	100,64	0,59	1,12	5,19
59.	167,86	103,03	0,61	0,83	4,69	315,34	203,73	0,65	1,00	4,96
60.	250,18	172,23	0,69	2,20	11,94	155,88	91,38	0,59	1,76	8,32
61.	152,91	94,16	0,62	0,76	4,48	293,79	188,58	0,64	0,88	4,55
62.	157,18	269,86	1,72	8,32	92,35	152,28	136,33	0,90	6,85	72,97
63.	169,75	172,36	1,02	2,80	13,36	181,96	109,07	0,60	2,45	15,80
64.	206,48	124,82	0,60	0,69	3,29	157,65	125,56	0,80	3,87	26,07
65.	121,54	139,31	1,15	2,44	10,24	119,90	71,57	0,60	1,51	6,43
66.	218,91	129,56	0,59	1,18	4,43	224,12	168,12	0,75	1,79	7,79
67.	99,50	77,87	0,78	1,90	7,73	125,79	77,56	0,62	0,94	4,00
68.	238,13	151,11	0,63	0,75	4,08	150,47	107,24	0,71	3,68	29,14
69.	181,08	141,37	0,78	1,40	4,90	210,28	156,41	0,74	2,59	14,40
70.	164,87	107,69	0,65	2,66	16,06	116,25	130,43	1,12	4,59	33,52
71.	159,83	93,12	0,58	1,74	8,02	125,69	128,72	1,02	2,68	11,94
72.	198,27	100,33	0,51	0,67	3,04	144,48	151,04	1,05	2,99	13,09
73.	63,88	50,63	0,79	2,97	14,90	54,57	106,53	1,95	4,18	21,05
74.	92,77	69,85	0,75	1,51	5,74	116,16	157,45	1,36	1,90	6,12
75.	123,54	96,17	0,78	2,51	13,26	147,11	198,69	1,35	2,93	14,85
76.	189,36	105,47	0,56	2,08	9,60	101,45	63,25	0,62	1,80	7,50
77.	74,22	104,37	1,41	4,31	24,15	103,13	53,56	0,52	2,01	9,80
78.	199,03	142,97	0,72	1,19	4,38	251,17	173,32	0,69	2,02	10,07
79.	265,27	163,68	0,62	1,14	4,80	181,00	145,73	0,81	2,32	10,92
80.	202,63	127,59	0,63	1,15	4,57	155,91	103,07	0,66	1,73	6,98
81.	77,47	51,38	0,66	2,02	10,15	128,07	63,43	0,50	0,54	3,26
82.	239,26	153,29	0,64	1,02	4,21	176,84	116,14	0,66	2,11	10,44
83.	184,78	109,60	0,59	0,78	4,60	299,84	223,58	0,75	1,14	5,23
84.	56,56	54,91	0,97	2,98	13,84	90,04	53,80	0,60	1,50	7,28
85.	200,89	114,43	0,57	1,20	5,55	120,27	75,45	0,63	1,68	7,13
86.	131,96	74,86	0,57	1,54	7,20	114,57	137,19	1,20	3,36	16,47
87.	111,44	72,02	0,65	1,40	5,83	94,68	78,36	0,83	2,06	9,05
88.	189,14	123,87	0,65	2,39	13,35	330,43	180,94	0,55	1,52	9,75
89.	146,18	75,79	0,52	0,33	3,17	287,22	153,16	0,53	0,36	3,05
90.	183,41	110,87	0,60	1,01	4,33	297,17	186,36	0,63	1,00	3,95
91.	143,09	77,87	0,54	0,84	4,86	292,16	146,89	0,50	1,14	5,28
92.	169,00	113,11	0,67	1,05	3,91	164,79	110,26	0,67	1,09	4,35
93.	158,92	94,36	0,59	1,57	7,05	258,57	178,64	0,69	1,29	5,65
94.	81,45	45,92	0,56	1,84	10,40	160,56	87,84	0,55	2,29	12,03
95.	115,71	66,97	0,58	0,96	4,34	164,85	122,83	0,75	1,15	4,61
96.	134,80	86,71	0,64	1,36	5,23	164,35	185,01	1,13	2,00	7,01
97.	215,55	111,22	0,52	1,32	5,42	115,22	57,69	0,50	1,05	4,66
98.	176,98	95,28	0,54	0,74	3,46	246,93	202,81	0,82	0,99	3,24
99.	168,04	101,07	0,60	1,17	4,75	189,47	212,00	1,12	1,76	5,95
100.	221,29	179,78	0,81	2,25	9,89	240,54	161,49	0,67	1,28	5,27
101.	178,08	100,86	0,57	1,25	5,82	203,74	217,03	1,07	1,99	7,59
102.	228,22	163,89	0,72	1,38	4,70	232,66	139,45	0,60	1,26	6,16
103.	84,97	49,71	0,59	1,02	4,14	65,90	65,24	0,99	3,30	18,73
104.	174,90	109,46	0,63	1,16	4,99	155,54	138,98	0,89	4,59	34,64
105.	109,64	89,48	0,82	1,21	4,30	165,22	197,93	1,20	1,38	4,19
106.	175,30	104,96	0,60	1,27	6,35	328,45	193,25	0,59	1,85	8,64
107.	60,54	66,76	1,10	4,22	23,68	93,27	47,41	0,51	1,02	4,46
108.	165,55	82,58	0,50	0,78	4,31	324,68	166,85	0,51	0,83	4,15
109.	146,28	101,05	0,69	1,12	4,09	115,09	78,32	0,68	2,40	16,04
110.	114,00	117,06	1,03	2,02	7,66	111,01	74,24	0,67	2,24	11,45
111.	271,02	160,11	0,59	1,41	6,22	185,06	171,83	0,93	4,37	33,91
112.	120,76	98,41	0,81	1,66	6,04	134,06	81,46	0,61	1,05	4,75
113.	283,53	162,59	0,57	1,38	6,37	146,93	86,38	0,59	1,74	8,27
114.	171,25	105,10	0,61	1,57	6,62	261,54	155,81	0,60	0,72	3,22
115.	246,74	144,44	0,59	0,97	4,33	245,19	223,51	0,91	2,26	9,70
116.	145,11	78,40	0,54	1,27	6,21	119,08	149,08	1,25	3,39	15,78
117.	206,60	109,14	0,53	1,31	5,90	129,37	120,59	0,93	3,36	17,56
118.	122,36	61,53	0,50	1,54	8,69	60,60	32,72	0,54	1,18	7,50
119.	164,38	92,90	0,57	0,86	3,93	232,04	200,85	0,87	1,23	4,41
120.	102,17	119,53	1,17	1,71	5,70	167,35	252,13	1,51	1,76	5,55
121.	263,32	142,62	0,54	1,01	4,00	226,07	193,47	0,86	1,64	5,41
122.	174,25	121,68	0,70	1,43	5,22	237,70	173,62	0,73	1,99	9,74
123.	142,13	86,53	0,61	1,54	6,63	172,13	117,75	0,68	1,22	4,81
124.	195,45	167,61	0,86	2,26	13,84	137,22	83,72	0,61	1,87	11,31
125.	257,00	140,56	0,55	0,91	3,67	169,55	151,23	0,89	4,37	29,71
126.	161,93	135,99	0,84	3,67	22,63	289,73	174,02	0,60	2,93	22,23
127.	89,01	44,21	0,50	0,87	3,96	53,43	65,46	1,23	4,99	30,66
128.	169,74	127,59	0,75	2,31	13,46	309,29	265,02	0,86	2,27	12,56
129.	122,26	75,43	0,62	1,01	4,62	184,00	138,81	0,75	1,41	5,96
130.	127,74	120,64	0,94	1,77	6,30	124,02	80,66	0,65	1,57	6,74
131.	227,25	141,18	0,62	1,44	6,64	209,80	257,32	1,23	3,32	16,72
132.	127,26	111,78	0,88	2,40	9,71	173,35	88,69	0,51	1,05	4,91
133.	75,40	50,97	0,68	1,53	6,17	108,10	87,21	0,81	1,73	8,30
134.	221,33	106,21	0,48	0,98	3,74	122,34	75,83	0,62	1,79	7,96
135.	234,71	132,74	0,57	0,95	3,94	219,68	189,83	0,86	2,06	8,81
136.	159,50	110,54	0,69	2,44	13,22	209,58	191,94	0,92	2,90	16,69
137.	173,60	100,13	0,58	1,01	4,97	339,85	202,52	0,60	1,01	4,90
138.	203,23	105,67	0,52	1,08	5,50	329,02	217,92	0,66	1,13	5,43

inst. n.	Distribution of CVP intermediate effort					Distribution of CIBI intermediate effort				
	$\mu$	$\sigma$	CV	skew	kurt	$\mu$	$\sigma$	CV	skew	kurt
139.	95,18	57,09	0,60	1,82	11,37	154,79	99,05	0,64	2,60	19,80
140.	135,77	122,93	0,91	2,09	8,05	173,69	125,03	0,72	5,96	60,52
141.	143,70	85,08	0,59	1,69	9,05	115,76	101,01	0,87	2,14	9,47
142.	119,39	79,32	0,66	1,09	4,14	126,11	86,34	0,68	1,59	6,97
143.	176,16	103,23	0,59	1,25	5,06	185,12	200,62	1,08	1,62	4,90
144.	138,96	122,60	0,88	1,47	4,25	144,37	77,82	0,54	1,11	5,57
145.	244,53	145,64	0,60	2,61	20,09	129,11	73,46	0,57	2,41	18,82
146.	198,05	196,67	0,99	4,33	33,93	218,00	142,34	0,65	1,81	8,25
147.	248,57	131,04	0,53	1,33	6,59	163,99	145,34	0,89	2,67	11,79
148.	50,64	45,52	0,90	2,22	8,20	72,27	40,40	0,56	1,22	5,00
149.	168,21	135,76	0,81	1,95	8,55	187,29	101,23	0,54	0,72	3,69
150.	184,08	201,36	1,09	2,32	11,82	129,97	91,06	0,70	2,27	13,14
151.	117,09	73,51	0,63	0,82	3,62	194,21	146,72	0,76	1,18	4,29
152.	170,85	102,75	0,60	1,01	4,56	217,86	158,22	0,73	1,16	4,19
153.	241,32	173,92	0,72	3,13	21,11	202,09	136,54	0,68	1,62	6,59
154.	106,19	67,03	0,63	1,68	7,19	180,50	103,29	0,57	1,39	7,05
155.	145,19	80,20	0,55	0,73	3,60	240,62	157,49	0,65	0,87	3,81
156.	117,84	104,65	0,89	3,41	20,52	89,53	77,50	0,87	2,27	9,79
157.	139,17	73,77	0,53	1,41	6,80	79,64	55,49	0,70	2,02	8,64
158.	164,48	98,69	0,60	1,55	7,03	170,37	145,96	0,86	2,36	12,88
159.	157,27	100,02	0,64	1,57	7,80	185,56	131,48	0,71	1,84	8,47
160.	183,30	89,85	0,49	1,37	7,16	107,18	64,66	0,60	1,70	8,74
161.	95,87	52,17	0,54	1,91	14,54	186,21	87,62	0,47	0,47	3,24
162.	228,62	138,77	0,61	1,21	4,64	350,86	193,47	0,55	1,33	5,31
163.	171,28	140,03	0,82	1,86	6,11	227,99	113,24	0,50	2,07	11,45
164.	81,51	65,30	0,80	1,45	5,33	107,11	138,03	1,29	1,94	6,56
165.	249,80	170,49	0,68	1,89	9,10	296,15	189,32	0,64	1,27	5,25
166.	78,68	43,27	0,55	1,69	8,48	50,54	63,03	1,25	5,45	43,37
167.	136,42	116,58	0,85	3,53	23,46	161,86	89,34	0,55	1,02	4,50
168.	150,97	95,96	0,64	1,61	7,17	236,18	161,34	0,68	1,52	7,11
169.	106,94	66,44	0,62	1,16	5,33	172,51	102,46	0,59	0,68	3,35
170.	105,64	58,10	0,55	1,13	4,34	78,15	98,11	1,26	2,92	11,45
171.	76,10	48,44	0,64	2,24	11,84	54,19	89,25	1,65	4,91	30,78
172.	125,03	78,72	0,63	3,74	30,84	65,87	49,73	0,76	4,19	29,79
173.	259,56	198,49	0,76	4,77	43,58	145,95	106,16	0,73	4,03	32,59
174.	110,74	101,01	0,91	2,82	12,83	160,12	90,28	0,56	1,46	9,26
175.	141,30	77,45	0,55	0,59	3,40	214,19	146,68	0,68	0,97	4,08
176.	165,39	99,71	0,60	0,95	4,33	293,59	196,65	0,67	0,97	3,93
177.	143,94	161,11	1,12	2,94	18,12	119,44	79,68	0,67	2,21	14,03
178.	188,20	101,69	0,54	1,38	6,34	109,93	103,92	0,95	3,30	17,03
179.	210,76	127,76	0,61	1,12	4,94	199,23	144,47	0,73	1,65	6,79
180.	151,49	105,37	0,70	1,80	6,80	189,11	135,06	0,71	2,11	10,51
181.	72,92	65,32	0,90	3,57	20,99	101,36	53,97	0,53	1,02	4,94
182.	227,47	130,89	0,58	0,99	4,57	299,78	194,71	0,65	1,21	5,27
183.	82,13	84,14	1,02	2,28	9,70	82,25	51,29	0,62	1,30	5,24
184.	88,26	57,51	0,65	1,53	7,22	163,88	112,85	0,69	1,60	7,97
185.	181,42	116,42	0,64	2,91	18,99	154,07	145,93	0,95	2,15	8,42
186.	199,78	111,37	0,56	1,16	6,17	323,99	226,31	0,70	0,95	3,92
187.	62,77	40,42	0,64	1,45	5,69	43,35	52,41	1,21	4,46	30,11
188.	268,95	169,50	0,63	1,15	4,30	196,52	133,95	0,68	1,86	9,24
189.	167,00	109,17	0,65	2,49	14,52	203,87	197,83	0,97	3,36	24,98
190.	138,30	92,47	0,67	2,48	13,75	142,28	172,10	1,21	2,50	10,63
191.	126,58	140,82	1,11	2,04	6,62	143,77	80,13	0,56	1,13	4,84
192.	190,28	110,27	0,58	2,22	13,52	146,13	128,60	0,88	2,01	6,99
193.	223,69	124,70	0,56	0,99	4,46	179,87	196,48	1,09	3,20	15,90
194.	195,11	119,49	0,61	1,73	8,29	198,26	247,04	1,25	2,82	13,06
195.	198,56	135,14	0,68	1,05	4,27	153,16	137,06	0,89	4,64	36,43
196.	70,82	58,03	0,82	2,15	9,00	90,37	73,08	0,81	4,18	34,55
197.	135,59	71,03	0,52	0,89	3,49	111,16	136,25	1,23	2,63	9,57
198.	85,20	66,00	0,77	2,06	8,91	77,72	56,53	0,73	1,31	4,41
199.	215,22	131,15	0,61	1,66	8,60	156,34	115,04	0,74	1,97	8,35
200.	104,94	74,81	0,71	1,08	5,33	170,52	161,27	0,95	1,21	4,98
201.	258,06	148,99	0,58	1,14	5,25	265,92	240,45	0,90	2,42	11,50
202.	197,95	127,77	0,65	2,64	14,77	224,47	159,52	0,71	1,32	5,24
203.	165,75	131,28	0,79	1,64	5,87	212,69	126,45	0,59	1,17	4,53
204.	231,71	114,34	0,49	0,87	4,65	146,87	119,83	0,82	3,83	24,61
205.	206,69	122,89	0,59	1,16	5,26	251,86	214,52	0,85	1,43	5,22
206.	223,04	160,30	0,72	3,90	29,86	114,01	81,53	0,72	3,65	27,72
207.	152,62	92,04	0,60	0,25	2,59	271,31	198,82	0,73	0,37	2,35
208.	140,99	101,35	0,72	1,53	5,59	154,02	95,86	0,62	0,86	3,58
209.	133,64	74,44	0,56	0,93	4,09	125,93	106,91	0,85	2,01	7,43
210.	123,60	73,33	0,59	1,56	6,72	217,93	128,20	0,59	1,54	7,27
211.	198,60	149,71	0,75	2,69	15,62	310,86	245,82	0,79	4,25	34,06
212.	258,95	131,93	0,51	0,76	3,54	151,41	109,50	0,72	2,35	12,98
213.	92,20	110,65	1,20	3,21	15,66	114,57	61,50	0,54	1,28	5,52
214.	150,81	94,74	0,63	1,29	5,91	247,98	150,40	0,61	0,93	3,84
215.	149,05	99,07	0,66	1,47	6,39	167,97	165,18	0,98	1,88	7,27
216.	141,87	147,82	1,04	2,02	7,45	135,37	79,54	0,59	0,97	4,07
217.	44,63	53,99	1,21	5,20	34,86	70,95	41,22	0,58	2,85	19,12
218.	118,98	67,62	0,57	0,69	3,58	210,97	129,16	0,61	0,96	3,95
219.	81,29	81,27	1,00	1,92	6,33	86,21	45,87	0,53	0,59	3,01



inst. n.	Distribution of CVP intermediate effort					Distribution of CIBI intermediate effort				
	$\mu$	$\sigma$	CV	skew	kurt	$\mu$	$\sigma$	CV	skew	kurt
220.	136,72	135,60	0,99	1,75	5,89	127,31	72,29	0,57	1,01	4,40
221.	105,11	118,11	1,12	5,04	44,19	142,42	252,14	1,77	4,75	38,71
222.	179,06	105,78	0,59	1,95	9,04	124,98	101,06	0,81	1,68	5,57
223.	68,79	76,85	1,12	2,70	11,14	86,24	49,79	0,58	1,05	3,74
224.	247,26	132,64	0,54	0,83	3,96	204,77	164,37	0,80	1,84	7,46
225.	88,42	63,85	0,72	1,37	5,92	111,40	105,60	0,95	1,48	5,36
226.	141,22	101,84	0,72	2,05	9,95	169,64	225,67	1,33	2,32	9,78
227.	113,98	87,55	0,77	3,26	21,44	194,22	147,30	0,76	5,13	43,80
228.	166,30	96,74	0,58	1,33	6,61	217,18	138,68	0,64	0,88	3,61
229.	267,09	173,52	0,65	1,47	5,66	296,79	192,21	0,65	0,94	3,88
230.	186,98	109,35	0,58	1,75	7,53	223,82	211,60	0,95	2,11	9,39
231.	225,34	140,87	0,63	1,63	9,12	310,73	253,75	0,82	2,54	15,40
232.	149,03	88,82	0,60	0,84	4,14	246,41	192,30	0,78	1,00	4,01
233.	255,16	183,90	0,72	2,67	16,56	209,41	152,50	0,73	1,50	5,55
234.	95,88	94,22	0,98	2,18	7,94	119,09	205,29	1,72	2,47	8,54
235.	286,42	152,65	0,53	1,33	6,60	147,56	81,48	0,55	1,18	5,90
236.	124,51	139,19	1,12	2,46	9,35	142,73	76,73	0,54	0,62	3,42
237.	126,03	63,31	0,50	1,25	5,16	75,65	89,72	1,19	5,11	33,16
238.	151,68	128,63	0,85	2,91	15,36	192,21	291,25	1,52	2,82	13,06
239.	200,03	144,39	0,72	1,33	6,01	124,57	84,50	0,68	1,85	10,14
240.	97,41	95,48	0,98	2,72	13,68	118,94	76,82	0,65	1,37	6,70
241.	173,99	123,73	0,71	1,47	5,83	200,73	115,26	0,57	1,05	5,11
242.	90,45	51,88	0,57	1,14	6,93	171,81	101,28	0,59	1,16	7,37
243.	178,23	131,32	0,74	3,26	22,63	323,36	215,51	0,67	5,29	48,48
244.	172,32	96,27	0,56	1,17	4,69	158,13	143,60	0,91	2,71	13,90
245.	145,83	77,61	0,53	0,58	3,41	242,03	148,53	0,61	0,70	3,42
246.	199,45	105,79	0,53	0,63	3,36	211,68	175,28	0,83	1,40	4,91
247.	162,36	83,53	0,51	0,85	3,76	144,20	155,02	1,08	2,29	8,79
248.	262,62	146,92	0,56	2,31	12,52	149,94	104,70	0,70	1,99	8,11
249.	195,58	105,84	0,54	0,92	4,49	380,21	209,46	0,55	1,11	4,64
250.	178,12	100,83	0,57	0,92	4,80	291,79	184,07	0,63	0,84	3,48
251.	200,13	118,46	0,59	0,91	3,83	144,79	102,90	0,71	1,40	5,23
252.	106,04	71,69	0,68	2,37	12,30	94,93	131,98	1,39	3,98	23,39
253.	198,10	104,41	0,53	1,31	5,88	120,83	94,87	0,79	2,19	9,46
254.	67,63	83,94	1,24	3,29	17,34	85,85	178,75	2,08	3,38	16,74
255.	245,12	127,78	0,52	1,21	5,81	216,46	208,21	0,96	3,11	17,36
256.	151,47	103,10	0,68	2,71	18,46	288,87	201,48	0,70	3,08	20,77
257.	109,05	69,43	0,64	2,01	10,08	196,64	126,58	0,64	2,17	12,44
258.	146,73	99,43	0,68	1,54	6,86	174,97	217,06	1,24	2,09	7,97
259.	135,99	82,31	0,61	1,44	6,62	138,09	165,87	1,20	2,91	12,49
260.	84,89	96,90	1,14	2,75	11,05	107,84	59,24	0,55	1,02	4,82
261.	87,56	93,44	1,07	2,01	6,63	106,27	78,23	0,74	5,29	48,95
262.	166,28	106,14	0,64	1,16	5,98	277,87	170,41	0,61	1,20	4,94
263.	140,81	111,64	0,79	2,84	13,71	229,72	115,56	0,50	1,24	6,08
264.	73,37	67,54	0,92	2,43	10,64	86,70	141,16	1,63	3,03	13,02
265.	116,18	54,92	0,47	0,57	3,48	67,41	50,50	0,75	4,35	30,67
266.	123,89	64,62	0,52	0,48	2,97	166,84	130,85	0,78	1,03	3,45
267.	17,29	8,19	0,47	0,90	4,10	9,36	4,45	0,48	0,55	3,78
268.	161,74	77,44	0,48	0,98	5,01	296,30	154,58	0,52	0,91	5,00
269.	181,36	111,62	0,62	1,09	5,11	296,55	226,40	0,76	1,15	4,64
270.	172,58	105,26	0,61	2,12	12,03	342,84	206,60	0,60	2,33	12,81
271.	91,78	65,38	0,71	1,15	4,15	77,84	53,04	0,68	1,33	4,78
272.	245,40	146,35	0,60	1,77	10,40	163,21	132,22	0,81	2,47	11,31
273.	181,83	104,63	0,58	1,09	4,91	205,61	225,56	1,10	1,93	6,81
274.	186,80	118,22	0,63	1,19	4,85	268,42	196,90	0,73	1,91	9,17
275.	189,31	163,06	0,86	2,85	19,10	153,67	96,13	0,63	1,61	8,97
276.	152,22	109,30	0,72	1,24	5,54	239,50	225,04	0,94	1,43	6,03
277.	160,92	84,13	0,52	0,68	3,28	120,13	102,75	0,86	3,12	17,51
278.	88,91	80,90	0,91	3,57	19,99	145,03	72,87	0,50	1,51	6,55
279.	115,77	140,30	1,21	2,77	11,63	140,92	74,66	0,53	1,27	6,51
280.	237,03	109,65	0,46	0,66	3,64	137,93	94,24	0,68	2,10	11,12
281.	161,74	136,35	0,84	2,33	9,38	230,89	118,68	0,51	1,49	6,69
282.	78,99	98,03	1,24	2,54	9,91	81,79	50,85	0,62	1,18	4,80
283.	261,88	138,61	0,53	0,93	4,73	137,42	79,40	0,58	1,59	9,61
284.	198,72	107,98	0,54	0,61	3,36	222,41	173,03	0,78	1,18	4,11
285.	226,94	150,93	0,67	2,58	14,42	224,12	162,56	0,73	0,83	3,28
286.	117,51	74,73	0,64	1,53	6,51	112,87	161,18	1,43	2,61	10,00
287.	192,10	91,75	0,48	0,56	4,60	382,01	173,28	0,45	0,94	5,00
288.	165,99	99,27	0,60	1,02	5,11	285,06	188,13	0,66	1,28	5,91
289.	42,01	41,28	0,98	3,98	22,62	73,94	37,85	0,51	1,30	5,46
290.	160,56	95,45	0,59	1,48	6,66	167,41	201,61	1,20	2,49	10,32
291.	86,51	107,83	1,25	2,85	10,85	115,45	68,31	0,59	1,19	4,64
292.	174,56	105,26	0,60	1,18	4,87	266,57	191,19	0,72	1,32	5,84
293.	148,85	75,16	0,50	0,93	4,10	119,38	126,46	1,06	2,50	10,20
294.	209,45	169,26	0,81	4,34	34,60	264,56	171,13	0,65	2,20	12,88
295.	171,67	115,14	0,67	1,83	7,04	256,15	142,11	0,55	1,27	6,57
296.	101,15	80,34	0,79	1,58	7,91	158,59	152,19	0,96	1,62	8,69
297.	278,65	142,04	0,51	0,70	2,88	163,49	95,57	0,58	1,14	5,63
298.	206,94	124,44	0,60	2,52	18,24	242,77	216,16	0,89	1,49	5,09
299.	40,30	19,49	0,48	1,18	6,16	21,33	13,43	0,63	2,32	15,22
300.	88,54	62,30	0,70	1,51	6,32	118,16	138,88	1,18	1,82	6,52

inst. n.	Distribution of CVP intermediate effort					Distribution of CIBI intermediate effort				
	$\mu$	$\sigma$	CV	skew	kurt	$\mu$	$\sigma$	CV	skew	kurt
301.	162,32	105,03	0,65	1,12	4,74	210,62	184,26	0,87	1,43	5,43
302.	201,18	132,84	0,66	1,24	5,62	193,00	124,59	0,65	1,48	6,67
303.	157,26	90,88	0,58	0,72	3,37	230,16	201,59	0,88	1,07	3,55
304.	211,55	116,53	0,55	0,88	3,47	111,96	69,06	0,62	1,24	6,44
305.	175,84	182,93	1,04	1,71	6,41	121,58	81,80	0,67	1,62	6,85
306.	174,07	102,28	0,59	1,33	6,48	196,65	186,85	0,95	1,71	6,22
307.	97,63	86,10	0,88	1,48	4,75	93,50	58,30	0,62	1,09	4,09
308.	61,71	82,38	1,33	3,43	16,43	82,27	48,70	0,59	1,33	4,89
309.	104,03	154,86	1,49	2,65	9,62	105,47	70,57	0,67	1,65	6,52
310.	140,05	81,04	0,58	1,11	4,56	150,24	172,44	1,15	1,96	6,76
311.	158,80	81,52	0,51	1,47	6,86	97,37	76,08	0,78	2,29	9,41
312.	219,95	131,82	0,60	2,91	20,46	156,27	148,18	0,95	2,72	12,10
313.	82,34	61,74	0,75	1,42	5,15	100,26	115,72	1,15	2,17	8,28
314.	151,59	93,40	0,62	0,88	4,21	253,87	210,12	0,83	0,75	3,29
315.	104,54	68,67	0,66	1,83	7,99	149,25	133,94	0,90	2,24	10,60
316.	182,94	105,17	0,57	1,12	4,30	158,98	96,78	0,61	0,85	3,30
317.	81,80	60,18	0,74	1,66	6,14	95,07	72,01	0,76	0,86	3,05
318.	103,28	66,10	0,64	1,32	5,39	89,22	79,21	0,89	2,87	14,77
319.	227,36	120,84	0,53	0,94	4,36	207,18	161,44	0,78	1,94	8,14
320.	171,10	92,79	0,54	0,41	3,14	296,05	206,15	0,70	0,53	2,71
321.	188,19	151,99	0,81	2,03	11,51	123,71	80,17	0,65	1,52	8,33
322.	169,03	187,91	1,11	3,50	22,89	194,58	244,93	1,26	10,61	135,43
323.	91,63	67,56	0,74	1,88	8,82	105,67	131,49	1,24	2,82	13,96
324.	101,61	64,57	0,64	2,16	10,77	98,31	94,88	0,97	2,31	9,68
325.	191,94	109,35	0,57	1,32	6,42	228,47	169,24	0,74	1,33	4,98
326.	129,61	83,20	0,64	3,97	34,32	77,27	79,16	1,02	4,07	22,72
327.	143,81	78,40	0,55	0,82	3,69	116,44	109,27	0,94	2,17	7,73
328.	161,05	89,90	0,56	0,92	4,29	192,94	193,68	1,00	1,25	3,50
329.	117,96	85,80	0,73	1,67	7,90	166,57	177,67	1,07	1,95	8,23
330.	162,60	81,54	0,50	1,07	4,86	160,78	175,81	1,09	2,13	7,84
331.	162,07	104,67	0,65	2,42	16,19	95,73	69,66	0,73	3,53	26,69
332.	120,61	87,70	0,73	2,50	12,76	121,41	173,31	1,43	3,33	17,39
333.	147,81	99,98	0,68	1,31	5,11	200,39	123,54	0,62	1,12	4,66
334.	161,64	90,66	0,56	0,95	4,58	258,47	188,02	0,73	1,03	4,73
335.	268,80	213,79	0,80	5,40	52,62	264,59	200,22	0,76	1,38	5,41
336.	102,33	64,32	0,63	0,94	4,28	162,83	127,39	0,78	1,18	4,92
337.	164,54	96,05	0,58	0,99	4,39	208,93	197,15	0,94	1,48	5,12
338.	110,68	89,44	0,81	1,39	4,77	157,94	201,87	1,28	1,55	4,61
339.	233,76	109,05	0,47	0,86	4,36	145,79	126,36	0,87	3,51	21,16
340.	206,89	108,53	0,52	0,77	3,50	185,62	161,79	0,87	2,08	7,72
341.	99,90	78,56	0,79	1,33	4,57	138,08	171,88	1,24	1,62	4,96
342.	102,47	57,25	0,56	1,39	5,95	59,36	47,66	0,80	3,33	22,31
343.	70,58	31,19	0,44	1,05	4,93	37,22	24,91	0,67	4,78	43,99
344.	189,84	145,08	0,76	2,01	9,08	266,80	174,99	0,66	1,26	5,60
345.	120,12	91,83	0,76	2,93	14,85	101,57	186,31	1,83	4,24	23,17
346.	91,06	63,30	0,70	1,88	7,90	83,78	130,53	1,56	3,16	13,46
347.	210,52	136,63	0,65	3,19	25,71	300,35	211,86	0,71	0,99	3,74
348.	266,08	144,55	0,54	0,95	4,21	255,61	232,32	0,91	1,92	7,93
349.	92,24	71,91	0,78	1,54	5,83	83,16	55,94	0,67	1,14	4,28
350.	109,98	146,61	1,33	2,95	14,31	95,02	70,53	0,74	2,25	11,37
351.	210,92	132,28	0,63	2,91	20,25	261,73	198,60	0,76	1,03	3,43
352.	250,11	173,22	0,69	1,40	5,41	214,57	147,27	0,69	1,25	4,76
353.	63,10	82,06	1,30	4,58	30,18	88,54	54,61	0,62	2,22	11,24
354.	256,90	146,84	0,57	0,90	3,62	162,36	108,33	0,67	2,27	13,78
355.	128,18	101,90	0,79	4,03	30,62	139,04	236,66	1,70	5,03	40,26
356.	172,76	108,90	0,63	1,43	6,19	278,36	231,26	0,83	1,59	6,36
357.	200,98	120,69	0,60	1,30	5,17	202,40	187,16	0,92	1,62	5,42
358.	96,58	68,81	0,71	1,95	7,84	95,40	139,12	1,46	2,93	12,17
359.	106,49	79,72	0,75	4,10	33,10	103,24	114,11	1,11	1,80	5,57
360.	53,83	41,85	0,78	3,72	23,17	96,77	47,87	0,49	0,87	3,95
361.	164,15	77,44	0,47	1,23	6,01	80,98	41,87	0,52	0,78	5,27
362.	110,15	130,46	1,18	2,30	7,84	124,05	67,66	0,55	0,87	3,37
363.	136,49	77,55	0,57	1,30	6,19	263,61	126,39	0,48	1,15	5,50
364.	146,85	72,68	0,49	0,49	2,47	94,07	82,30	0,87	2,81	12,76
365.	107,89	65,42	0,61	1,42	5,90	141,43	104,10	0,74	1,21	5,03
366.	208,24	155,85	0,75	2,04	7,89	263,11	154,53	0,59	2,20	13,27
367.	282,37	155,44	0,55	0,73	3,47	234,97	194,98	0,83	1,76	6,43
368.	161,92	107,20	0,66	1,36	5,40	152,57	103,64	0,68	1,32	5,23
369.	221,38	134,11	0,61	1,05	3,73	181,80	119,73	0,66	1,33	6,40
370.	271,79	173,24	0,64	1,37	5,64	207,47	151,04	0,73	1,71	7,00
371.	189,20	100,76	0,53	1,26	5,79	282,30	197,88	0,70	1,17	5,77
372.	127,18	71,71	0,56	1,26	4,77	111,87	142,08	1,27	2,58	9,26
373.	207,89	129,94	0,63	2,11	11,22	124,58	105,15	0,84	3,14	17,03
374.	99,69	62,51	0,63	0,99	3,95	141,47	116,40	0,82	1,82	7,09
375.	50,08	59,01	1,18	3,25	16,50	62,20	39,99	0,64	1,31	5,51
376.	142,14	96,25	0,68	3,99	32,37	108,09	123,78	1,15	3,27	17,68
377.	281,65	184,02	0,65	2,03	9,66	183,36	148,36	0,81	1,87	7,06
378.	140,52	139,76	0,99	2,20	8,32	173,50	90,62	0,52	1,45	7,58
379.	230,67	120,28	0,52	1,24	6,07	162,82	134,10	0,82	2,25	9,82
380.	159,83	118,76	0,74	1,61	6,83	130,42	77,48	0,59	2,00	11,06
381.	143,56	100,39	0,70	1,08	4,02	235,75	213,23	0,90	1,02	3,70

inst. n.	Distribution of CVP intermediate effort					Distribution of CIBI intermediate effort				
	$\mu$	$\sigma$	CV	skew	kurt	$\mu$	$\sigma$	CV	skew	kurt
382.	126,05	80,68	0,64	1,74	10,03	207,50	158,17	0,76	2,27	12,73
383.	219,25	120,56	0,55	0,97	4,73	269,03	236,20	0,88	1,40	4,55
384.	73,32	72,36	0,99	2,48	10,92	91,31	159,54	1,75	2,65	10,49
385.	222,26	149,08	0,67	0,93	3,51	203,41	106,27	0,52	1,04	5,12
386.	191,84	117,47	0,61	1,80	8,43	293,97	203,44	0,69	1,81	9,13
387.	171,23	185,94	1,09	2,88	13,74	151,30	99,92	0,66	1,61	6,78
388.	215,71	161,21	0,75	4,37	37,47	304,71	210,74	0,69	1,14	4,83
389.	150,32	93,01	0,62	1,58	7,46	212,95	188,82	0,89	1,57	7,00
390.	136,98	79,27	0,58	1,21	5,06	251,97	135,06	0,54	1,18	5,07
391.	192,55	151,87	0,79	2,39	13,50	201,86	130,75	0,65	1,33	5,30
392.	205,34	113,17	0,55	0,99	4,05	310,10	189,60	0,61	1,03	4,21
393.	158,02	114,19	0,72	0,99	3,76	256,14	260,35	1,02	0,99	3,23
394.	98,51	78,11	0,79	1,64	6,00	98,20	69,98	0,71	1,55	6,96
395.	172,68	102,33	0,59	0,69	3,03	119,67	81,92	0,68	1,83	8,08
396.	207,95	161,05	0,77	1,27	5,47	133,34	82,83	0,62	1,22	5,48
397.	91,55	67,90	0,74	1,84	8,13	147,19	87,57	0,59	1,06	5,05
398.	149,14	113,87	0,76	2,68	13,57	246,34	146,91	0,60	1,23	5,26
399.	108,43	90,71	0,84	3,44	19,64	190,95	102,69	0,54	1,37	5,74
400.	242,72	174,91	0,72	1,31	5,13	180,72	101,82	0,56	1,01	4,23
401.	148,77	89,10	0,60	2,58	18,04	214,44	143,51	0,67	1,01	4,03
402.	87,59	58,81	0,67	1,53	7,63	162,09	99,51	0,61	0,95	4,53
403.	125,34	110,09	0,88	3,97	28,80	208,07	112,28	0,54	1,12	4,95
404.	146,62	94,31	0,64	1,46	6,14	184,50	143,84	0,78	1,54	7,31
405.	94,15	85,63	0,91	3,02	16,53	114,81	74,82	0,65	1,45	6,68
406.	127,26	79,04	0,62	1,67	7,75	145,13	149,37	1,03	2,64	13,47
407.	115,31	87,25	0,76	1,77	7,10	139,81	188,83	1,35	2,04	7,23
408.	157,59	100,17	0,64	1,23	5,95	276,69	154,78	0,56	0,89	4,62
409.	210,49	156,72	0,74	2,13	9,88	232,57	166,58	0,72	1,56	6,59
410.	97,16	75,87	0,78	1,73	6,61	111,25	73,45	0,66	1,56	7,06
411.	105,74	137,36	1,30	3,35	17,15	123,31	72,29	0,59	1,65	7,72
412.	226,04	144,45	0,64	1,04	3,70	180,97	115,45	0,64	1,92	10,53
413.	156,07	109,20	0,70	1,57	6,70	203,75	165,75	0,81	1,64	6,83
414.	157,81	141,75	0,90	3,27	17,75	233,56	129,98	0,56	1,16	4,78
415.	106,38	84,33	0,79	1,14	3,64	140,47	186,59	1,33	1,55	4,26
416.	91,00	67,35	0,74	3,57	24,45	78,60	140,68	1,79	4,79	31,72
417.	122,47	64,18	0,52	0,63	4,29	244,60	118,30	0,48	0,99	5,03
418.	73,16	44,49	0,61	1,26	6,87	148,86	82,66	0,56	1,71	8,25
419.	203,27	111,01	0,55	0,99	5,23	174,58	193,01	1,11	2,95	12,69
420.	110,14	55,28	0,50	1,00	4,66	98,55	98,65	1,00	2,16	7,75
421.	208,85	123,86	0,59	1,63	7,80	281,61	202,15	0,72	1,07	4,09
422.	269,44	130,03	0,48	0,74	3,48	149,53	79,64	0,53	0,93	4,64
423.	213,22	137,32	0,64	1,58	6,98	182,51	119,85	0,66	1,66	6,65
424.	193,96	99,69	0,51	0,89	3,91	182,79	168,25	0,92	2,22	9,44
425.	107,32	71,78	0,67	1,25	5,50	146,32	149,36	1,02	1,33	4,21
426.	175,93	95,54	0,54	1,14	4,34	218,03	186,27	0,85	1,58	5,93
427.	169,30	92,88	0,55	1,10	4,78	136,47	170,58	1,25	2,88	11,61
428.	101,64	55,42	0,55	3,33	26,58	57,88	51,72	0,89	3,90	22,49
429.	284,25	167,58	0,59	2,54	16,81	186,31	133,50	0,72	1,49	5,36
430.	218,12	114,47	0,52	1,04	5,47	180,99	157,16	0,87	1,97	6,88
431.	173,41	101,67	0,59	1,22	6,28	308,04	186,47	0,61	1,22	6,16
432.	166,55	106,45	0,64	1,34	5,85	201,02	164,86	0,82	1,33	5,15
433.	140,55	110,14	0,78	3,27	24,26	130,66	94,28	0,72	1,49	6,08
434.	108,40	55,71	0,51	1,29	5,92	55,04	30,79	0,56	1,04	5,15
435.	235,97	128,28	0,54	1,03	5,25	278,95	200,09	0,72	1,07	3,45
436.	237,47	144,91	0,61	1,79	8,37	134,57	104,16	0,77	2,87	16,52
437.	232,33	129,74	0,56	1,94	10,03	179,31	151,79	0,85	2,46	11,60
438.	133,78	97,16	0,73	3,19	22,50	134,93	109,46	0,81	1,79	7,40
439.	130,30	83,34	0,64	2,05	13,60	243,95	157,98	0,65	2,54	17,21
440.	114,67	109,70	0,96	2,86	14,60	149,86	100,84	0,67	2,42	14,67
441.	191,60	141,77	0,74	1,24	4,20	168,33	103,52	0,61	1,15	4,62
442.	116,73	74,07	0,63	1,77	8,19	128,39	156,11	1,22	2,60	11,22
443.	190,11	113,42	0,60	0,96	4,69	294,21	196,16	0,67	1,26	5,75
444.	98,90	94,29	0,95	1,64	5,30	94,91	50,91	0,54	0,56	2,91
445.	293,57	164,84	0,56	1,64	7,54	205,57	158,93	0,77	2,43	11,48
446.	106,58	96,12	0,90	1,98	8,28	96,19	60,63	0,63	1,38	5,89
447.	167,87	170,63	1,02	1,74	5,13	182,84	96,76	0,53	0,59	2,98
448.	86,34	86,72	1,00	2,72	10,90	122,76	67,05	0,55	2,42	16,03
449.	192,85	114,23	0,59	1,07	4,98	195,05	140,27	0,72	1,38	4,99
450.	62,53	44,90	0,72	1,20	4,99	83,33	73,08	0,88	1,33	5,07
451.	220,80	131,95	0,60	1,23	5,21	222,13	214,17	0,96	2,53	11,87
452.	170,42	135,75	0,80	2,37	10,77	274,93	145,93	0,53	1,29	6,35
453.	87,30	53,47	0,61	1,28	5,44	86,10	64,67	0,75	1,60	6,64
454.	121,83	67,86	0,56	0,60	2,97	178,54	137,63	0,77	0,90	3,12
455.	92,79	51,51	0,56	1,26	5,72	87,98	89,95	1,02	2,50	10,28
456.	146,03	152,65	1,05	2,02	7,09	154,41	85,44	0,55	0,58	2,88
457.	122,28	103,91	0,85	1,02	3,17	200,03	229,71	1,15	1,05	2,95
458.	152,19	92,45	0,61	1,08	5,68	90,44	60,15	0,67	2,13	9,84
459.	250,10	161,67	0,65	1,02	4,03	160,78	101,65	0,63	1,72	10,70
460.	170,01	93,51	0,55	1,06	5,19	172,39	196,82	1,14	2,03	7,70
461.	141,05	84,86	0,60	1,23	5,64	201,19	156,25	0,78	1,37	5,97
462.	232,07	137,59	0,59	1,65	7,75	300,52	190,21	0,63	1,10	4,25

inst.	Distribution of CVP intermediate effort					Distribution of CIBI intermediate effort				
	n.	$\mu$	$\sigma$	CV	skew	kurt	$\mu$	$\sigma$	CV	skew
463.	161,94	112,43	0,69	2,04	11,90	229,54	147,32	0,64	1,26	5,41
464.	241,86	135,88	0,56	0,92	4,09	266,10	191,94	0,72	1,45	5,36
465.	230,19	124,49	0,54	1,17	4,48	326,62	209,54	0,64	0,96	4,05
466.	101,87	99,08	0,97	7,42	81,83	102,02	111,29	1,09	1,88	6,42
467.	222,29	120,16	0,54	1,02	4,65	245,80	211,21	0,86	1,65	6,33
468.	191,36	101,19	0,53	1,25	7,11	380,56	199,07	0,52	1,37	7,44
469.	33,65	33,90	1,01	3,93	22,11	28,56	66,67	2,33	5,22	32,35
470.	218,39	124,06	0,57	1,18	4,89	172,01	194,93	1,13	2,94	13,10
471.	113,73	64,69	0,57	1,40	6,27	139,30	139,10	1,00	1,62	5,44
472.	184,40	101,88	0,55	1,07	4,90	284,16	191,50	0,67	0,82	3,44
473.	237,93	129,53	0,54	1,89	11,48	175,63	182,41	1,04	2,86	12,79
474.	81,04	70,14	0,87	4,37	35,08	133,66	70,38	0,53	1,55	7,08
475.	145,79	95,29	0,65	2,68	18,22	171,24	218,69	1,28	2,79	15,49
476.	153,19	146,09	0,95	6,61	66,56	157,88	139,60	0,88	1,82	7,06
477.	193,58	110,23	0,57	1,81	10,00	333,27	190,15	0,57	0,86	3,95
478.	78,45	81,39	1,04	2,06	7,05	89,50	51,43	0,57	0,95	3,61
479.	63,86	47,75	0,75	1,34	5,25	74,21	99,19	1,34	2,24	7,85
480.	91,22	118,85	1,30	2,35	8,31	98,52	63,17	0,64	1,43	5,86
481.	45,93	38,69	0,84	1,56	5,76	56,24	82,59	1,47	2,17	7,22
482.	224,84	119,08	0,53	1,33	6,41	136,45	88,41	0,65	2,01	9,94
483.	164,48	97,25	0,59	1,28	5,70	118,60	102,73	0,87	2,36	9,93
484.	65,37	76,95	1,18	3,33	15,07	75,07	165,25	2,20	3,49	15,31
485.	232,73	142,38	0,61	2,11	10,06	206,02	177,76	0,86	1,96	7,49
486.	243,16	147,78	0,61	1,01	4,12	139,06	87,20	0,63	1,48	8,47
487.	67,58	39,69	0,59	2,04	11,26	50,11	75,44	1,51	4,52	27,22
488.	239,50	181,23	0,76	4,44	33,99	188,22	165,89	0,88	2,25	9,20
489.	219,53	122,72	0,56	1,65	7,33	175,44	170,76	0,97	2,42	10,31
490.	176,62	155,91	0,88	1,91	6,98	209,59	123,79	0,59	2,19	12,62
491.	150,81	85,29	0,57	1,38	6,50	158,23	167,00	1,06	1,80	5,80
492.	107,40	62,86	0,59	1,90	9,76	83,62	126,82	1,52	3,81	19,57
493.	253,65	155,33	0,61	1,27	5,14	204,19	144,29	0,71	1,85	9,14
494.	148,19	100,36	0,68	1,27	4,86	144,20	96,35	0,67	1,06	3,85
495.	244,78	131,49	0,54	1,27	6,28	148,01	104,74	0,71	2,09	8,26
496.	100,74	109,75	1,09	2,03	6,76	119,01	67,67	0,57	1,02	4,39
497.	192,38	111,66	0,58	1,95	11,92	381,88	212,65	0,56	2,38	14,10
498.	109,17	69,37	0,64	3,36	18,90	68,44	101,70	1,49	6,74	58,70
499.	200,25	104,53	0,52	0,99	4,95	166,20	138,51	0,83	2,05	7,85
500.	122,39	78,82	0,64	0,71	3,71	215,84	169,33	0,78	0,80	3,43
501.	268,41	155,43	0,58	2,28	13,08	137,62	91,68	0,67	2,08	11,39
502.	216,03	192,27	0,89	3,87	28,01	210,57	134,73	0,64	1,58	7,72
503.	66,81	57,94	0,87	2,76	12,56	102,80	48,46	0,47	0,89	3,64
504.	113,26	69,73	0,62	1,41	5,94	143,18	136,33	0,95	1,80	7,29
505.	146,62	82,36	0,56	1,14	4,80	82,68	49,23	0,60	1,09	5,27
506.	208,20	121,23	0,58	1,41	7,32	225,87	180,95	0,80	0,96	3,11
507.	208,34	123,46	0,59	1,74	8,50	293,89	189,85	0,65	1,18	5,62
508.	211,19	130,91	0,62	1,34	5,68	251,72	242,68	0,96	2,41	10,67
509.	283,58	147,36	0,52	1,25	5,40	208,56	168,56	0,81	1,74	6,14
510.	171,35	106,48	0,62	1,39	5,51	257,56	173,40	0,67	1,52	6,72
511.	144,35	147,38	1,02	1,84	6,29	146,13	88,33	0,60	1,05	4,01
512.	25,85	12,70	0,49	0,91	5,14	50,91	23,70	0,47	1,32	5,71
513.	138,48	79,28	0,57	1,20	6,23	275,94	154,06	0,56	1,41	6,67
514.	183,96	109,32	0,59	0,79	3,42	129,42	79,56	0,61	1,31	5,45
515.	121,09	70,10	0,58	1,91	10,57	235,16	120,76	0,51	1,40	7,14
516.	207,85	100,74	0,48	0,93	4,03	107,85	62,52	0,58	0,89	4,60
517.	54,12	41,40	0,77	2,11	9,24	53,89	87,98	1,63	3,09	12,92
518.	138,55	90,60	0,65	1,44	6,27	253,35	158,00	0,62	1,78	8,12
519.	185,33	120,30	0,65	0,89	3,54	180,67	134,35	0,74	1,88	7,72
520.	106,33	106,02	1,00	1,75	6,27	169,33	230,56	1,36	1,73	5,69
521.	71,45	89,07	1,25	5,75	45,35	114,83	62,73	0,55	1,79	8,54
522.	255,24	135,28	0,53	1,01	4,10	146,24	83,81	0,57	1,09	5,59
523.	135,65	84,58	0,62	1,33	6,07	193,31	167,63	0,87	1,72	7,87
524.	100,71	57,37	0,57	2,06	13,84	190,41	96,43	0,51	0,65	3,54
525.	184,00	112,67	0,61	1,67	8,27	275,73	199,18	0,72	1,47	8,37
526.	118,92	67,94	0,57	0,71	3,87	198,00	125,25	0,63	0,95	4,06
527.	237,47	140,76	0,59	2,77	19,88	154,35	116,95	0,76	2,19	8,62
528.	174,28	85,98	0,49	0,49	2,90	278,22	200,95	0,72	0,59	2,62
529.	98,20	55,55	0,57	0,64	3,61	196,59	103,80	0,53	0,87	3,87
530.	140,90	125,56	0,89	3,31	24,87	102,48	66,95	0,65	2,29	15,70
531.	78,62	42,11	0,54	0,94	4,41	42,50	23,97	0,56	1,13	5,46
532.	108,21	100,26	0,93	2,19	8,35	140,54	76,90	0,55	1,98	12,60
533.	41,82	46,83	1,12	4,73	34,71	41,90	97,94	2,34	5,19	36,60
534.	122,80	84,59	0,69	2,54	18,45	189,27	179,34	0,95	2,64	17,52
535.	157,68	100,87	0,64	1,38	5,31	208,63	164,25	0,79	1,49	5,65
536.	139,52	104,19	0,75	3,08	18,67	242,81	124,63	0,51	0,56	2,94
537.	191,48	112,07	0,59	1,13	4,67	218,08	199,86	0,92	1,62	5,62
538.	96,24	66,32	0,69	2,05	8,69	124,45	81,42	0,65	0,89	3,87
539.	113,50	78,13	0,69	1,77	8,01	197,30	105,70	0,54	0,91	3,79
540.	88,86	80,43	0,91	1,64	6,11	125,66	175,02	1,39	1,84	6,11
541.	210,19	129,76	0,62	2,07	12,16	330,92	177,27	0,54	0,83	3,85
542.	153,43	89,35	0,58	0,72	3,14	211,31	194,67	0,92	1,00	3,14
543.	79,37	70,01	0,88	1,44	4,81	73,84	53,09	0,72	1,45	6,24

inst. n.	Distribution of CVP intermediate effort					Distribution of CIBI intermediate effort				
	$\mu$	$\sigma$	CV	skew	kurt	$\mu$	$\sigma$	CV	skew	kurt
544.	210,73	121,63	0,58	1,66	8,16	132,39	89,72	0,68	3,19	22,65
545.	188,50	103,05	0,55	1,31	5,38	126,94	137,29	1,08	4,04	23,67
546.	171,99	97,25	0,57	0,71	3,40	175,32	139,36	0,79	1,69	6,31
547.	132,87	116,56	0,88	1,46	4,85	139,79	86,04	0,62	1,54	6,55
548.	210,73	143,33	0,68	1,34	5,14	176,15	113,91	0,65	1,12	4,18
549.	182,83	125,37	0,69	2,50	15,31	335,64	230,12	0,69	2,95	20,66
550.	95,21	77,06	0,81	2,42	14,57	86,17	63,70	0,74	1,56	5,60
551.	152,56	78,12	0,51	1,02	4,10	90,82	70,06	0,77	3,73	25,44
552.	143,72	153,57	1,07	1,97	7,26	122,46	70,00	0,57	1,17	5,48
553.	128,57	103,19	0,80	1,44	6,81	215,48	197,71	0,92	0,87	3,24
554.	93,88	93,91	1,00	1,74	6,63	80,49	53,28	0,66	1,09	4,36
555.	157,68	130,23	0,83	1,19	3,80	151,30	87,03	0,58	1,19	5,15
556.	164,63	92,89	0,56	1,17	6,04	142,26	127,25	0,89	2,15	8,48
557.	248,27	135,95	0,55	1,04	4,66	173,67	131,53	0,76	1,91	7,46
558.	205,72	150,43	0,73	1,63	7,03	244,02	134,15	0,55	0,95	3,83
559.	96,53	61,43	0,64	0,95	3,84	124,10	102,71	0,83	1,17	3,83
560.	111,29	71,17	0,64	1,63	8,25	149,15	117,90	0,79	1,29	4,95
561.	181,98	111,38	0,61	1,33	5,51	246,54	168,11	0,68	1,05	3,72
562.	175,69	119,80	0,68	1,90	8,93	174,24	136,31	0,78	1,99	9,90
563.	164,16	89,72	0,55	1,09	5,11	124,67	115,45	0,93	3,10	17,43
564.	168,60	117,28	0,70	1,24	4,24	227,39	141,92	0,62	1,50	5,99
565.	84,73	126,22	1,49	2,97	12,55	92,49	61,91	0,67	1,75	7,40
566.	174,74	108,05	0,62	0,93	3,63	134,66	97,39	0,72	1,51	6,56
567.	100,56	62,55	0,62	1,05	4,49	103,94	74,86	0,72	1,36	4,90
568.	203,17	126,74	0,62	1,73	10,17	230,51	252,20	1,09	2,65	14,62
569.	103,10	64,57	0,63	1,89	9,48	185,05	94,11	0,51	0,64	2,85
570.	234,77	126,50	0,54	0,75	3,56	140,52	93,68	0,67	1,42	6,14
571.	150,60	112,78	0,75	2,53	12,75	190,25	119,25	0,63	1,08	5,17
572.	150,33	87,94	0,58	1,04	4,26	214,86	173,92	0,81	1,16	4,01
573.	107,87	90,60	0,84	2,45	10,17	171,40	103,25	0,60	1,80	7,53
574.	170,57	81,32	0,48	0,58	2,68	170,30	175,15	1,03	1,66	4,90
575.	137,34	154,08	1,12	2,69	10,48	166,39	84,30	0,51	0,92	4,38
576.	186,42	83,54	0,45	1,04	5,04	93,24	54,34	0,58	2,54	19,47
577.	143,80	70,73	0,49	0,72	3,64	93,36	88,26	0,95	3,40	17,67
578.	197,66	125,65	0,64	2,35	15,20	242,34	225,71	0,93	3,81	29,40
579.	85,53	62,36	0,73	0,97	3,80	122,87	136,99	1,11	1,29	4,04
580.	113,96	128,66	1,13	6,04	57,01	92,97	81,02	0,87	3,49	22,62
581.	101,43	67,99	0,67	0,96	3,80	153,85	133,34	0,87	1,18	4,34
582.	155,86	103,68	0,67	1,47	8,12	327,32	191,09	0,58	1,92	9,98
583.	149,02	118,10	0,79	3,12	21,09	132,55	95,49	0,72	1,66	6,99
584.	213,34	144,11	0,68	1,59	7,67	113,72	71,30	0,63	1,54	7,50
585.	163,52	82,77	0,51	0,69	4,27	276,09	175,85	0,64	0,83	4,02
586.	157,81	124,41	0,79	1,71	6,97	192,92	113,01	0,59	1,39	6,06
587.	180,34	99,71	0,55	1,23	6,37	269,44	185,87	0,69	0,74	3,17
588.	134,78	99,15	0,74	2,55	15,37	158,63	174,38	1,10	1,85	7,15
589.	216,30	121,39	0,56	2,05	14,57	323,64	267,89	0,83	2,03	12,69
590.	266,89	131,96	0,49	1,22	4,86	175,82	135,59	0,77	2,45	11,41
591.	97,38	68,25	0,70	1,64	7,11	113,88	144,36	1,27	2,29	9,06
592.	89,99	109,30	1,21	2,39	8,54	101,98	60,54	0,59	1,27	5,63
593.	160,10	103,71	0,65	0,94	4,16	297,55	207,66	0,70	0,92	4,17
594.	226,15	154,52	0,68	1,60	6,65	291,57	207,07	0,71	1,86	8,51
595.	143,54	96,33	0,67	0,88	3,68	221,29	218,42	0,99	0,93	3,11
596.	112,40	72,11	0,64	0,82	3,15	132,05	151,53	1,15	1,55	4,53
597.	206,89	137,04	0,66	2,70	18,87	284,40	182,57	0,64	0,82	3,28
598.	263,99	164,03	0,62	0,42	2,31	149,68	86,88	0,58	0,37	2,90
599.	189,67	129,81	0,68	1,16	5,79	137,38	97,63	0,71	2,28	10,46
600.	201,09	100,64	0,50	0,84	3,60	134,46	119,04	0,89	3,03	15,04
601.	260,13	163,14	0,63	0,65	3,66	143,25	88,85	0,62	0,86	4,63
602.	229,29	168,74	0,74	1,67	8,38	204,26	129,53	0,63	1,48	6,38
603.	191,19	100,51	0,53	2,48	15,72	100,13	64,41	0,64	2,57	13,62
604.	240,81	171,86	0,71	2,37	12,35	264,87	188,28	0,71	1,38	5,63
605.	84,06	84,81	1,01	3,07	17,04	102,78	59,98	0,58	1,18	4,86
606.	242,62	136,67	0,56	1,39	6,13	221,41	162,50	0,73	1,18	4,17
607.	129,20	88,50	0,69	1,78	8,14	157,40	104,65	0,66	1,66	8,68
608.	149,85	89,17	0,60	1,43	9,48	237,64	160,58	0,68	0,69	3,13
609.	170,81	98,61	0,58	1,65	7,77	121,10	113,09	0,93	4,13	32,41
610.	248,23	163,56	0,66	2,28	11,15	313,53	217,57	0,69	1,10	5,01
611.	198,51	154,40	0,78	2,13	9,63	227,08	134,23	0,59	0,78	3,30
612.	134,39	82,42	0,61	0,96	4,88	224,98	153,44	0,68	0,72	3,19
613.	60,71	63,08	1,04	2,02	7,82	87,26	138,70	1,59	2,10	7,29
614.	213,73	144,06	0,67	2,72	15,93	295,59	212,32	0,72	1,32	5,78
615.	193,57	111,37	0,58	2,35	14,38	113,07	92,43	0,82	2,77	12,47
616.	106,75	76,30	0,71	1,35	5,61	145,69	169,41	1,16	1,65	5,74
617.	120,92	74,47	0,62	1,52	6,34	139,60	101,97	0,73	1,08	3,92
618.	138,25	90,30	0,65	1,63	9,64	217,63	132,24	0,61	0,69	3,20
619.	166,69	108,40	0,65	1,73	6,83	265,73	142,55	0,54	0,73	3,39
620.	82,93	60,40	0,73	1,80	9,42	68,76	53,13	0,77	1,61	5,77
621.	118,40	66,99	0,57	0,70	4,48	214,74	127,91	0,60	1,07	5,24
622.	99,88	43,31	0,43	0,82	4,04	51,03	23,79	0,47	0,88	5,25
623.	126,38	87,53	0,69	1,77	9,80	180,31	182,25	1,01	2,07	10,43
624.	139,80	109,62	0,78	1,94	8,11	224,26	133,15	0,59	1,68	8,40

inst. n.	Distribution of CVP intermediate effort					Distribution of CIBI intermediate effort				
	$\mu$	$\sigma$	CV	skew	kurt	$\mu$	$\sigma$	CV	skew	kurt
625.	223,92	133,52	0,60	2,03	10,24	268,56	199,74	0,74	2,01	10,55
626.	112,52	79,29	0,70	1,88	8,37	187,52	110,64	0,59	1,79	7,88
627.	214,00	123,01	0,57	0,88	3,85	242,82	182,67	0,75	1,10	3,52
628.	152,70	86,46	0,57	0,91	4,47	249,62	188,76	0,76	1,02	4,23
629.	55,07	53,39	0,97	4,61	29,96	99,81	51,14	0,51	0,73	3,81
630.	204,69	147,98	0,72	1,16	4,18	146,85	88,63	0,60	0,79	3,57
631.	259,14	195,49	0,75	2,02	8,88	230,58	167,32	0,73	1,99	10,31
632.	217,22	117,10	0,54	1,88	10,82	110,71	67,78	0,61	1,83	10,27
633.	240,96	126,20	0,52	0,64	2,84	144,25	88,39	0,61	1,79	9,72
634.	116,84	62,78	0,54	0,98	4,03	112,64	111,57	0,99	1,68	5,22
635.	160,80	102,41	0,64	1,69	7,04	141,09	194,01	1,38	3,15	14,25
636.	71,25	54,64	0,77	2,23	10,67	108,29	63,55	0,59	1,08	4,80
637.	69,81	37,20	0,53	0,88	3,51	48,71	56,14	1,15	3,31	15,36
638.	151,68	111,26	0,73	1,40	6,50	112,50	70,62	0,63	1,52	6,20
639.	192,04	190,64	0,99	2,20	10,33	152,16	95,00	0,62	1,57	7,73
640.	104,42	66,21	0,63	2,29	14,18	191,24	111,07	0,58	2,94	23,34
641.	152,98	105,84	0,69	1,77	7,83	237,11	134,43	0,57	1,16	4,79
642.	117,58	57,88	0,49	0,49	3,79	229,04	106,72	0,47	0,65	3,28
643.	102,52	69,18	0,67	2,14	10,34	185,50	97,68	0,53	1,33	5,91
644.	191,09	124,10	0,65	1,27	6,13	310,62	216,96	0,70	1,42	6,71
645.	170,41	111,14	0,65	1,59	6,49	227,65	144,78	0,64	0,95	4,00
646.	127,94	60,60	0,47	0,97	4,37	103,39	114,19	1,10	2,72	11,52
647.	250,68	145,94	0,58	1,74	9,53	150,04	96,35	0,64	1,50	6,34
648.	192,10	109,04	0,57	1,37	6,79	295,60	175,98	0,60	0,69	3,24
649.	106,86	83,60	0,78	1,68	6,92	143,81	187,17	1,30	1,93	6,79
650.	120,49	68,07	0,56	1,13	4,62	127,99	103,70	0,81	1,44	5,63
651.	155,24	135,97	0,88	2,44	14,69	270,04	291,60	1,08	2,23	12,39
652.	150,10	157,09	1,05	2,18	8,53	140,93	84,75	0,60	1,28	4,57
653.	288,88	151,14	0,52	1,00	4,39	162,15	120,91	0,75	2,71	14,49
654.	238,85	142,76	0,60	1,46	6,81	185,29	169,58	0,92	2,42	9,87
655.	118,33	97,42	0,82	1,34	5,69	185,75	212,19	1,14	1,46	5,31
656.	135,66	84,01	0,62	1,65	7,82	228,69	137,59	0,60	1,59	8,87
657.	107,14	117,87	1,10	1,91	6,03	116,24	62,85	0,54	0,89	3,38
658.	145,87	99,89	0,68	1,84	9,33	181,90	220,13	1,21	2,21	9,58
659.	199,76	141,59	0,71	1,25	4,67	137,86	93,26	0,68	1,46	5,98
660.	95,83	54,51	0,57	0,89	3,96	96,56	80,60	0,83	1,33	4,01
661.	64,11	46,31	0,72	1,14	4,53	74,73	74,14	0,99	1,54	5,24
662.	147,00	80,76	0,55	1,50	10,70	298,51	149,68	0,50	2,07	13,37
663.	59,29	47,59	0,80	2,49	15,77	80,71	86,59	1,07	3,48	26,07
664.	268,33	136,93	0,51	1,61	8,49	144,50	97,58	0,68	2,46	13,12
665.	100,89	57,90	0,57	2,71	20,95	188,71	90,99	0,48	0,49	3,26
666.	180,64	125,47	0,69	1,86	8,89	120,14	76,55	0,64	1,56	6,31
667.	264,63	117,91	0,45	0,58	3,25	132,84	78,62	0,59	2,37	19,58
668.	203,92	163,71	0,80	2,18	11,86	127,32	90,75	0,71	1,82	8,47
669.	232,61	130,59	0,56	0,98	4,18	166,87	182,96	1,10	4,32	27,47
670.	196,58	110,79	0,56	0,98	4,59	310,95	235,90	0,76	1,12	4,42
671.	67,55	105,84	1,57	3,17	13,24	69,13	51,11	0,74	2,05	8,36
672.	157,45	109,74	0,70	1,47	5,33	135,00	107,73	0,80	2,80	15,22
673.	118,18	78,12	0,66	1,79	8,06	144,23	93,58	0,65	1,18	5,37
674.	98,46	61,70	0,63	2,49	17,11	139,12	93,74	0,67	0,92	3,63
675.	137,66	72,84	0,53	0,51	3,12	238,59	140,78	0,59	0,52	2,88
676.	84,97	91,01	1,07	3,15	16,12	101,34	64,48	0,64	1,16	4,32
677.	63,72	76,24	1,20	6,15	54,16	94,63	53,08	0,56	1,58	9,02
678.	134,33	71,31	0,53	1,20	4,58	140,57	161,12	1,15	1,96	6,41
679.	96,99	49,57	0,51	0,33	3,18	194,51	94,63	0,49	0,55	3,17
680.	105,35	59,93	0,57	0,99	3,92	101,44	86,10	0,85	1,98	8,82
681.	108,82	110,99	1,02	4,15	32,42	133,78	82,37	0,62	1,62	8,18
682.	152,11	81,48	0,54	0,99	5,15	255,91	153,54	0,60	0,55	3,55
683.	110,90	70,23	0,63	0,54	2,86	174,05	155,21	0,89	0,81	2,74
684.	100,02	58,37	0,58	1,44	5,27	159,86	90,84	0,57	0,93	3,96
685.	208,95	119,19	0,57	1,38	7,58	114,60	63,20	0,55	1,01	6,11
686.	141,69	84,17	0,59	1,00	4,03	127,17	125,80	0,99	2,36	9,95
687.	56,45	44,72	0,79	1,34	4,41	66,40	64,93	0,98	2,05	8,65
688.	133,45	75,56	0,57	0,90	3,84	154,15	165,42	1,07	1,66	5,28
689.	157,81	106,99	0,68	1,75	8,01	111,61	87,18	0,78	2,85	16,28
690.	228,89	144,91	0,63	1,71	7,73	283,22	197,65	0,70	1,10	4,49
691.	117,73	75,97	0,65	1,94	11,68	218,56	108,06	0,49	0,75	3,86
692.	113,62	75,36	0,66	1,82	8,92	171,71	125,91	0,73	2,40	15,24
693.	138,48	116,00	0,84	2,89	13,07	221,81	126,23	0,57	3,56	29,80
694.	44,78	53,87	1,20	3,62	17,22	63,50	38,04	0,60	2,81	20,36
695.	187,45	131,16	0,70	1,80	9,12	111,57	67,53	0,61	1,60	7,69
696.	124,20	84,69	0,68	1,09	4,01	153,27	190,60	1,24	1,54	4,41
697.	269,92	133,77	0,50	0,62	3,54	193,07	160,25	0,83	2,78	13,60
698.	143,88	79,84	0,55	1,18	5,47	79,62	55,94	0,70	2,07	10,05
699.	90,78	60,89	0,67	1,42	7,24	139,08	132,86	0,96	1,66	6,93
700.	150,71	169,17	1,12	2,76	11,75	184,28	91,35	0,50	1,12	4,42
701.	98,65	73,74	0,75	3,25	19,18	81,82	150,00	1,83	4,79	29,33
702.	279,34	142,69	0,51	1,36	7,18	140,95	85,99	0,61	1,10	6,21
703.	213,31	170,28	0,80	1,12	4,04	133,15	77,66	0,58	1,05	4,10
704.	137,77	93,75	0,68	1,22	4,31	142,45	96,20	0,68	1,82	8,19
705.	229,40	115,31	0,50	1,26	6,09	165,90	149,14	0,90	2,58	12,24

inst. n.	Distribution of CVP intermediate effort					Distribution of CIBI intermediate effort				
	$\mu$	$\sigma$	CV	skew	kurt	$\mu$	$\sigma$	CV	skew	kurt
706.	210,15	131,24	0,62	1,05	4,77	361,24	224,61	0,62	1,29	5,36
707.	158,06	102,47	0,65	1,07	4,49	249,99	176,93	0,71	1,06	4,71
708.	141,54	90,95	0,64	1,14	4,80	226,04	174,63	0,77	1,33	5,47
709.	173,86	89,68	0,52	0,30	3,35	340,34	167,75	0,49	0,67	3,46
710.	116,32	62,23	0,53	1,06	6,96	231,44	117,08	0,51	1,38	8,20
711.	206,91	183,20	0,89	2,30	10,73	189,33	122,56	0,65	1,43	5,40
712.	116,81	73,00	0,62	1,51	6,51	138,28	109,39	0,79	1,21	4,14
713.	131,80	86,88	0,66	3,13	21,25	69,60	62,00	0,89	3,41	19,75
714.	105,46	69,66	0,66	1,67	6,68	109,91	157,08	1,43	2,37	8,40
715.	129,39	68,80	0,53	1,42	5,96	103,21	133,93	1,30	2,95	12,52
716.	222,84	115,59	0,52	0,97	4,10	142,49	117,06	0,82	3,52	22,74
717.	217,02	122,03	0,56	1,46	8,11	264,71	196,81	0,74	1,50	5,83
718.	83,10	48,89	0,59	1,34	5,93	69,81	73,81	1,06	3,03	16,22
719.	232,69	140,18	0,60	1,23	6,70	271,58	251,81	0,93	1,55	5,58
720.	159,34	174,78	1,10	1,76	6,50	125,73	77,05	0,61	1,46	6,43
721.	206,30	118,13	0,57	1,20	5,15	122,24	73,52	0,60	1,20	5,21
722.	163,54	136,24	0,83	1,16	3,71	126,72	74,46	0,59	0,84	4,42
723.	103,24	61,34	0,59	1,04	4,23	138,59	114,20	0,82	1,11	3,87
724.	167,46	98,02	0,59	1,02	5,13	197,37	151,76	0,77	1,11	3,53
725.	196,60	120,20	0,61	2,26	13,65	368,86	212,50	0,58	1,05	5,05
726.	130,17	90,50	0,70	1,00	3,47	163,73	197,26	1,20	1,52	4,39
727.	196,33	114,25	0,58	1,66	7,45	249,82	182,30	0,73	1,29	5,30
728.	213,27	113,79	0,53	0,86	4,42	130,83	77,50	0,59	1,70	8,93
729.	165,76	84,07	0,51	1,14	4,42	104,96	85,55	0,82	3,11	17,39
730.	134,93	114,75	0,85	2,37	9,70	194,37	100,15	0,52	1,05	5,19
731.	88,09	52,07	0,59	1,13	5,97	176,68	93,03	0,53	1,45	6,88
732.	153,05	74,67	0,49	1,00	4,52	83,34	62,12	0,75	2,68	14,88
733.	56,11	45,05	0,80	2,08	8,16	59,19	94,08	1,59	2,91	11,52
734.	180,70	105,95	0,59	1,46	6,94	202,79	197,12	0,97	2,20	10,45
735.	140,69	124,65	0,89	4,22	31,76	212,65	162,34	0,76	3,46	24,82
736.	206,10	136,93	0,66	3,00	21,81	364,26	270,07	0,74	3,21	24,49
737.	81,94	82,80	1,01	3,61	18,64	130,73	63,81	0,49	1,06	5,04
738.	175,74	91,93	0,52	0,89	4,61	303,57	169,27	0,56	0,63	2,99
739.	127,09	109,12	0,86	2,62	11,41	182,84	95,82	0,52	0,57	2,88
740.	146,55	83,73	0,57	0,94	5,69	285,61	165,17	0,58	1,34	6,22
741.	140,59	74,97	0,53	1,30	5,67	127,51	143,26	1,12	2,04	6,69
742.	172,64	102,60	0,59	1,23	7,52	325,86	178,90	0,55	1,64	9,23
743.	99,74	62,17	0,62	1,58	6,85	95,63	126,10	1,32	2,76	11,52
744.	58,27	57,39	0,99	2,85	13,03	74,92	47,41	0,63	1,65	7,38
745.	175,44	138,35	0,79	1,04	3,53	138,31	77,86	0,56	0,83	3,87
746.	239,46	162,46	0,68	1,46	5,56	243,10	166,10	0,68	2,27	13,71
747.	172,11	126,28	0,73	0,73	3,20	323,81	263,36	0,81	0,73	3,04
748.	98,38	56,88	0,58	2,54	17,11	54,73	46,66	0,85	3,89	25,41
749.	208,09	119,69	0,58	1,43	7,07	325,45	226,67	0,70	1,69	8,76
750.	168,61	121,42	0,72	2,58	12,42	272,59	132,40	0,49	0,90	4,61
751.	270,98	178,84	0,66	1,28	4,94	311,45	229,42	0,74	1,44	6,00
752.	189,53	124,06	0,65	1,60	7,29	292,29	230,17	0,79	1,76	8,44
753.	186,34	101,30	0,54	0,96	4,74	265,73	227,24	0,86	1,24	4,78
754.	56,18	48,75	0,87	3,68	19,54	47,29	102,94	2,18	4,60	25,02
755.	133,72	86,32	0,65	1,37	5,70	213,19	139,73	0,66	1,92	9,60
756.	181,33	120,42	0,66	1,46	5,45	263,38	179,53	0,68	1,79	9,35
757.	242,33	141,14	0,58	0,89	3,76	299,37	226,92	0,76	1,24	4,78
758.	214,36	107,66	0,50	1,09	4,85	111,76	72,38	0,65	2,63	16,14
759.	248,51	174,12	0,70	3,30	22,42	215,71	182,99	0,85	1,92	7,42
760.	194,98	193,21	0,99	3,04	15,90	241,98	137,68	0,57	1,16	4,93
761.	141,24	77,86	0,55	1,26	5,21	126,65	126,91	1,00	2,26	8,87
762.	85,75	90,03	1,05	2,25	8,07	99,51	52,86	0,53	0,99	4,48
763.	117,08	67,01	0,57	0,85	3,09	114,69	135,45	1,18	1,85	5,61
764.	181,99	104,96	0,58	0,92	3,92	254,24	217,10	0,85	1,14	3,86
765.	178,87	115,43	0,65	1,20	4,54	156,85	106,74	0,68	1,51	5,55
766.	128,84	84,81	0,66	1,45	7,74	163,31	129,48	0,79	1,29	4,36
767.	188,43	104,65	0,56	1,03	5,54	362,80	201,46	0,56	1,21	6,19
768.	145,21	83,84	0,58	1,66	6,47	245,13	140,10	0,57	1,19	5,26
769.	147,65	120,39	0,82	1,52	5,87	112,14	68,33	0,61	1,19	5,35
770.	102,69	102,87	1,00	1,47	4,34	98,42	58,91	0,60	1,29	6,03
771.	200,89	120,79	0,60	1,39	7,96	280,95	218,77	0,78	2,26	13,46
772.	156,05	128,86	0,83	2,73	13,46	244,76	120,14	0,49	1,10	5,22
773.	173,29	109,65	0,63	1,50	6,38	125,06	99,45	0,80	2,75	14,41
774.	135,60	147,32	1,09	7,74	84,86	86,44	84,67	0,98	5,34	46,54
775.	150,58	81,06	0,54	0,88	4,70	256,42	146,76	0,57	0,84	3,66
776.	224,55	146,01	0,65	1,70	12,17	156,76	103,92	0,66	1,76	7,21
777.	163,27	125,95	0,77	1,78	6,10	195,21	109,51	0,56	1,28	6,55
778.	128,12	104,68	0,82	4,98	43,76	112,81	209,53	1,86	6,35	57,99
779.	148,61	119,24	0,80	5,58	50,81	266,49	129,02	0,48	1,07	5,06
780.	42,40	58,95	1,39	4,05	21,71	47,34	123,84	2,62	4,24	22,26
781.	131,27	110,44	0,84	1,43	5,67	118,80	77,99	0,66	1,66	6,68
782.	119,86	105,29	0,88	1,74	6,02	130,96	78,62	0,60	0,91	3,94
783.	109,68	60,50	0,55	0,51	3,09	219,95	114,86	0,52	0,71	3,27
784.	198,30	111,47	0,56	1,24	5,18	225,98	181,58	0,80	1,86	8,98
785.	149,42	74,49	0,50	0,80	5,01	300,59	141,05	0,47	1,16	5,41
786.	236,72	179,92	0,76	1,65	6,70	194,83	113,89	0,58	1,35	5,67

inst. n.	Distribution of CVP intermediate effort					Distribution of CIBI intermediate effort				
	$\mu$	$\sigma$	CV	skew	kurt	$\mu$	$\sigma$	CV	skew	kurt
787.	270,41	212,39	0,79	3,00	21,98	166,98	111,90	0,67	2,53	16,86
788.	82,82	65,09	0,79	2,81	15,10	105,95	65,70	0,62	1,38	5,93
789.	84,05	54,58	0,65	2,04	10,01	154,53	79,60	0,52	1,38	6,53
790.	54,29	38,10	0,70	1,97	7,70	43,25	65,37	1,51	3,56	16,07
791.	128,38	76,61	0,60	1,66	9,64	222,44	160,78	0,72	1,74	9,30
792.	214,36	209,80	0,98	5,94	54,74	313,09	181,76	0,58	1,18	5,21
793.	277,49	130,71	0,47	0,81	3,76	142,99	96,66	0,68	5,01	49,29
794.	195,78	138,97	0,71	1,44	5,32	225,93	136,43	0,60	1,52	8,29
795.	92,01	55,10	0,60	1,42	6,00	104,00	126,35	1,21	2,05	7,08
796.	241,83	190,93	0,79	5,54	54,43	189,76	137,53	0,72	2,28	12,48
797.	161,73	133,33	0,82	4,59	42,41	104,89	83,86	0,80	3,31	21,04
798.	127,35	111,07	0,87	1,72	5,82	159,72	87,04	0,54	1,41	5,88
799.	183,87	181,06	0,98	1,40	4,80	136,57	80,22	0,59	1,01	4,54
800.	120,29	77,89	0,65	1,35	6,49	124,45	106,82	0,86	4,05	31,68
801.	154,10	103,66	0,67	1,20	4,56	247,49	223,38	0,90	1,11	3,92
802.	121,48	111,40	0,92	6,66	68,01	132,94	223,78	1,68	7,30	77,88
803.	250,92	147,66	0,59	1,35	6,25	286,67	215,54	0,75	1,48	5,64
804.	139,22	100,96	0,73	5,85	61,34	270,87	128,39	0,47	0,77	3,75
805.	236,22	166,16	0,70	0,82	3,37	159,39	95,07	0,60	1,44	7,05
806.	139,73	114,70	0,82	1,53	5,23	150,69	95,78	0,64	1,49	6,91
807.	196,75	137,92	0,70	1,64	7,10	245,12	147,93	0,60	1,38	5,64
808.	152,35	168,14	1,10	3,48	22,12	190,29	118,55	0,62	3,21	23,60
809.	120,87	82,97	0,69	0,57	2,77	200,74	180,71	0,90	0,70	2,61
810.	133,30	77,89	0,58	1,66	7,01	103,78	140,01	1,35	3,43	16,19
811.	176,42	110,57	0,63	2,43	16,68	306,82	205,06	0,67	0,80	4,07
812.	113,58	69,13	0,61	1,72	9,55	199,89	107,64	0,54	0,90	4,48
813.	114,68	74,11	0,65	1,29	5,29	177,14	132,59	0,75	1,47	6,03
814.	148,02	89,80	0,61	1,00	4,99	220,21	195,88	0,89	1,20	4,82
815.	175,29	137,61	0,79	3,13	23,17	142,21	90,71	0,64	1,58	8,20
816.	37,33	43,45	1,16	4,28	24,35	36,36	90,18	2,48	4,86	27,87
817.	151,07	94,24	0,62	1,95	12,43	263,95	206,32	0,78	1,75	10,01
818.	119,11	96,13	0,81	2,05	8,34	153,02	83,40	0,55	1,02	5,62
819.	194,57	141,36	0,73	1,79	7,21	157,24	101,56	0,65	1,02	4,04
820.	234,07	146,27	0,62	1,24	4,79	289,83	232,67	0,80	1,61	6,05
821.	99,73	57,42	0,58	1,30	7,15	190,52	93,64	0,49	0,87	4,47
822.	180,00	93,29	0,52	0,41	3,07	326,59	191,11	0,59	0,57	2,95
823.	93,16	69,20	0,74	2,40	12,50	81,08	86,72	1,07	2,93	14,45
824.	80,86	49,66	0,61	1,46	5,97	74,26	82,59	1,11	2,87	14,58
825.	200,44	112,47	0,56	1,81	10,25	143,84	123,81	0,86	2,54	11,22
826.	245,73	142,64	0,58	1,12	5,82	181,03	115,52	0,64	1,44	5,64
827.	184,24	159,97	0,87	2,87	15,10	246,55	135,41	0,55	1,90	11,42
828.	152,50	80,33	0,53	1,17	5,62	99,23	113,26	1,14	4,75	33,87
829.	220,26	228,76	1,04	2,98	15,16	202,47	132,93	0,66	1,93	8,14
830.	84,10	64,36	0,77	2,33	11,03	89,54	60,80	0,68	1,05	3,61
831.	251,03	186,46	0,74	2,34	14,74	363,28	332,20	0,91	3,83	27,62
832.	107,43	55,11	0,51	0,80	4,25	210,12	102,94	0,49	0,69	3,57
833.	106,25	90,36	0,85	3,22	20,80	116,81	195,54	1,67	3,56	20,92
834.	200,09	133,30	0,67	1,43	6,18	192,13	123,11	0,64	1,43	5,99
835.	161,87	121,48	0,75	1,22	4,39	139,51	88,16	0,63	1,68	8,07
836.	91,93	55,09	0,60	1,68	8,88	80,89	114,82	1,42	3,23	15,30
837.	130,36	76,35	0,59	2,23	11,83	84,93	106,11	1,25	6,09	54,04
838.	236,26	144,13	0,61	1,01	4,84	150,94	104,36	0,69	1,63	7,18
839.	122,89	119,65	0,97	1,74	5,45	131,95	64,42	0,49	0,54	2,94
840.	185,73	106,43	0,57	0,72	3,84	293,16	213,22	0,73	0,92	3,83
841.	118,14	161,75	1,37	4,24	28,64	124,30	80,00	0,64	2,85	19,34
842.	112,11	120,21	1,07	2,17	7,90	122,76	67,94	0,55	0,82	3,71
843.	214,60	139,98	0,65	1,42	5,96	184,97	144,38	0,78	2,02	8,75
844.	196,83	113,80	0,58	0,87	4,06	290,94	228,02	0,78	1,09	4,10
845.	98,24	55,36	0,56	1,03	5,59	183,69	99,98	0,54	1,50	7,50
846.	89,56	49,93	0,56	1,12	5,02	68,17	62,48	0,92	2,65	11,37
847.	178,74	107,74	0,60	1,80	9,74	318,51	225,55	0,71	1,77	9,25
848.	120,95	86,74	0,72	1,18	4,01	148,38	100,82	0,68	1,83	8,02
849.	171,19	96,08	0,56	1,29	5,28	97,77	71,18	0,73	2,01	8,81
850.	80,35	48,93	0,61	2,55	18,07	158,54	77,51	0,49	1,34	5,72
851.	116,83	63,59	0,54	0,53	3,15	238,79	118,79	0,50	0,78	3,29
852.	182,55	103,13	0,56	0,99	4,09	219,09	182,29	0,83	1,47	5,26
853.	135,99	131,42	0,97	2,83	15,55	163,43	112,60	0,69	2,19	11,48
854.	120,67	125,53	1,04	2,39	9,64	134,00	78,20	0,58	1,35	7,54
855.	73,73	44,97	0,61	0,85	3,25	83,30	77,31	0,93	1,61	5,52
856.	138,13	163,08	1,18	2,76	12,80	132,40	74,89	0,57	1,92	9,80
857.	261,99	210,16	0,80	4,16	27,85	276,41	263,63	0,95	3,07	17,62
858.	159,12	118,78	0,75	3,58	23,50	201,12	134,53	0,67	0,83	3,30
859.	163,66	99,31	0,61	1,75	7,27	130,13	127,93	0,98	2,25	9,10
860.	155,82	116,93	0,75	1,39	5,31	275,08	255,04	0,93	1,24	4,58
861.	101,38	58,46	0,58	1,78	9,63	82,68	82,05	0,99	2,65	13,72
862.	137,93	81,70	0,59	1,18	4,46	122,80	105,13	0,86	2,06	8,40
863.	165,79	262,02	1,58	11,43	149,62	187,62	540,67	2,88	11,13	143,39
864.	72,37	48,82	0,67	2,61	15,63	129,47	61,08	0,47	0,73	3,14
865.	151,10	76,56	0,51	0,59	3,39	100,18	107,07	1,07	3,42	17,94
866.	166,46	96,25	0,58	0,78	4,39	300,27	176,88	0,59	0,51	3,01
867.	128,84	81,63	0,63	1,19	6,02	183,02	140,47	0,77	1,11	4,30



inst. n.	Distribution of CVP intermediate effort					Distribution of CIBI intermediate effort				
	$\mu$	$\sigma$	CV	skew	kurt	$\mu$	$\sigma$	CV	skew	kurt
868.	167,22	99,36	0,59	1,02	4,09	144,94	114,03	0,79	1,98	8,16
869.	141,65	81,17	0,57	0,94	4,05	140,10	127,39	0,91	1,75	6,21
870.	239,68	117,06	0,49	0,39	2,96	189,69	156,08	0,82	2,73	14,42
871.	186,13	102,80	0,55	0,81	4,22	173,02	196,94	1,14	2,48	10,03
872.	241,58	147,48	0,61	1,57	6,17	305,84	187,55	0,61	0,85	4,25
873.	267,02	128,47	0,48	1,06	5,13	133,37	68,91	0,52	0,62	4,57
874.	123,55	72,22	0,58	0,61	3,32	216,94	148,14	0,68	0,82	3,48
875.	59,15	40,60	0,69	1,57	6,90	61,84	75,17	1,22	2,87	13,78
876.	129,09	147,85	1,15	1,84	6,21	108,37	67,58	0,62	1,18	4,74
877.	228,20	191,89	0,84	3,61	23,60	323,08	196,98	0,61	1,62	7,86
878.	93,65	68,96	0,74	1,69	6,20	131,67	66,55	0,51	0,97	5,10
879.	177,43	108,10	0,61	1,69	7,73	205,99	157,95	0,77	1,77	8,62
880.	163,09	99,14	0,61	1,17	5,82	237,51	170,73	0,72	0,97	3,77
881.	153,81	83,23	0,54	1,28	6,36	302,03	154,95	0,51	1,27	6,32
882.	275,34	152,37	0,55	1,66	7,88	134,75	80,92	0,60	1,30	6,85
883.	208,95	114,11	0,55	1,01	4,14	319,69	207,06	0,65	1,20	5,12
884.	211,86	130,19	0,61	1,36	5,93	193,34	141,50	0,73	1,71	6,97
885.	187,31	96,09	0,51	0,87	4,24	307,39	196,39	0,64	0,65	3,05
886.	101,66	137,57	1,35	2,73	12,56	90,26	63,18	0,70	2,07	10,88
887.	248,11	181,91	0,73	1,17	4,24	186,30	110,27	0,59	1,72	10,63
888.	224,89	115,49	0,51	0,78	3,48	327,09	217,63	0,67	1,11	4,44
889.	139,23	80,78	0,58	1,81	7,18	92,27	144,01	1,56	4,72	25,18
890.	66,07	38,80	0,59	1,00	3,94	62,26	74,37	1,19	2,32	8,27
891.	165,04	132,67	0,80	2,15	11,06	134,64	96,85	0,72	1,73	7,74
892.	168,45	104,16	0,62	0,71	3,70	296,06	227,53	0,77	0,75	3,27
893.	189,37	129,91	0,69	1,39	5,13	213,05	139,00	0,65	1,39	5,28
894.	240,40	143,64	0,60	1,20	4,71	303,55	194,55	0,64	1,36	5,91
895.	176,37	98,47	0,56	1,33	5,32	157,43	140,42	0,89	2,18	9,40
896.	121,07	69,47	0,57	0,83	4,74	237,08	133,69	0,56	1,01	5,14
897.	164,05	104,53	0,64	1,68	9,56	297,32	211,60	0,71	1,72	9,72
898.	150,78	101,66	0,67	2,03	10,68	275,29	157,06	0,57	1,28	6,34
899.	195,64	107,76	0,55	1,25	5,79	282,42	225,03	0,80	1,53	6,49
900.	254,11	124,22	0,49	0,94	4,09	162,08	111,04	0,69	2,12	9,42
901.	98,54	71,09	0,72	1,76	8,59	150,37	127,29	0,85	1,61	7,24
902.	261,01	157,09	0,60	1,25	6,48	171,99	121,95	0,71	2,10	10,27
903.	115,86	67,56	0,58	1,58	7,69	76,72	71,96	0,94	2,55	10,33
904.	84,06	44,24	0,53	0,42	3,57	163,55	84,81	0,52	0,56	3,68
905.	238,59	190,30	0,80	6,89	73,61	161,02	140,49	0,87	3,01	16,97
906.	160,19	100,92	0,63	1,86	8,37	206,70	162,55	0,79	1,62	8,09
907.	206,82	129,60	0,63	1,08	4,46	233,91	194,67	0,83	1,77	7,38
908.	213,31	113,52	0,53	0,68	3,62	347,73	229,84	0,66	0,98	4,15
909.	134,20	94,06	0,70	1,19	6,06	216,09	205,17	0,95	1,26	5,44
910.	135,07	93,43	0,69	1,61	6,08	144,56	195,56	1,35	2,46	9,09
911.	191,01	161,83	0,85	2,45	11,64	195,88	129,68	0,66	1,27	5,27
912.	185,35	134,14	0,72	2,45	13,07	296,97	169,71	0,57	1,21	5,87
913.	106,50	91,21	0,86	2,45	10,64	161,90	85,77	0,53	1,90	11,56
914.	149,14	87,43	0,59	1,37	6,79	137,51	111,91	0,81	2,10	8,28
915.	284,69	244,49	0,86	5,66	54,55	281,14	255,66	0,91	2,30	10,40
916.	99,12	77,55	0,78	2,95	15,84	155,71	77,16	0,50	0,96	4,42
917.	98,87	73,63	0,74	3,49	20,01	76,11	134,27	1,76	5,46	37,42
918.	190,83	120,03	0,63	1,54	7,30	317,03	185,29	0,58	1,37	5,97
919.	147,78	102,04	0,69	2,53	14,20	195,78	186,62	0,95	2,46	15,50
920.	91,94	72,49	0,79	1,66	6,52	79,56	54,51	0,69	1,37	5,58
921.	119,12	66,18	0,56	1,36	7,19	220,73	112,96	0,51	0,98	4,97
922.	119,76	92,63	0,77	2,66	12,73	195,61	122,46	0,63	1,96	9,95
923.	212,82	138,49	0,65	1,66	8,04	305,20	170,48	0,56	0,79	3,92
924.	147,68	95,28	0,65	2,39	14,27	129,50	110,61	0,85	1,75	5,97
925.	93,99	104,70	1,11	7,29	78,64	130,90	91,63	0,70	1,86	8,93
926.	102,38	128,10	1,25	3,31	15,25	133,74	78,40	0,59	2,03	10,58
927.	132,84	84,23	0,63	1,52	6,35	178,70	114,77	0,64	0,92	3,87
928.	57,13	68,66	1,20	4,04	21,42	88,66	43,12	0,49	1,01	4,56
929.	253,33	155,77	0,61	2,23	13,23	177,25	138,46	0,78	2,17	8,44
930.	168,92	99,01	0,59	1,93	11,14	305,33	178,70	0,59	2,47	16,09
931.	186,46	118,75	0,64	1,05	5,10	336,41	181,46	0,54	0,89	3,85
932.	208,37	117,28	0,56	1,23	5,90	187,34	157,84	0,84	2,00	8,51
933.	113,24	110,72	0,98	3,43	18,81	165,34	86,42	0,52	1,47	6,54
934.	283,69	142,18	0,50	0,86	3,62	150,99	88,48	0,59	1,10	5,17
935.	167,01	88,98	0,53	0,87	3,65	141,96	115,68	0,81	1,72	6,24
936.	154,39	135,22	0,88	2,10	8,02	191,84	123,14	0,64	2,85	16,80
937.	182,88	99,57	0,54	1,37	6,39	108,83	81,00	0,74	2,02	9,61
938.	234,85	147,90	0,63	2,60	14,32	138,08	113,25	0,82	2,74	13,42
939.	81,99	57,95	0,71	1,40	5,98	87,46	110,28	1,26	2,24	8,04
940.	160,57	126,35	0,79	1,52	6,35	122,89	74,46	0,61	1,21	5,30
941.	119,72	67,77	0,57	1,27	5,12	91,42	128,26	1,40	3,32	14,19
942.	119,42	158,25	1,33	2,68	11,32	106,52	74,59	0,70	1,94	8,59
943.	191,78	125,40	0,65	1,64	6,73	249,14	160,55	0,64	1,28	5,54
944.	107,90	75,11	0,70	2,36	15,11	142,86	151,64	1,06	3,06	19,07
945.	183,86	125,36	0,68	1,93	8,20	277,44	151,25	0,55	1,10	4,55
946.	125,42	72,15	0,58	0,65	3,39	210,31	148,88	0,71	0,71	3,28
947.	85,67	75,21	0,88	2,14	8,97	98,66	164,12	1,66	2,62	10,03
948.	129,77	100,57	0,77	3,58	24,88	91,17	94,21	1,03	3,68	22,79

inst.	Distribution of CVP intermediate effort					Distribution of CIBI intermediate effort				
	n.	$\mu$	$\sigma$	CV	skew	kurt	$\mu$	$\sigma$	CV	skew
949.	192,41	107,03	0,56	0,70	3,73	334,88	217,85	0,65	0,81	3,69
950.	201,15	105,37	0,52	1,31	5,59	140,84	150,12	1,07	3,10	14,31
951.	293,46	168,80	0,58	2,67	18,08	158,99	104,43	0,66	2,52	13,68
952.	104,03	95,54	0,92	5,79	53,25	92,17	198,33	2,15	6,60	59,29
953.	86,74	62,01	0,71	1,62	7,09	96,90	69,59	0,72	1,37	5,31
954.	100,12	68,78	0,69	1,63	7,24	82,83	76,49	0,92	2,22	8,98
955.	73,23	38,48	0,53	2,29	14,10	43,10	39,54	0,92	4,67	33,25
956.	248,48	164,47	0,66	1,99	9,31	326,56	196,95	0,60	0,77	3,79
957.	58,08	57,81	1,00	6,33	53,53	41,54	113,79	2,74	8,31	75,96
958.	82,16	59,53	0,72	1,74	8,01	87,07	125,05	1,44	2,66	11,29
959.	101,94	122,67	1,20	2,73	10,46	125,00	67,67	0,54	1,33	5,73
960.	121,28	80,60	0,66	2,08	12,04	182,41	167,72	0,92	2,03	11,74
961.	83,13	112,42	1,35	4,79	35,17	104,44	65,82	0,63	2,21	12,24
962.	167,58	155,82	0,93	2,57	11,52	210,72	116,45	0,55	0,92	3,49
963.	143,00	123,83	0,87	1,90	8,89	102,16	64,68	0,63	1,57	7,22
964.	107,01	67,76	0,63	1,23	5,34	101,61	89,42	0,88	1,89	6,93
965.	244,82	157,27	0,64	1,58	7,29	130,23	80,45	0,62	1,38	6,47
966.	158,97	103,37	0,65	1,89	9,54	243,21	173,99	0,72	2,67	17,87
967.	123,34	85,77	0,70	1,93	8,92	200,00	134,73	0,67	2,74	18,95
968.	130,55	63,62	0,49	1,12	5,24	86,96	90,45	1,04	3,84	22,45
969.	122,10	89,00	0,73	1,59	6,43	110,71	70,54	0,64	1,22	5,07
970.	76,58	80,38	1,05	2,27	8,10	94,42	52,32	0,55	1,01	4,38
971.	141,04	141,13	1,00	3,13	15,62	200,95	128,26	0,64	1,77	7,75
972.	146,19	85,41	0,58	0,67	3,87	198,67	187,92	0,95	1,32	4,45
973.	192,67	140,46	0,73	1,85	9,29	189,13	113,34	0,60	1,48	5,96
974.	119,65	73,27	0,61	0,98	4,49	145,13	132,40	0,91	1,82	6,81
975.	97,36	64,33	0,66	1,96	10,19	81,53	91,68	1,12	3,17	17,28
976.	169,78	96,10	0,57	1,98	10,29	83,38	51,16	0,61	1,56	8,72
977.	144,71	77,97	0,54	1,21	4,70	112,72	145,99	1,30	3,08	12,85
978.	193,17	118,49	0,61	1,36	5,24	154,27	103,46	0,67	1,14	4,28
979.	85,34	158,30	1,85	4,84	33,89	101,71	80,22	0,79	3,56	23,42
980.	184,18	101,60	0,55	1,17	5,81	167,95	169,68	1,01	1,91	6,26
981.	155,18	89,94	0,58	1,37	5,11	112,51	155,78	1,38	3,38	14,95
982.	168,46	96,19	0,57	1,34	7,14	309,38	180,22	0,58	1,64	8,61
983.	80,66	60,36	0,75	2,21	11,27	105,98	135,65	1,28	2,32	10,20
984.	95,48	60,36	0,63	1,96	11,26	70,73	65,50	0,93	2,33	8,96
985.	56,64	75,97	1,34	3,23	14,18	68,76	46,01	0,67	1,58	6,23
986.	207,28	173,41	0,84	1,79	8,62	154,98	91,81	0,59	1,37	6,98
987.	205,20	144,52	0,70	1,60	6,03	286,69	185,88	0,65	2,08	11,98
988.	140,95	78,91	0,56	1,01	6,02	245,38	165,11	0,67	1,22	5,83
989.	108,94	50,16	0,46	0,26	3,25	219,27	95,25	0,43	0,46	3,27
990.	222,87	134,77	0,60	1,07	4,95	279,41	176,57	0,63	0,88	3,52
991.	111,88	83,26	0,74	1,11	4,38	80,56	53,96	0,67	1,29	5,73
992.	193,07	143,36	0,74	5,28	51,91	191,53	181,42	0,95	1,29	3,81
993.	149,62	72,30	0,48	0,16	3,02	299,29	135,23	0,45	0,42	3,00
994.	217,27	150,08	0,69	1,35	5,21	242,71	150,59	0,62	1,20	5,32
995.	254,63	128,77	0,51	1,11	4,81	197,79	150,35	0,76	1,66	6,10
996.	211,90	142,59	0,67	2,02	10,44	275,30	181,89	0,66	1,17	5,60
997.	165,94	111,69	0,67	1,02	4,17	257,64	244,95	0,95	1,11	3,75
998.	215,86	146,46	0,68	2,16	12,81	322,78	233,89	0,72	1,32	4,90
999.	100,04	66,54	0,67	2,79	19,83	181,40	130,06	0,72	3,06	22,64
1000.	110,50	100,32	0,91	3,14	16,71	160,97	84,24	0,52	1,04	4,49

## Appendix D: Graphs of Statistical Evaluation

This appendix presents statistical evaluations of several indicative sample instances of the CVP vs CIBI design problem. The simulated effort assessments are graphically represented and referred to 7<sup>th</sup> fully stochastic simulation state in Table 6-4. The parameters of each sample instance have been pooled by the randomly generated sample in Appendix B. The interpretation of each subgraph is analyzed in subsection 6.5.5. The error rate (Er) in upper mid graph and the frequency distributions in the bottom graphs have been assessed based on repeated (Monte Carlo) simulations. The rest graphs are referred to indicative outcomes of a single (on-time) simulation.

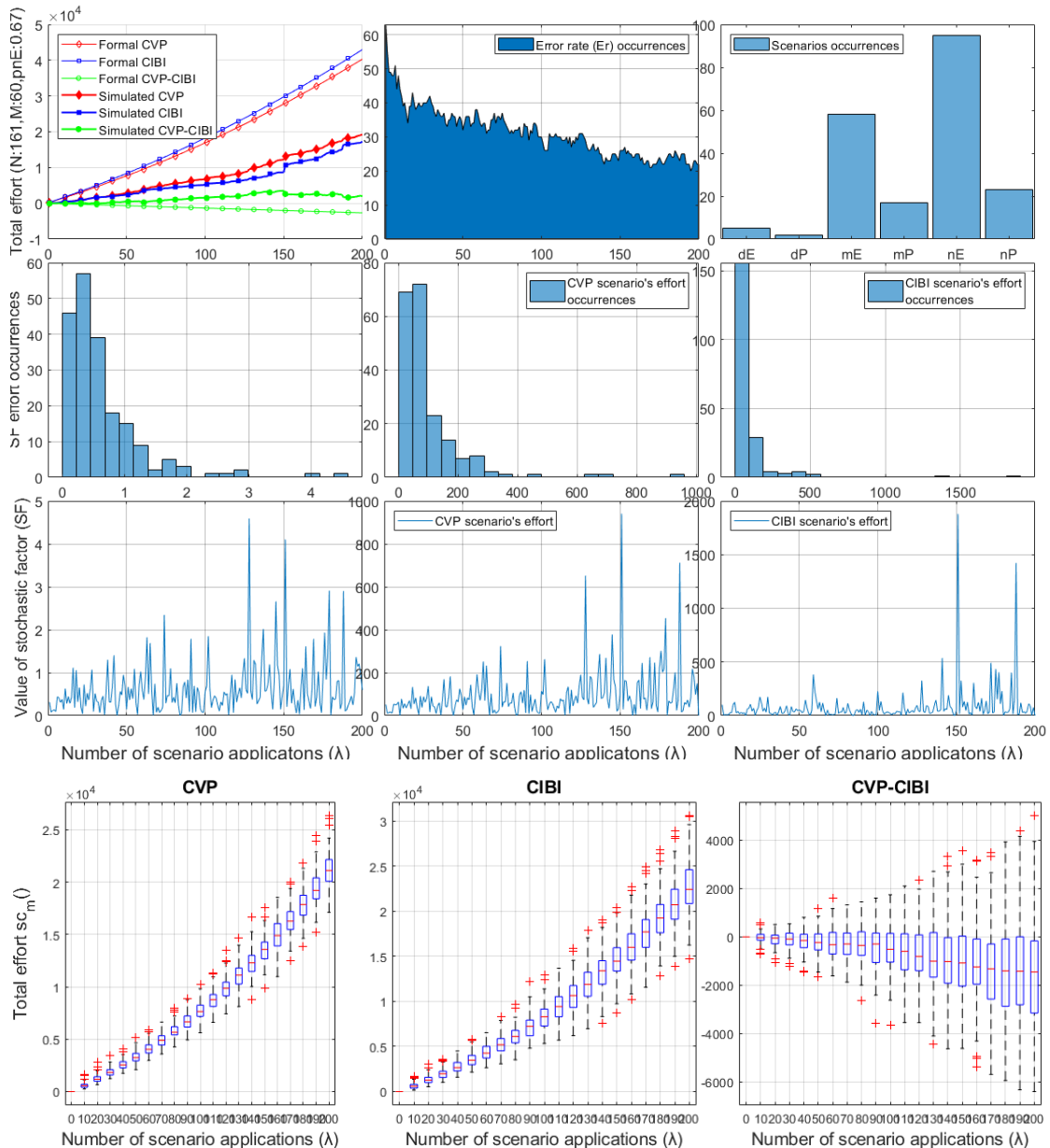


Figure 0-1: Sample instance N. 001 (N=161, M=60,  $p_{nE}=0.67$ ,  $p_{nP}=0.33$ )

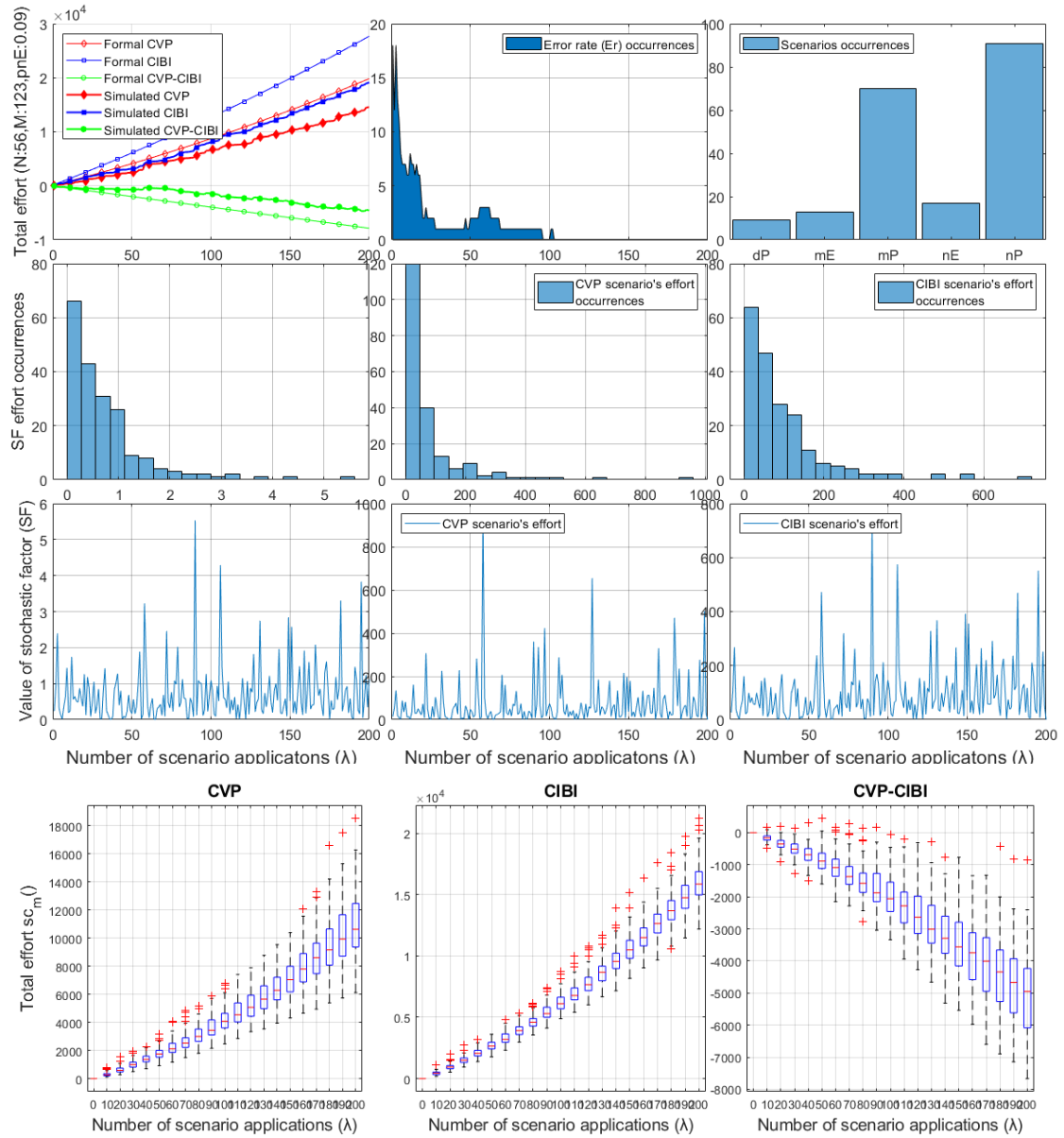


Figure 0-2: Sample instance N. 004 (N=56, M=123,  $p_{nE}=0.09$ ,  $p_{nP}=0.91$ )

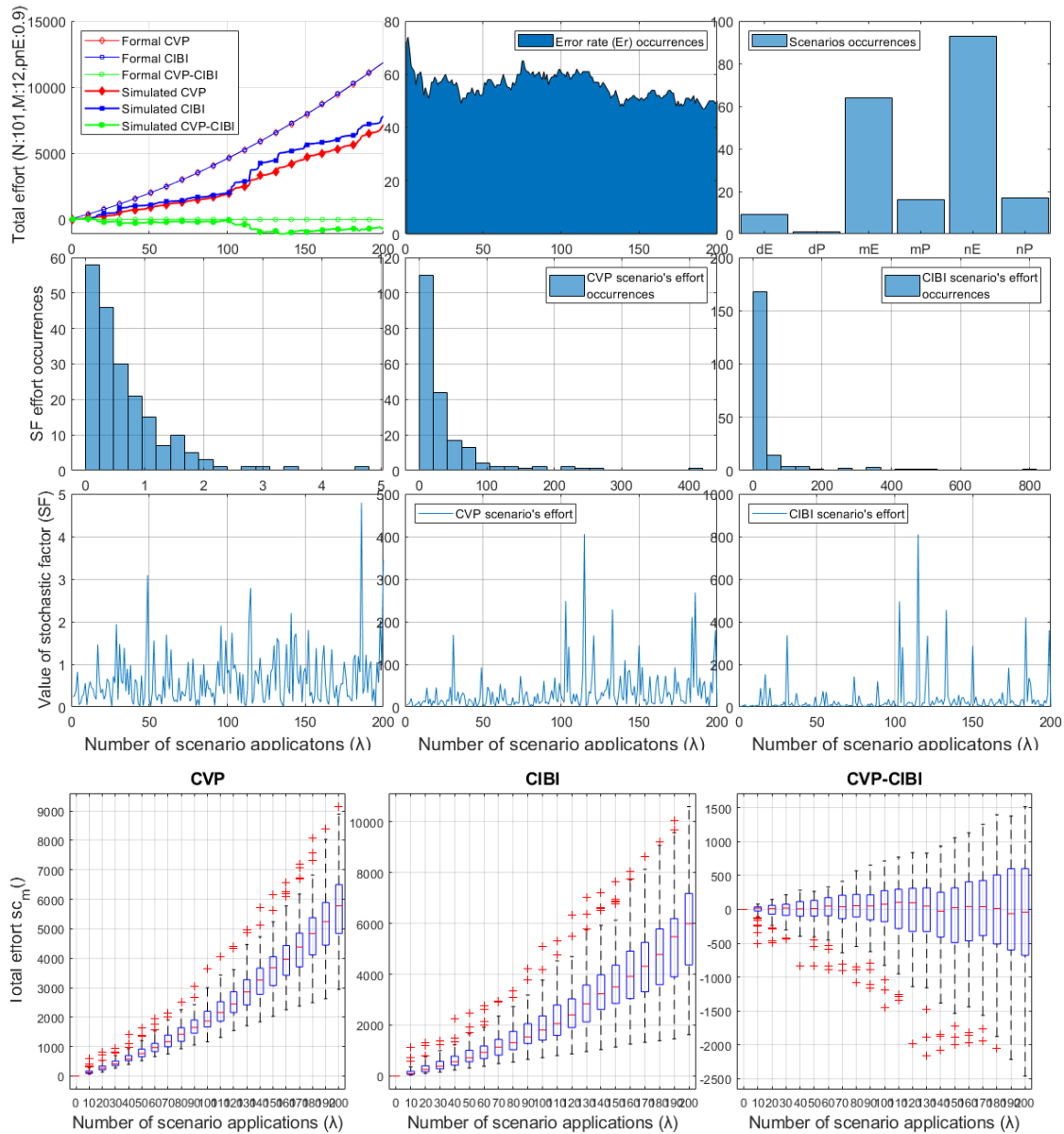


Figure 0-3: Sample instance N. 006 (N=101, M=12, p<sub>nE</sub>=0.90, p<sub>nP</sub>=0.10)

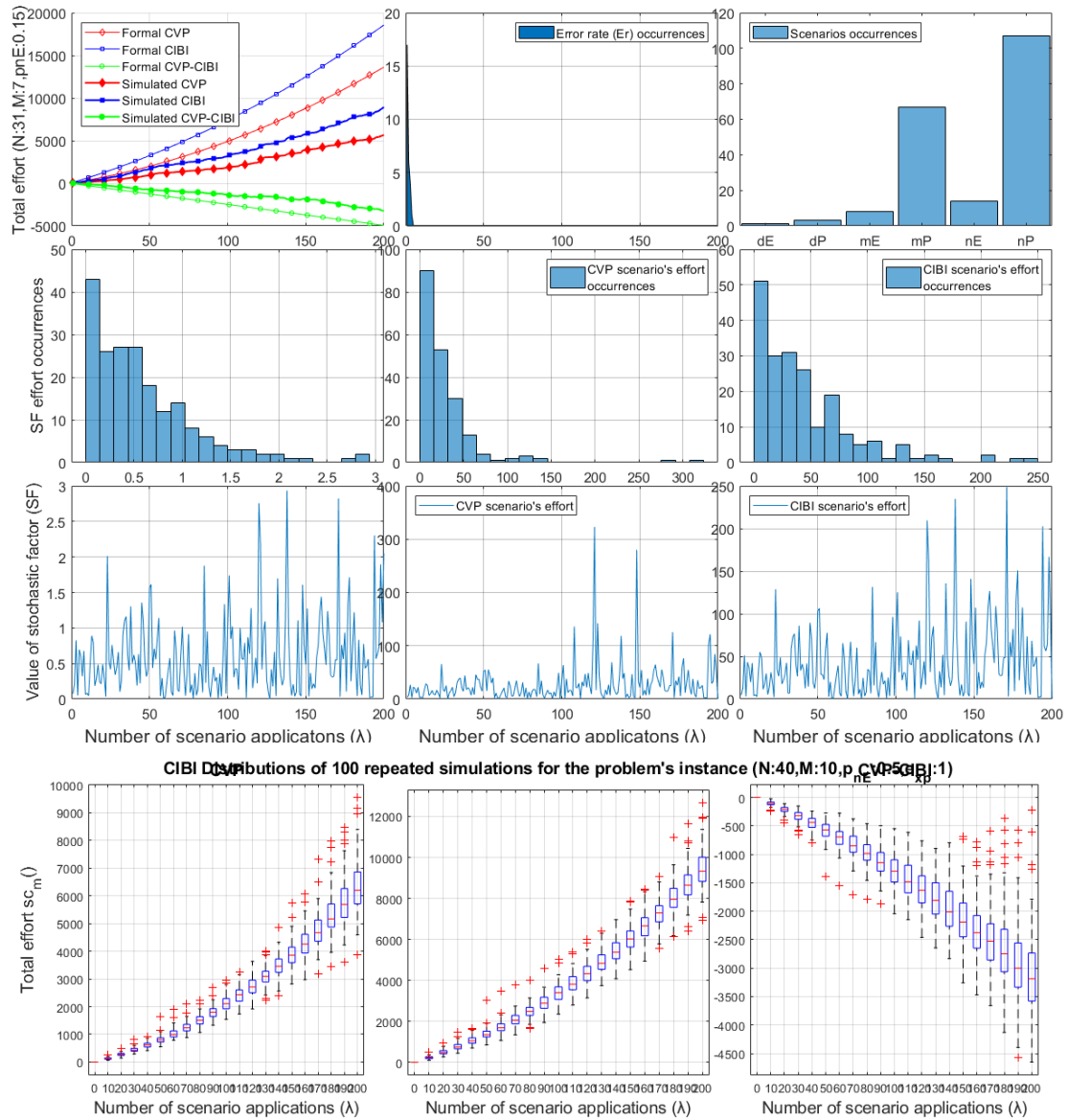


Figure 0-4: Sample instance N. 007 (N=31, M=7, p<sub>nE</sub>=0.15, p<sub>nP</sub>=0.85)

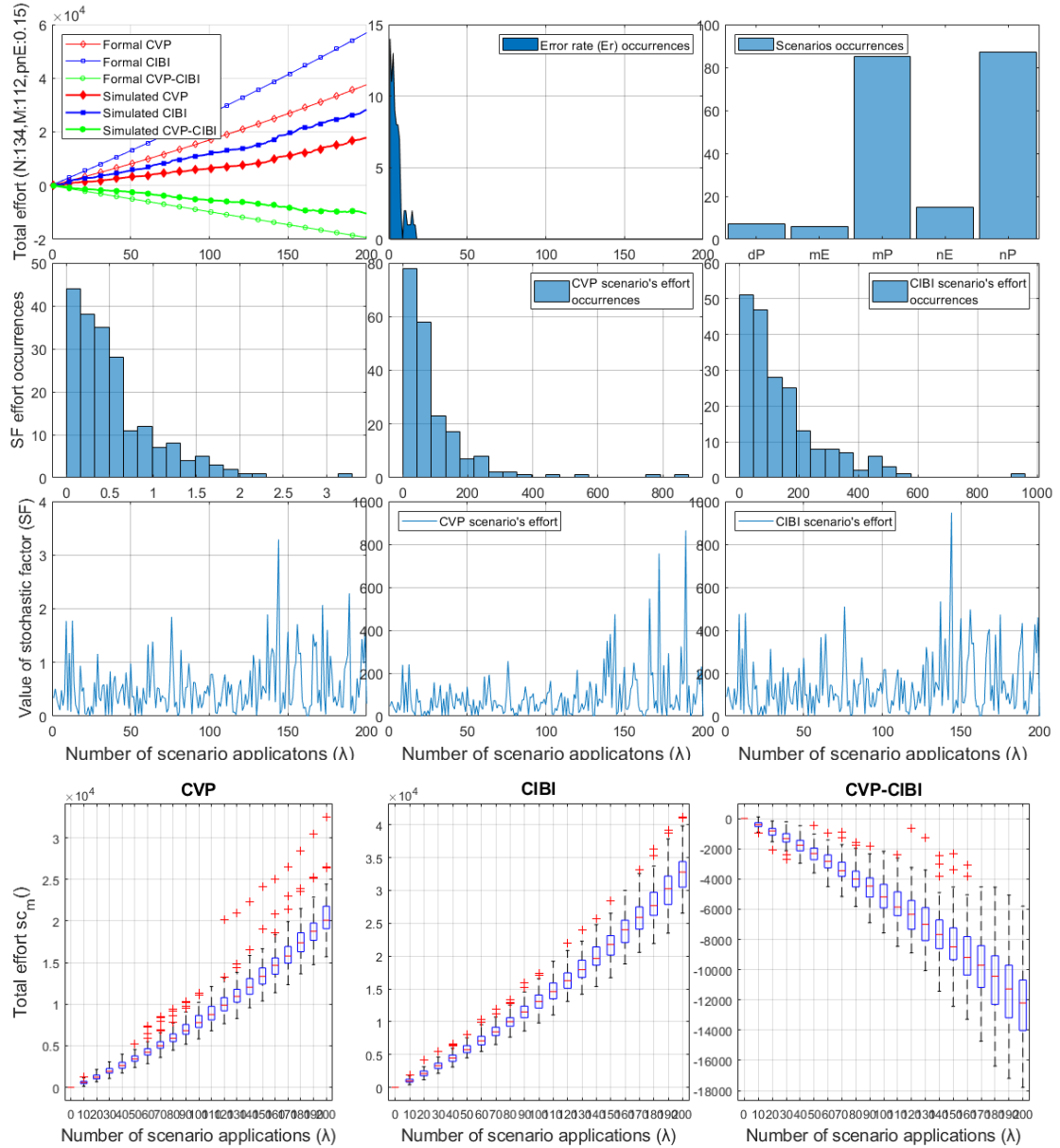


Figure 0-5: Sample instance N. 008 (N=134, M=112,  $p_{nE}=0.15$ ,  $p_{nP}=0.85$ )

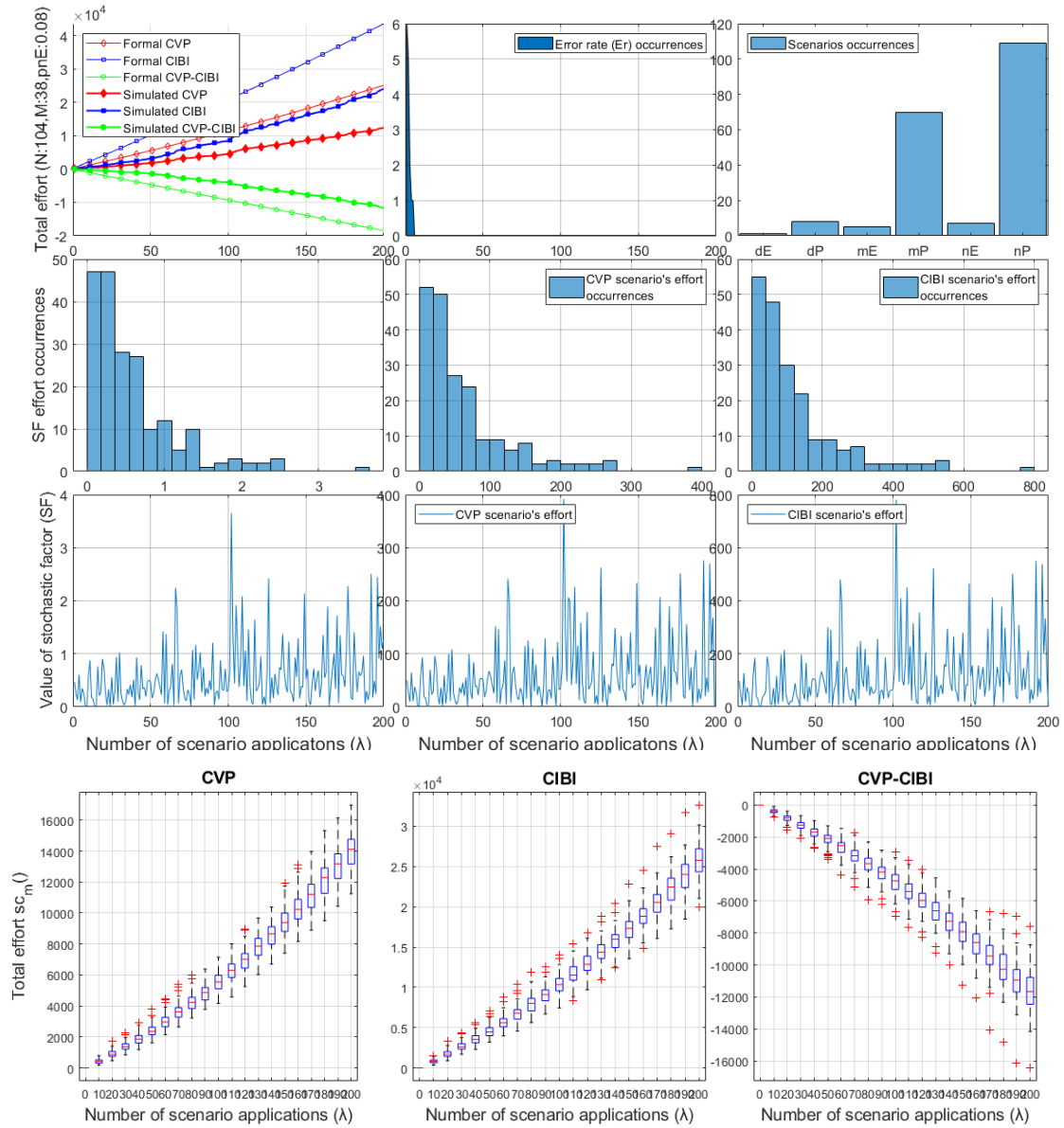


Figure 0-6: Sample instance N. 009 (N=104, M=38,  $p_{nE}=0.08$ ,  $p_{nP}=0.92$ )



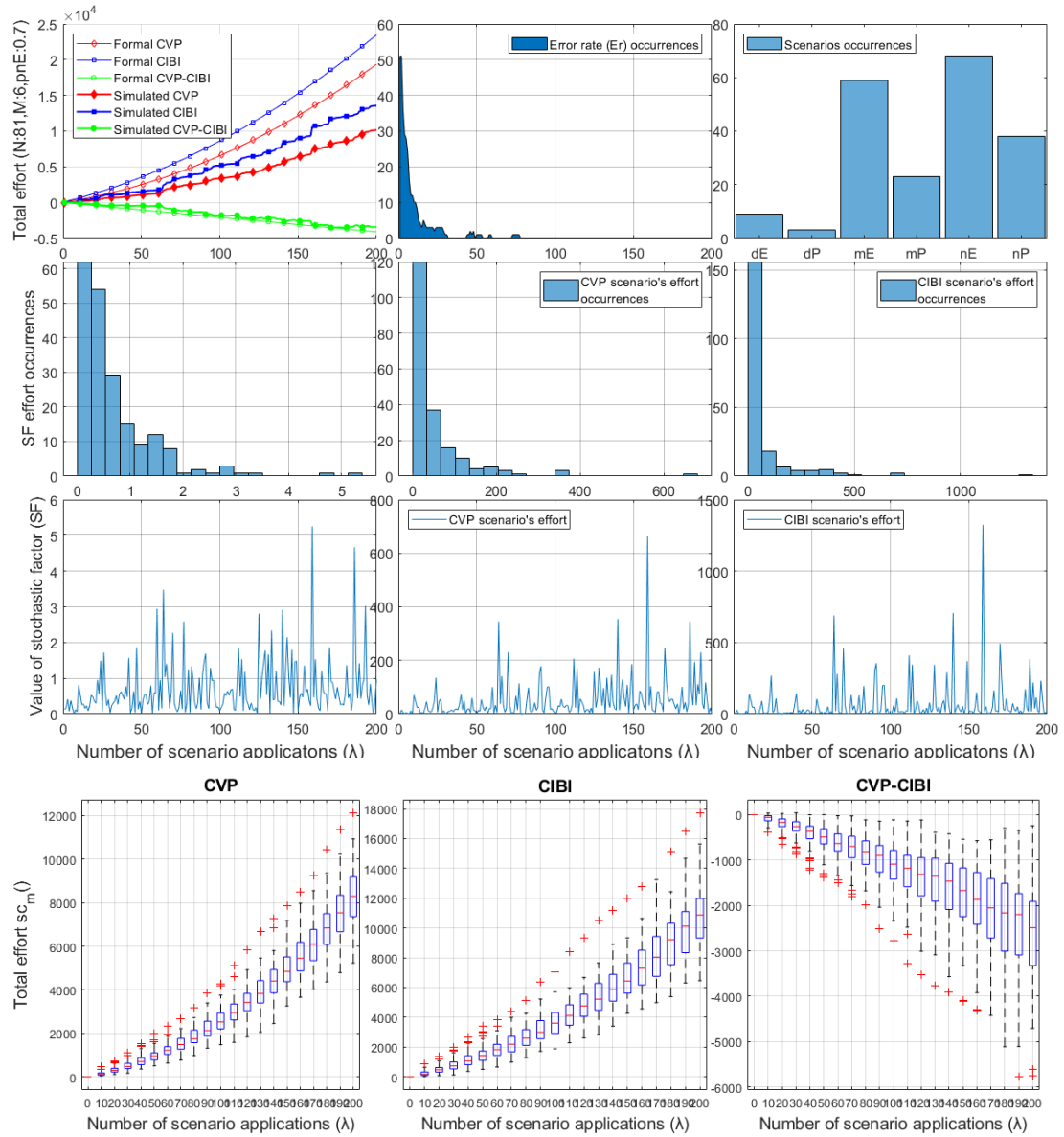


Figure 0-7: Sample instance N. 010 ( $N=81$ ,  $M=6$ ,  $p_{nE}=0.70$ ,  $p_{nP}=0.30$ )

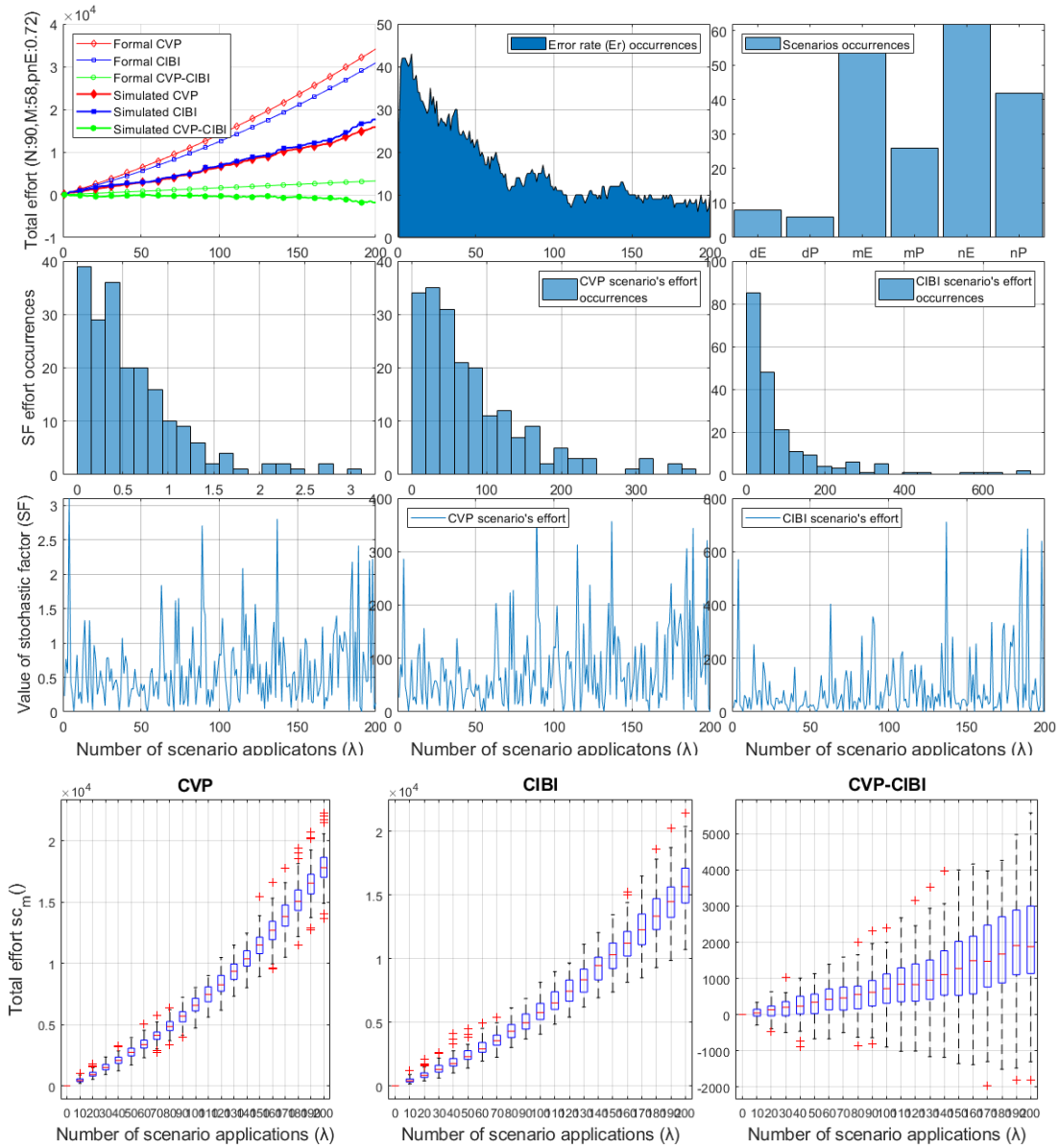


Figure 0-8: Sample instance N. 011 ( $N=90$ ,  $M=58$ ,  $p_{nE}=0.72$ ,  $p_{nP}=0.28$ )

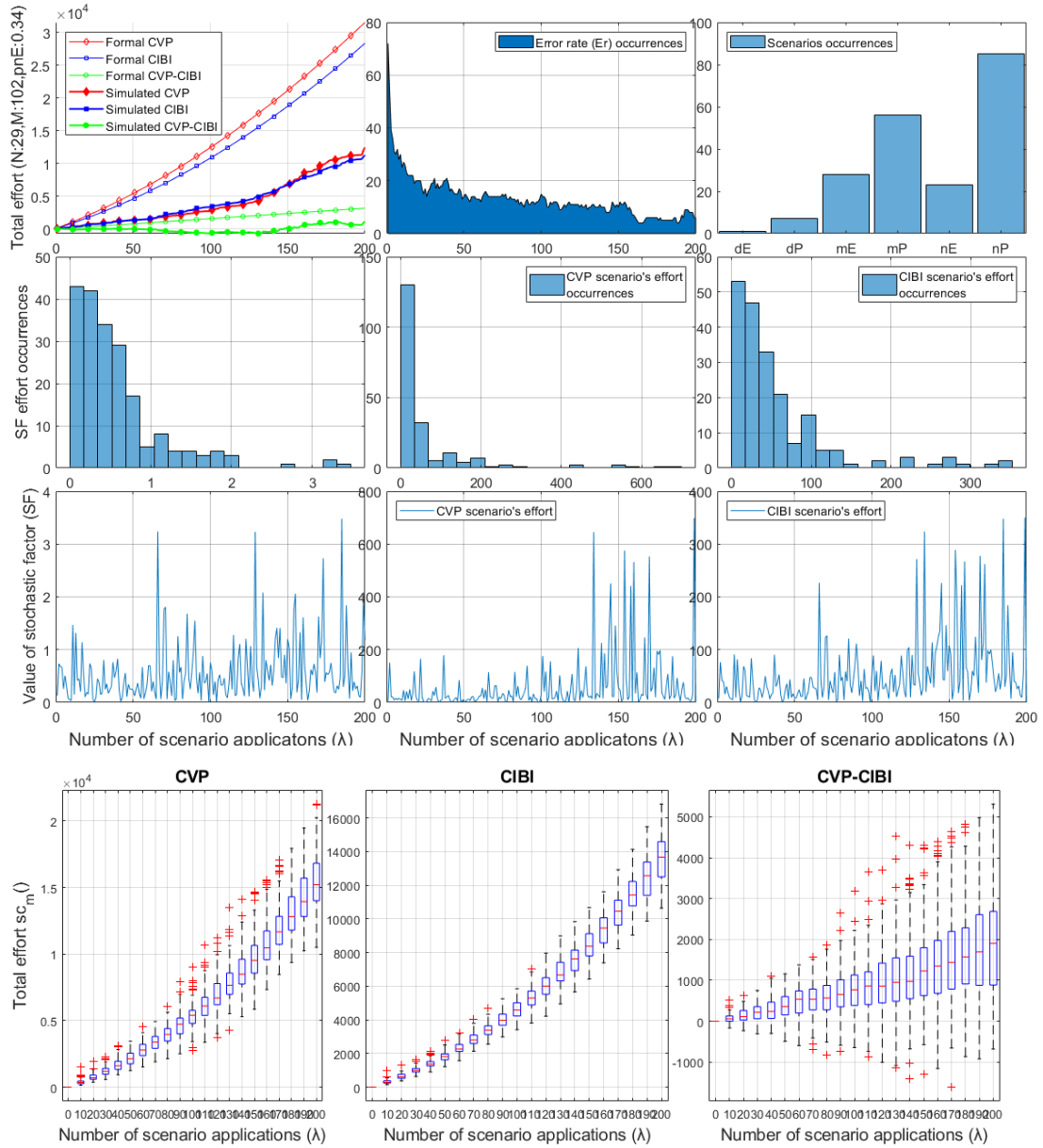


Figure 0-9: Sample instance N. 012 ( $N=29$ ,  $M=102$ ,  $p_{nE}=0.34$ ,  $p_{nP}=0.66$ )

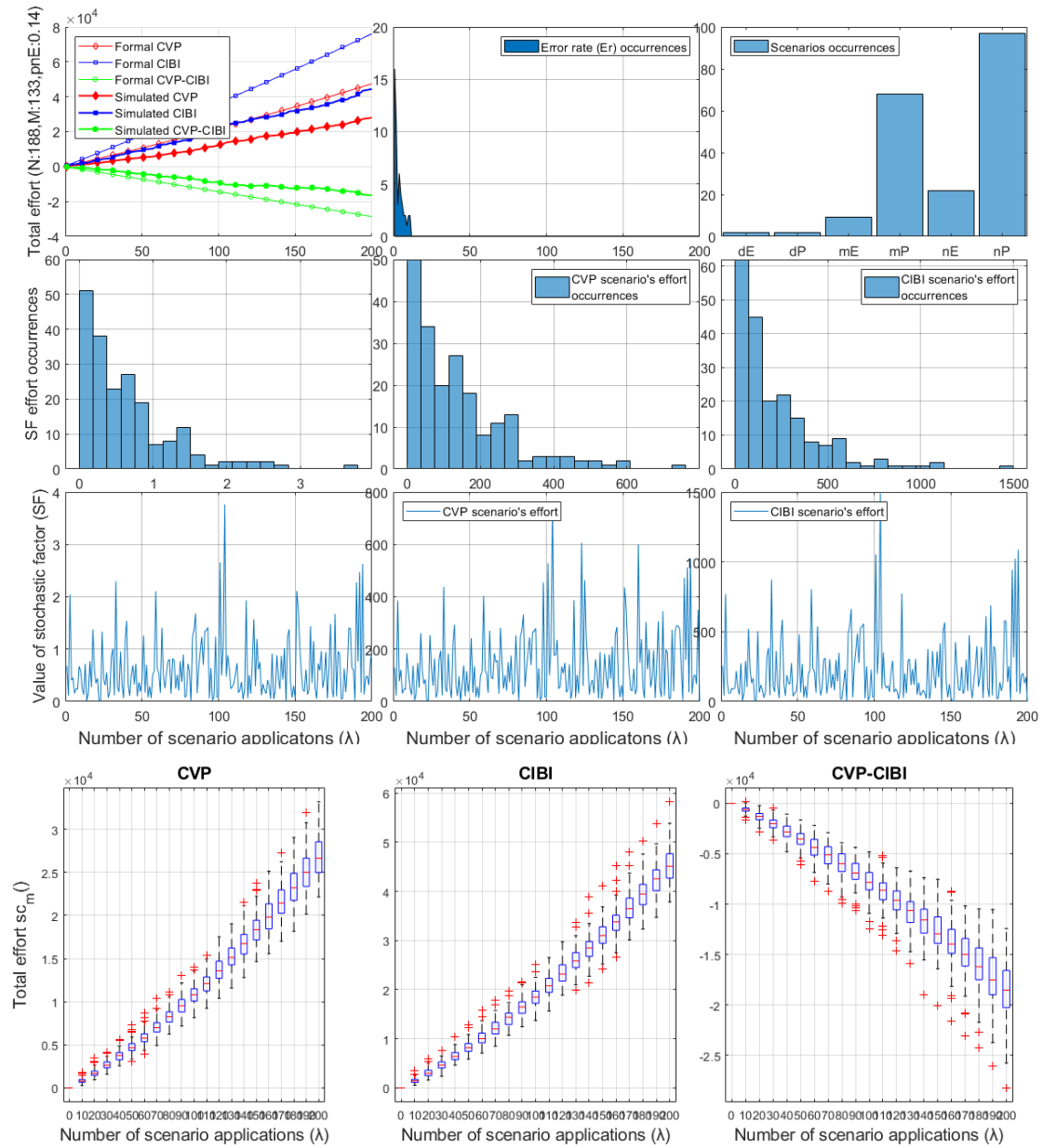


Figure 0-10: Sample instance N. 014 (N=188, M=133,  $p_{nE}=0.14$ ,  $p_{nP}=0.86$ )

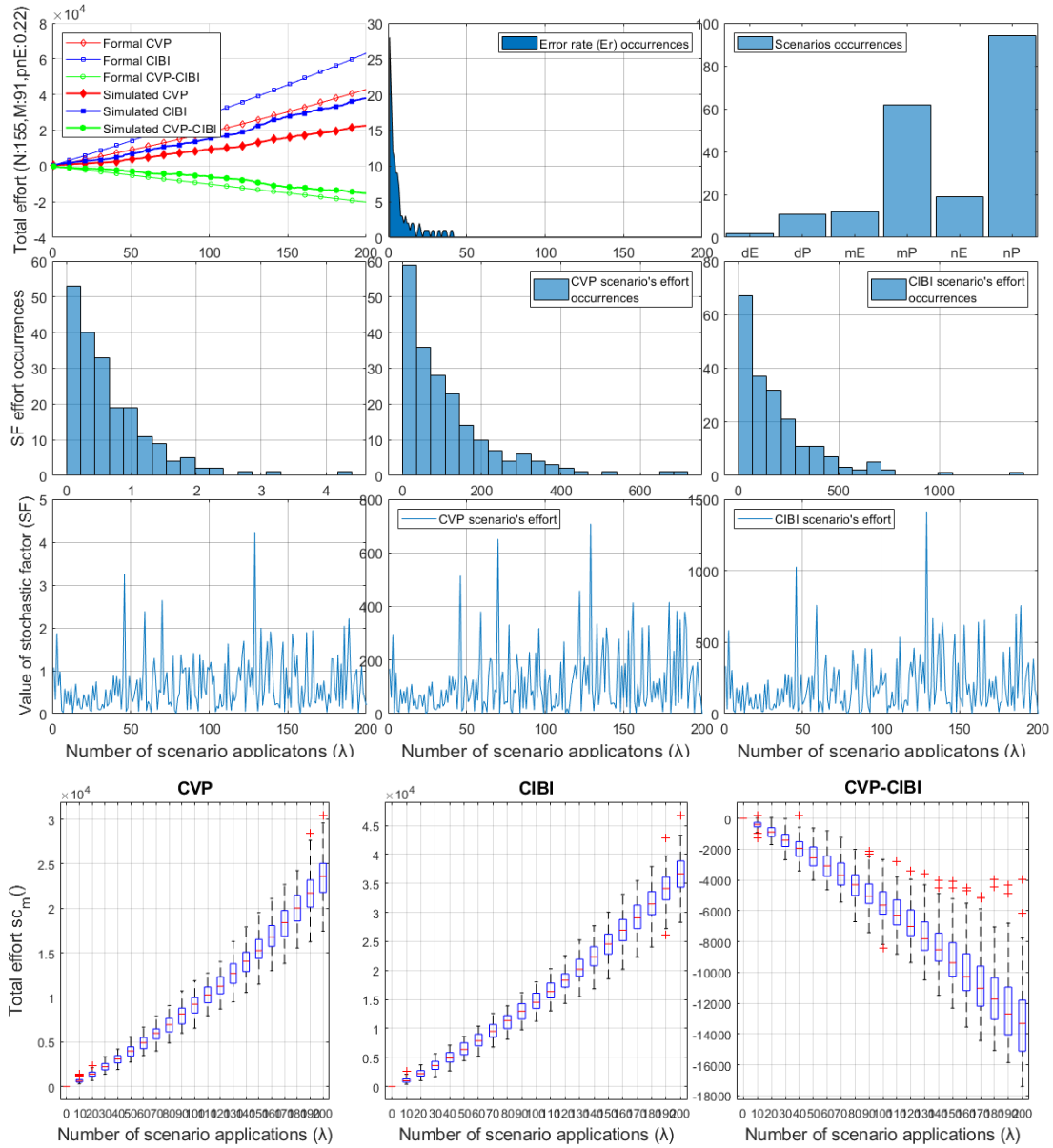


Figure 0-11: Sample instance N. 016 (N=155, M=91, p<sub>nE</sub>=0.22, p<sub>nP</sub>=0.78)

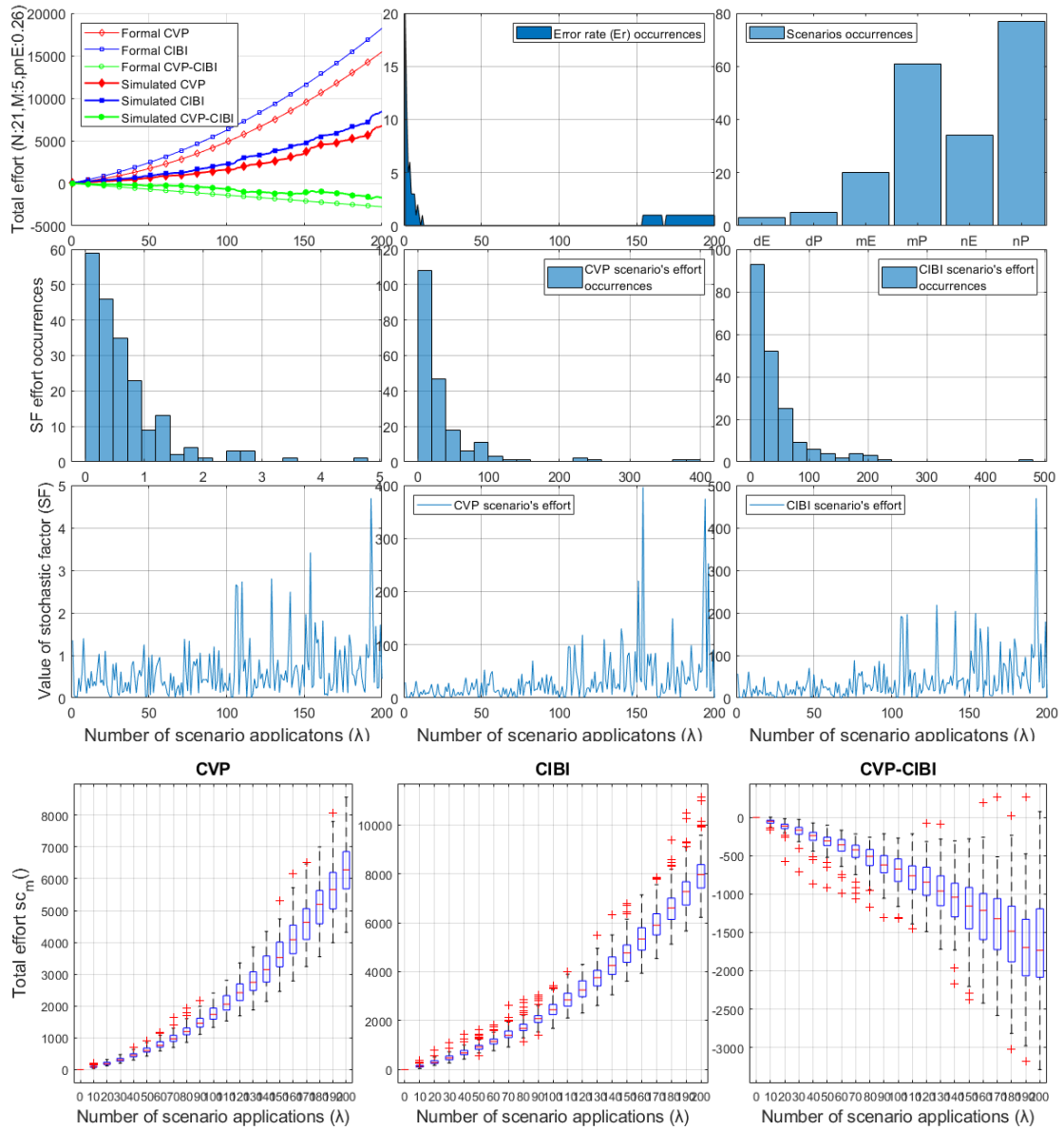


Figure 0-12: Sample instance N. 017 (N=21, M=5,  $p_{nE}=0.26$ ,  $p_{nP}=0.74$ )

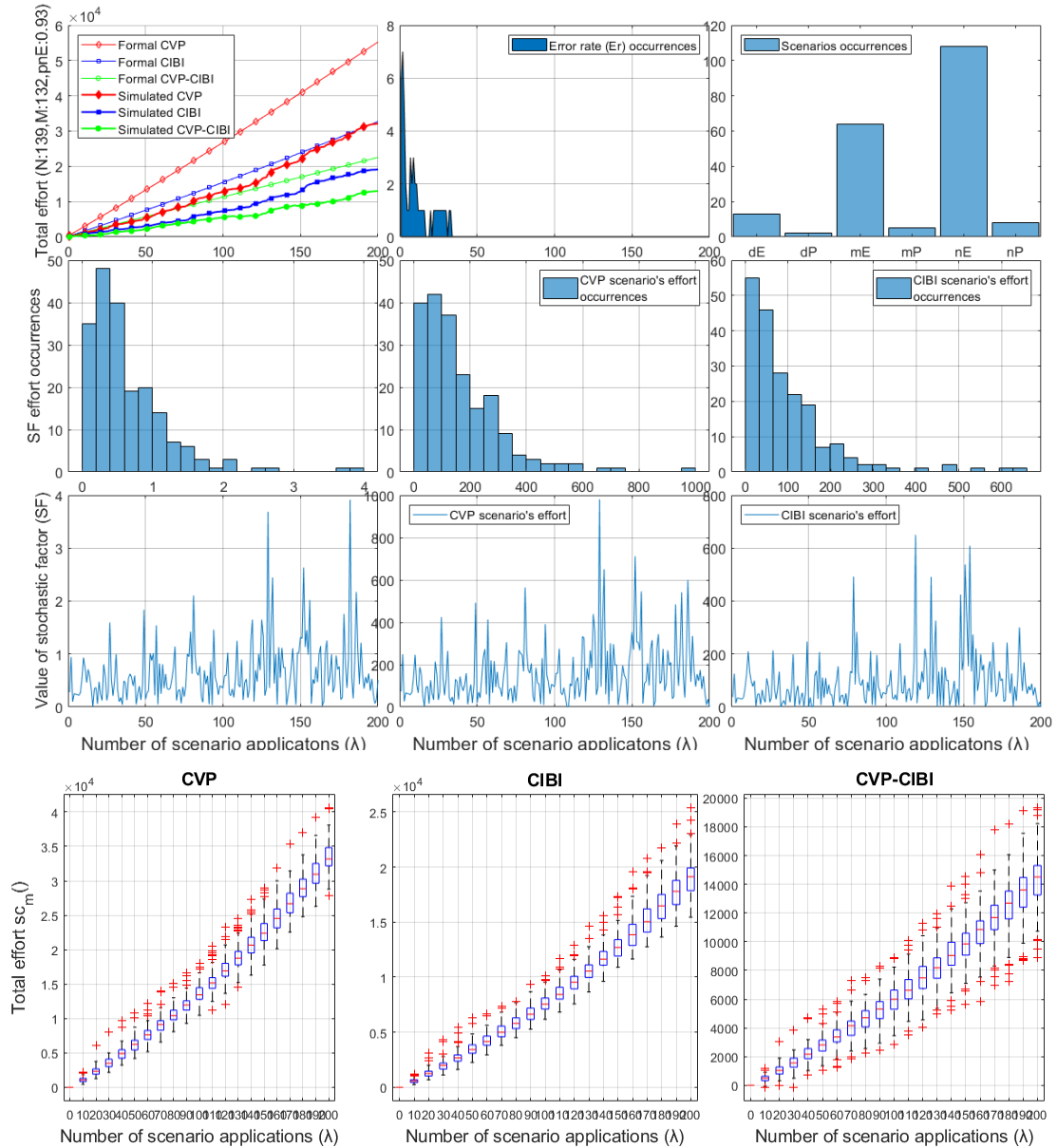


Figure 0-13: Sample instance N. 018 (N=139, M=132,  $p_{nE}=0.93$ ,  $p_{nP}=0.07$ )

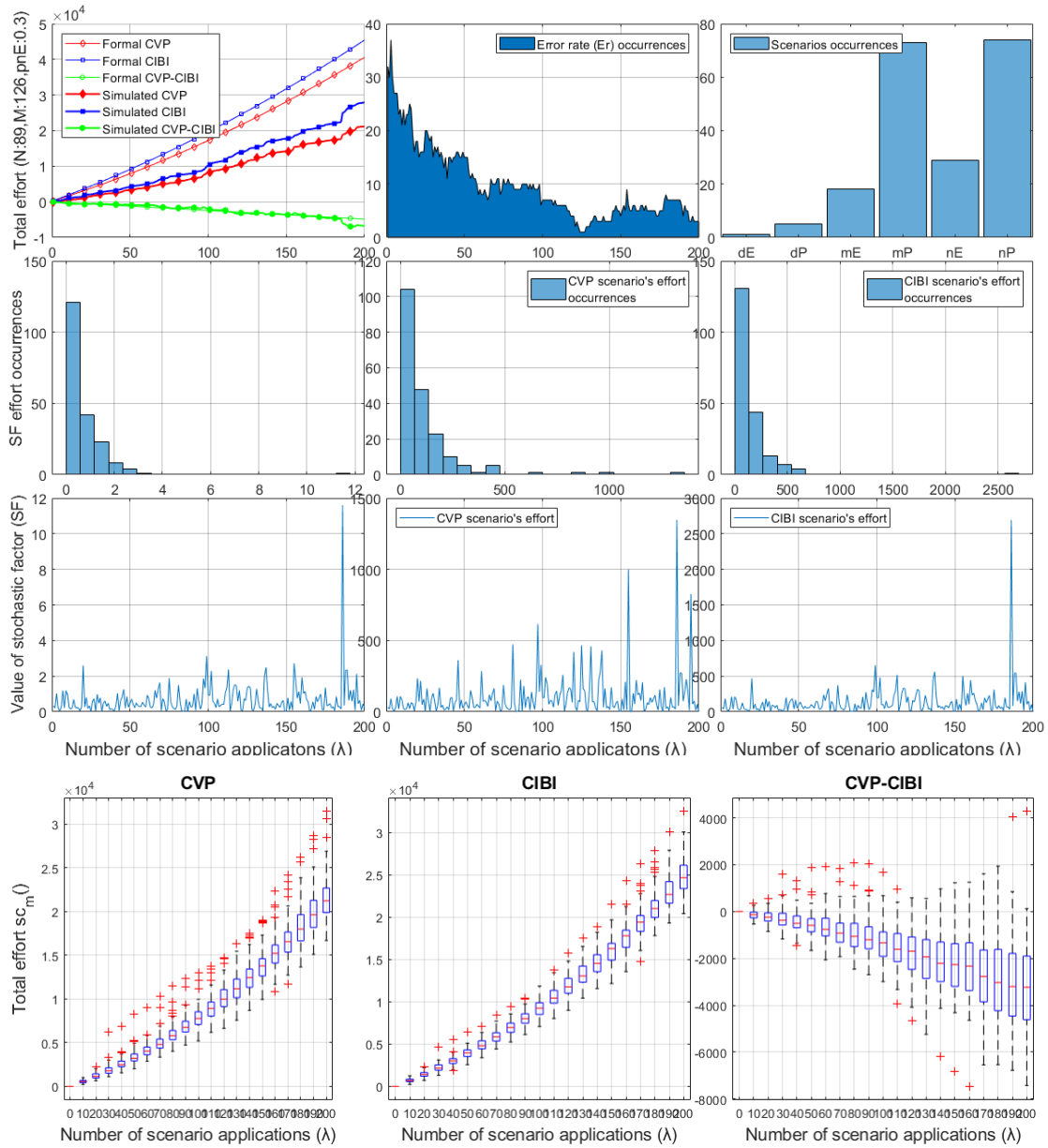


Figure 0-14: Sample instance N. 019 (N=89, M=126,  $p_{nE}=0.30$ ,  $p_{nP}=0.70$ )



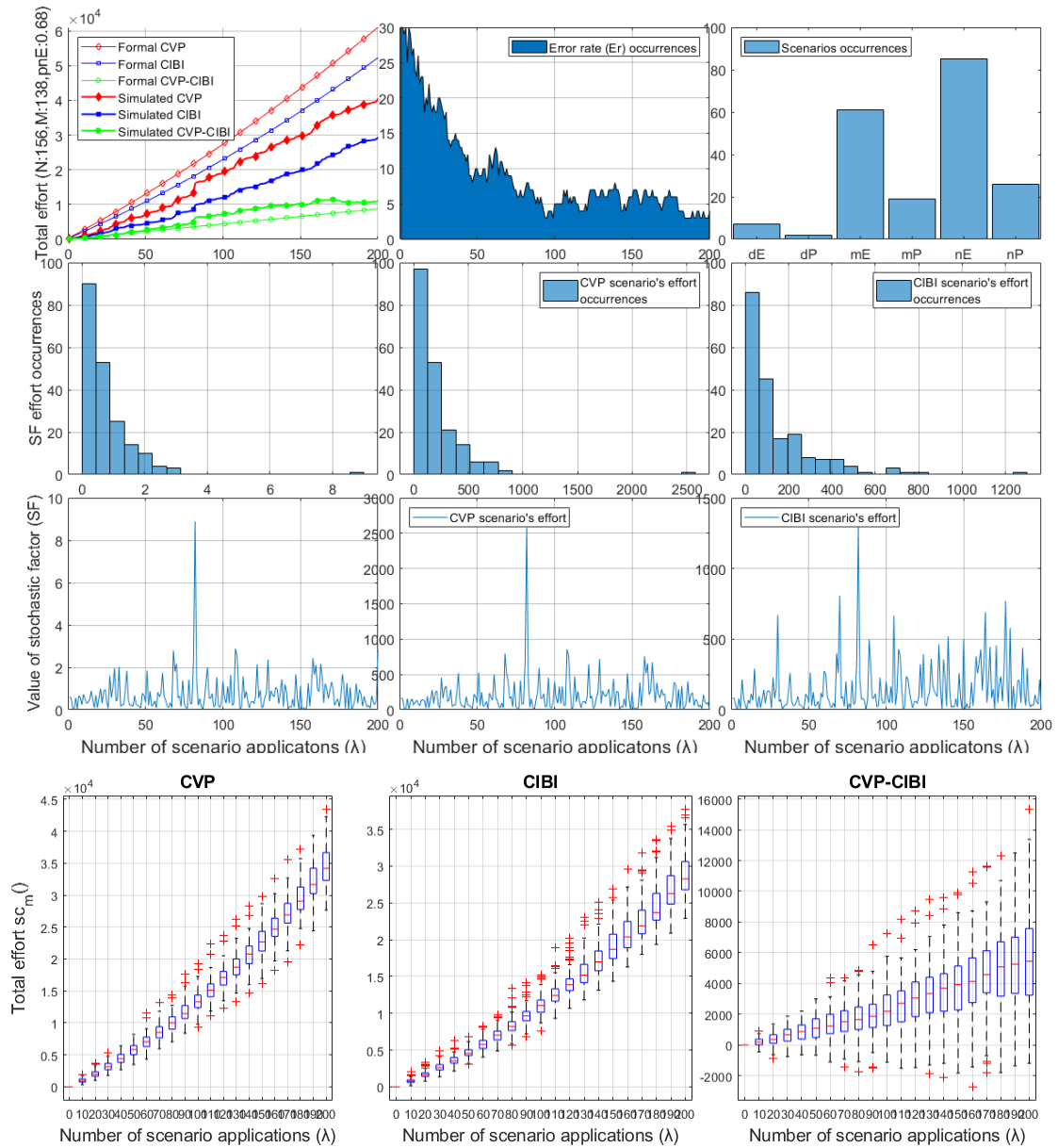


Figure 0-15: Sample instance N. 020 (N=156, M=138, p<sub>nE</sub>=0.68, p<sub>nP</sub>=0.32)

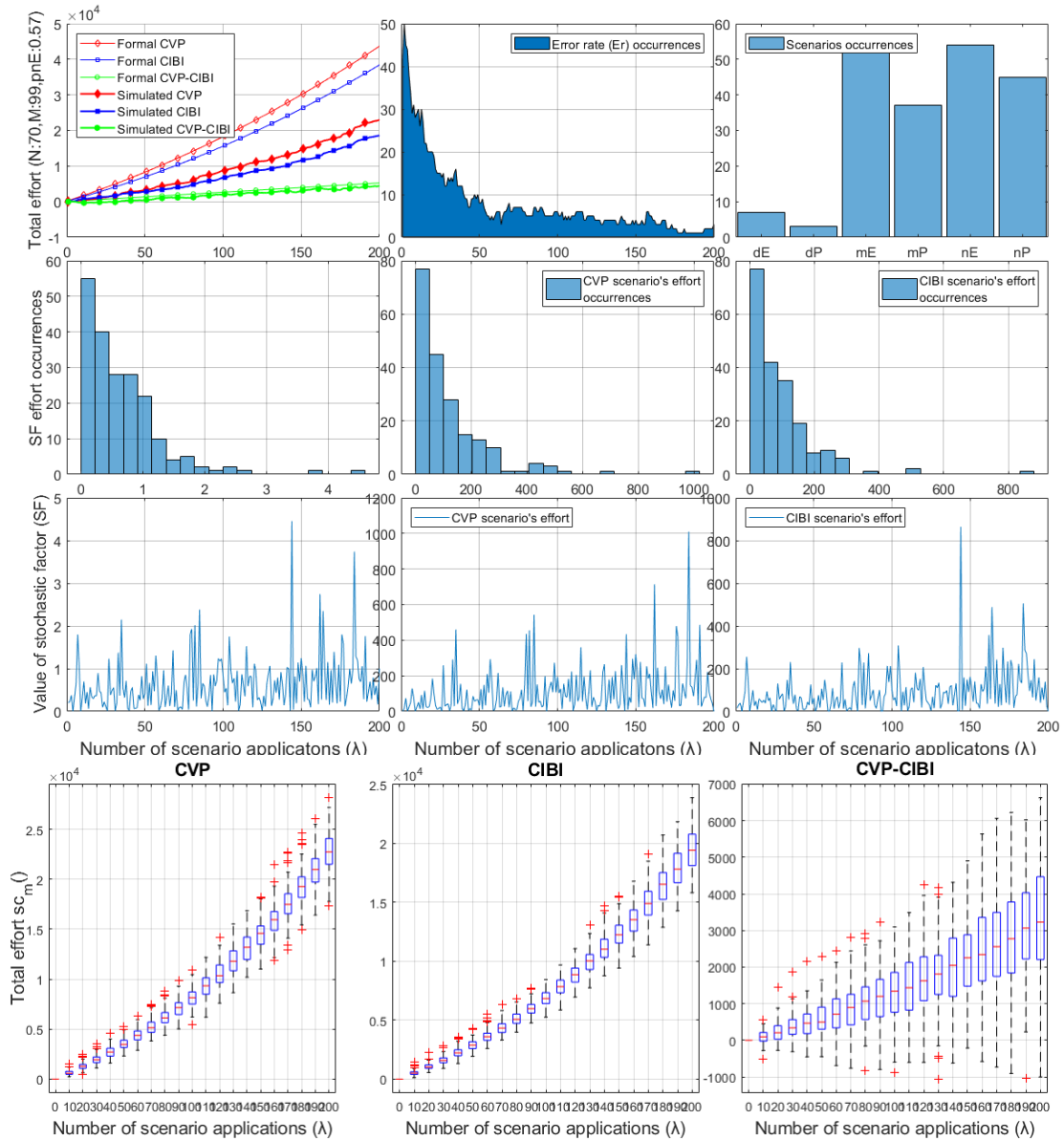


Figure 0-16: Sample instance N. 021 (N=70, M=99,  $p_{nE}=0.57$ ,  $p_{nP}=0.43$ )

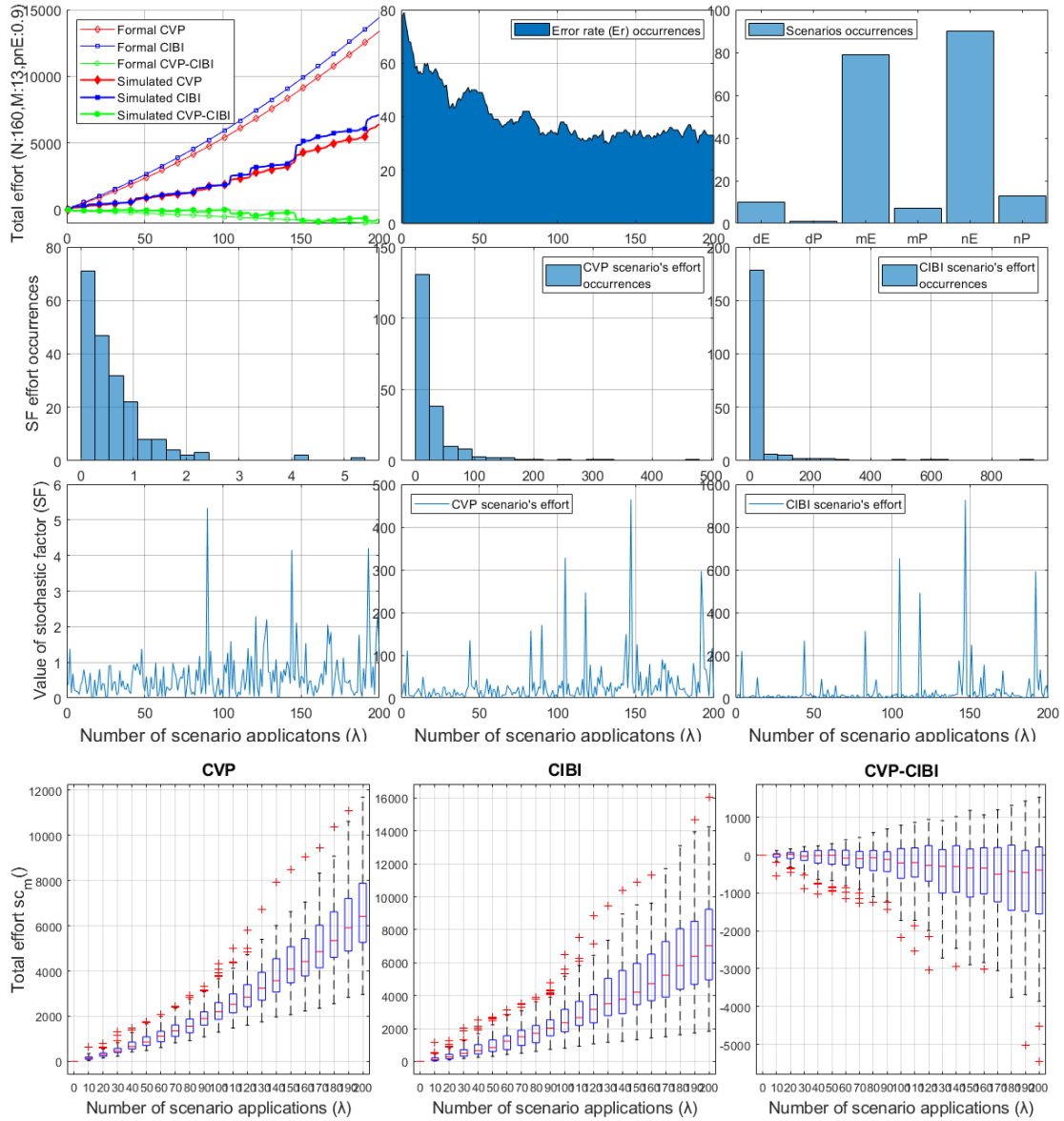


Figure 0-17: Sample instance N. 022 (N=160, M=13,  $p_{nE}=0.90$ ,  $p_{nP}=0.10$ )

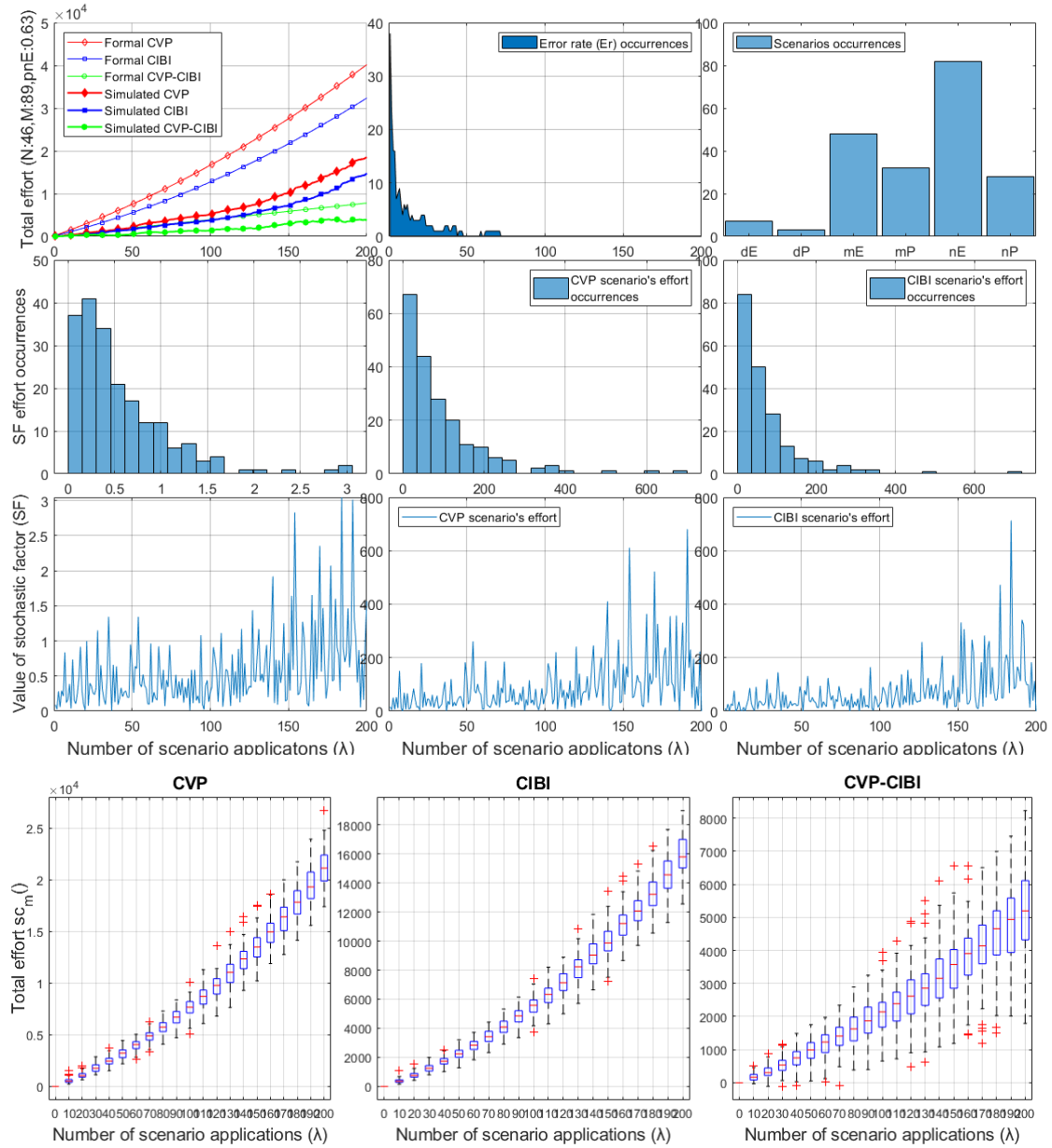


Figure 0-18: Sample instance N. 024 ( $N=46$ ,  $M=89$ ,  $p_{nE}=0.63$ ,  $p_{nP}=0.37$ )

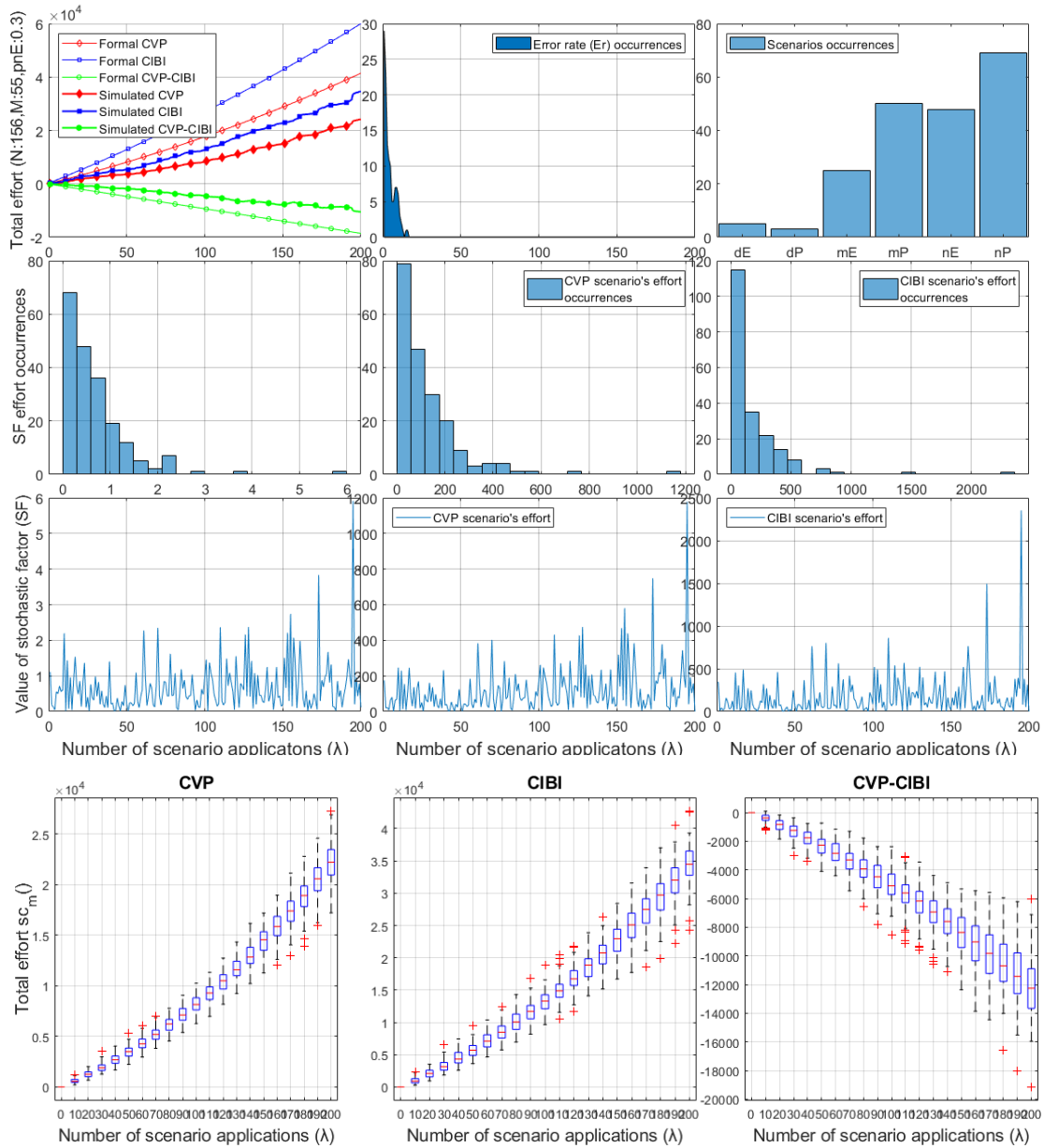


Figure 0-19: Sample instance N. 025 (N=156, M=55,  $p_{nE}=0.30$ ,  $p_{nP}=0.70$ )

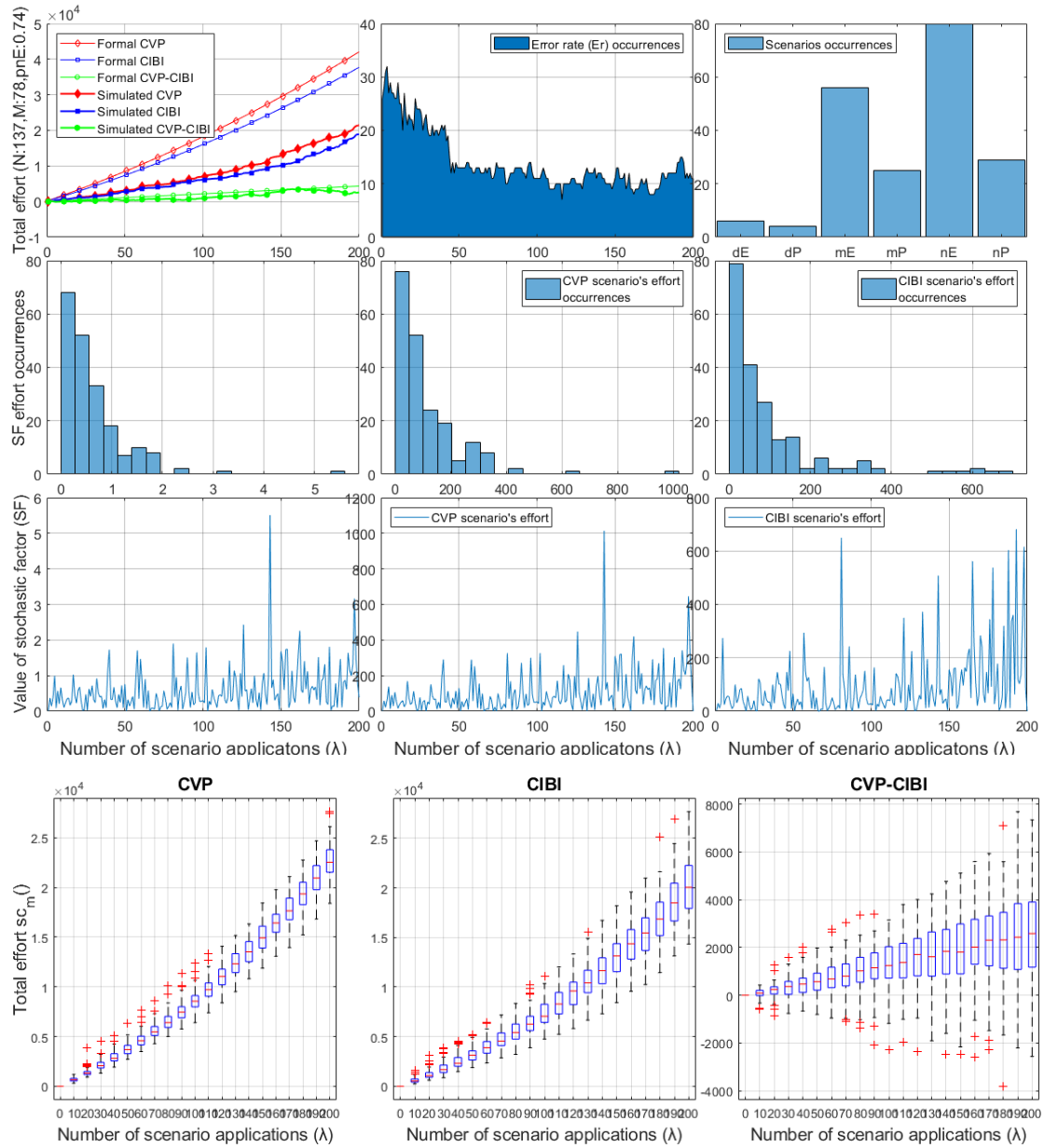


Figure 0-20: Sample instance N. 026 (N=137, M=78,  $p_{nE}=0.74$ ,  $p_{nP}=0.26$ )

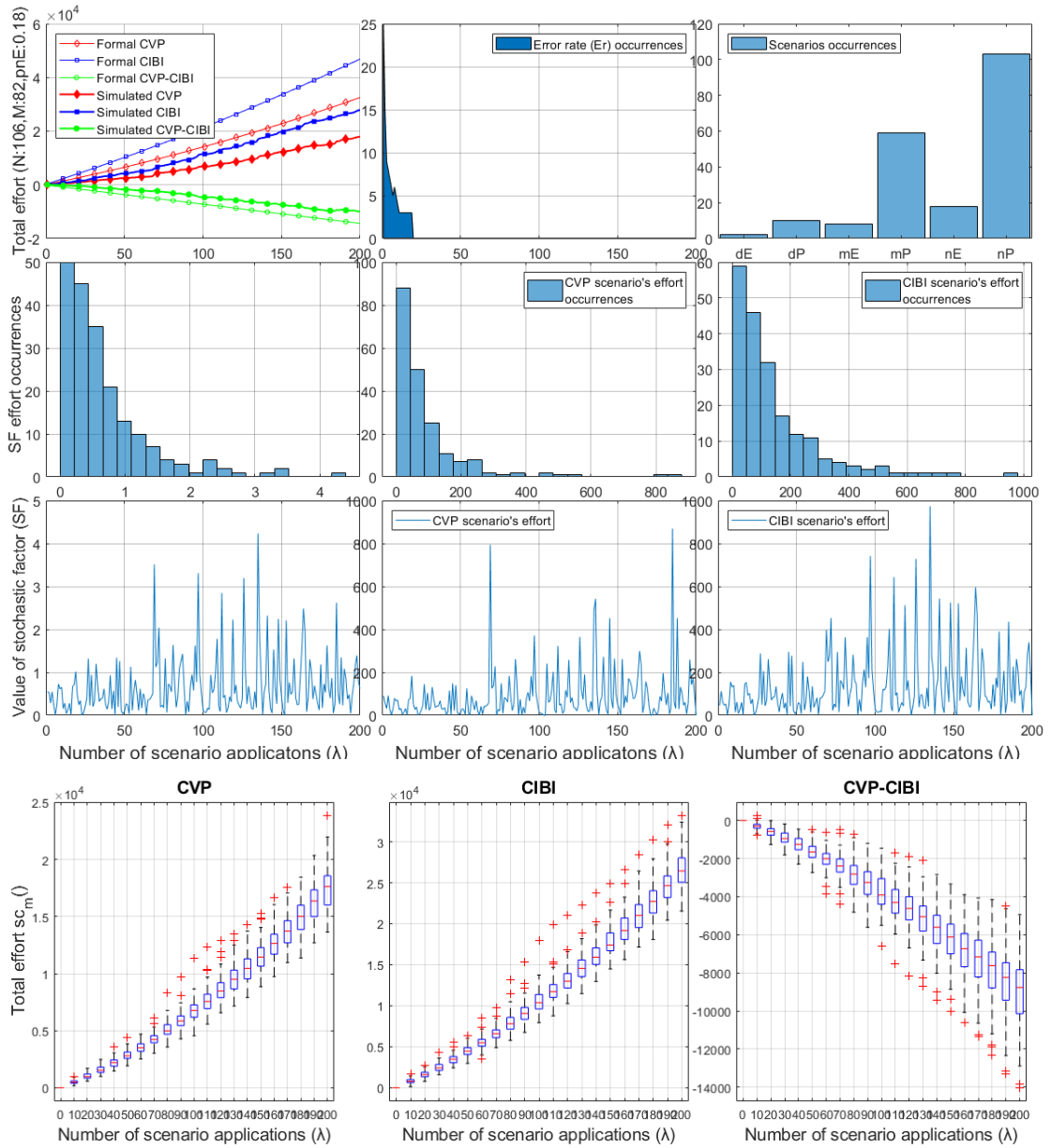


Figure 0-21: Sample instance N. 027 (N=106, M=82,  $p_{nE}=0.18$ ,  $p_{nP}=0.82$ )

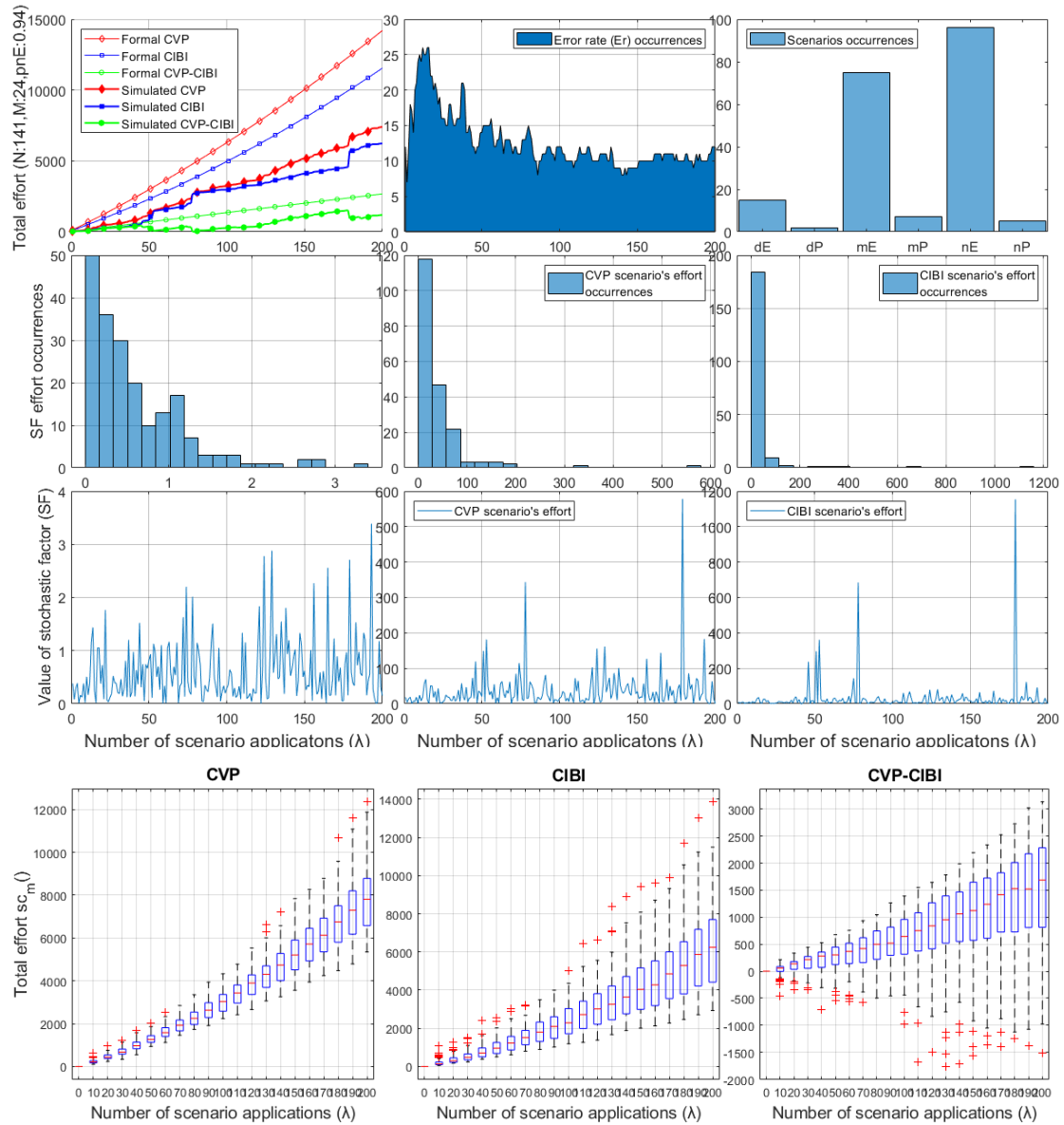


Figure 0-22: Sample instance N. 029 (N=141, M=24,  $p_{nE}=0.94$ ,  $p_{nP}=0.06$ )



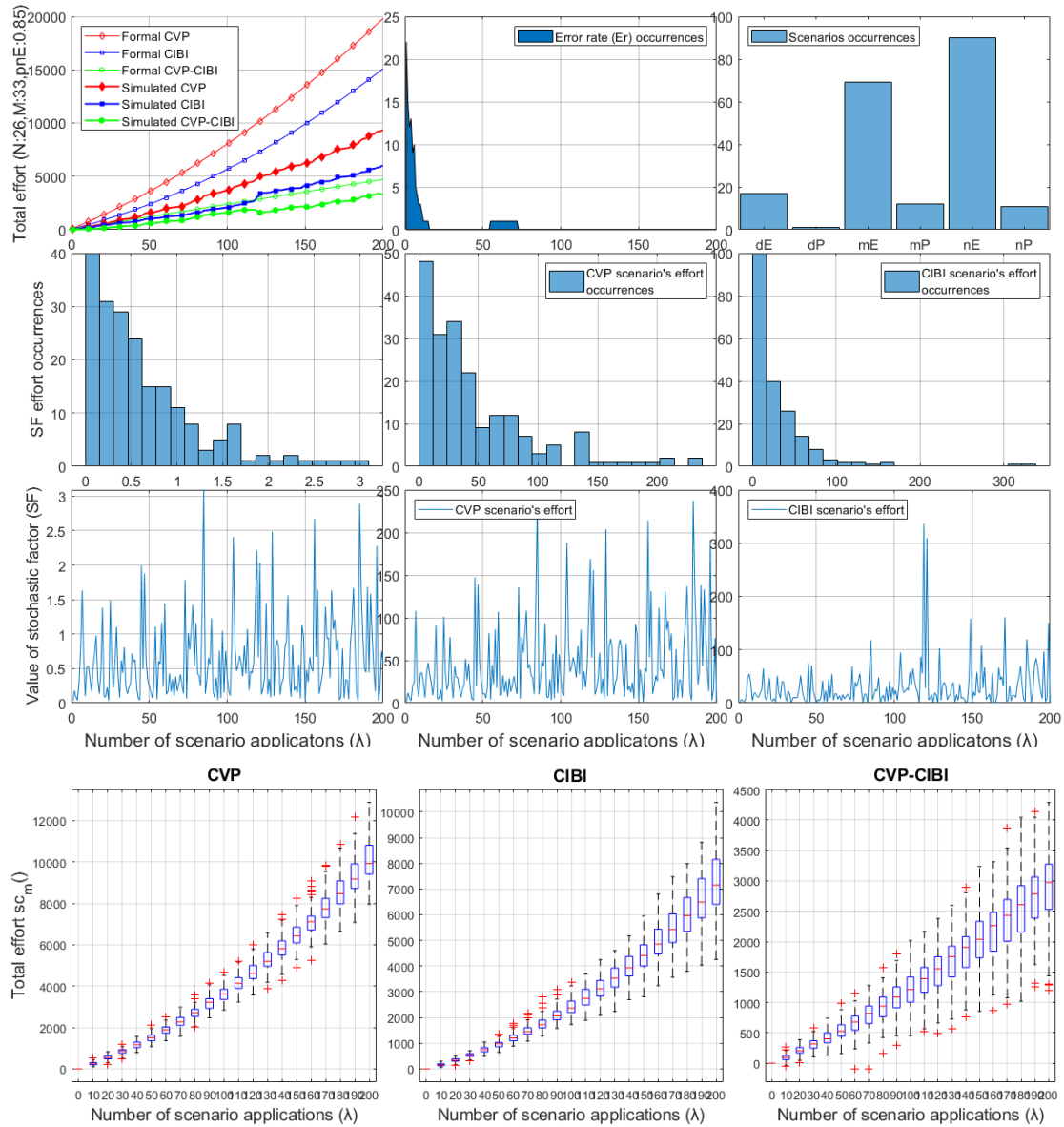


Figure 0-23: Sample instance N. 031 (N=26, M=33,  $p_{nE}=0.85$ ,  $p_{nP}=0.15$ )

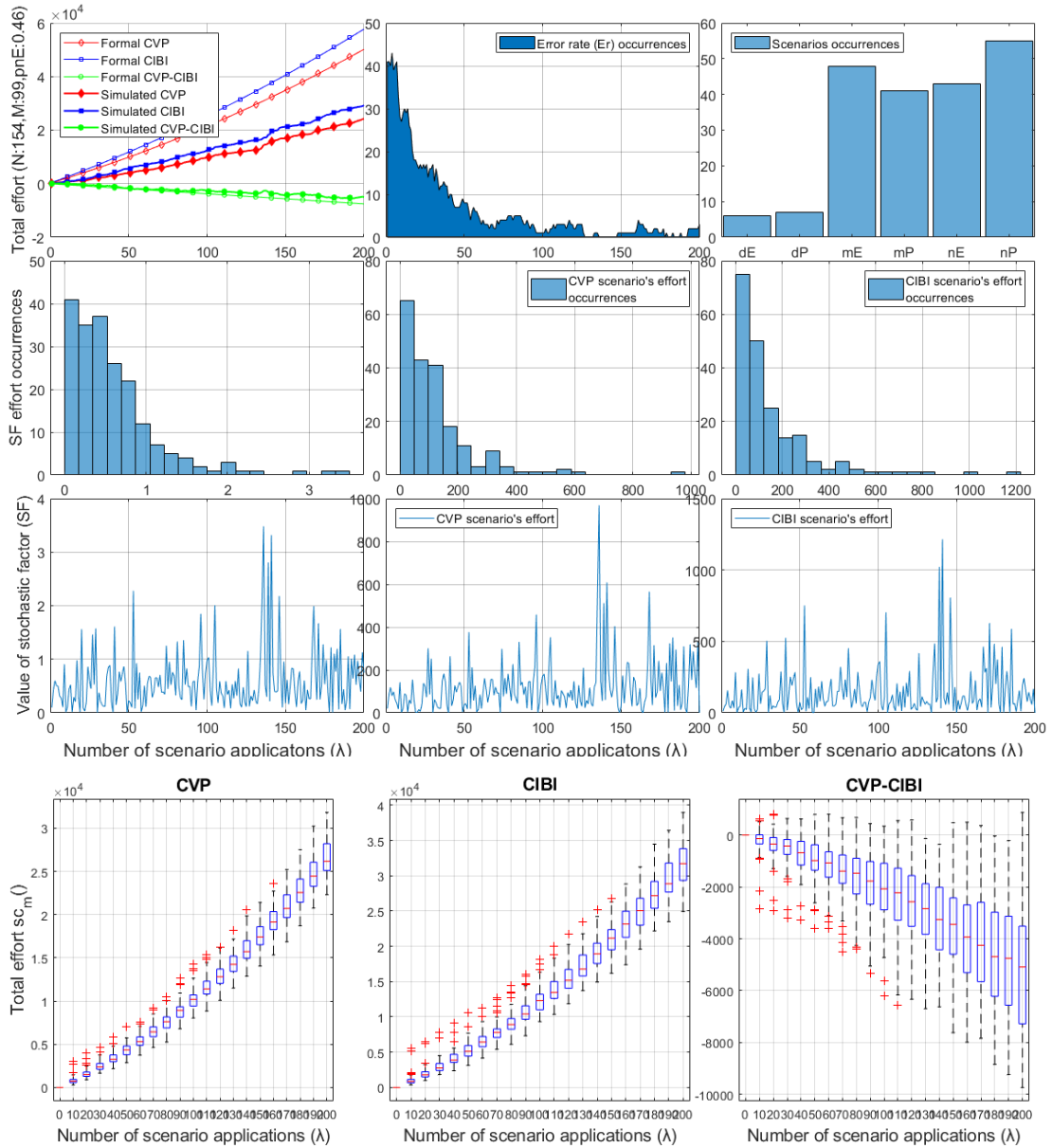


Figure 0-24: Sample instance N. 032 (N=154, M=99,  $p_{nE}=0.46$ ,  $p_{nP}=0.54$ )

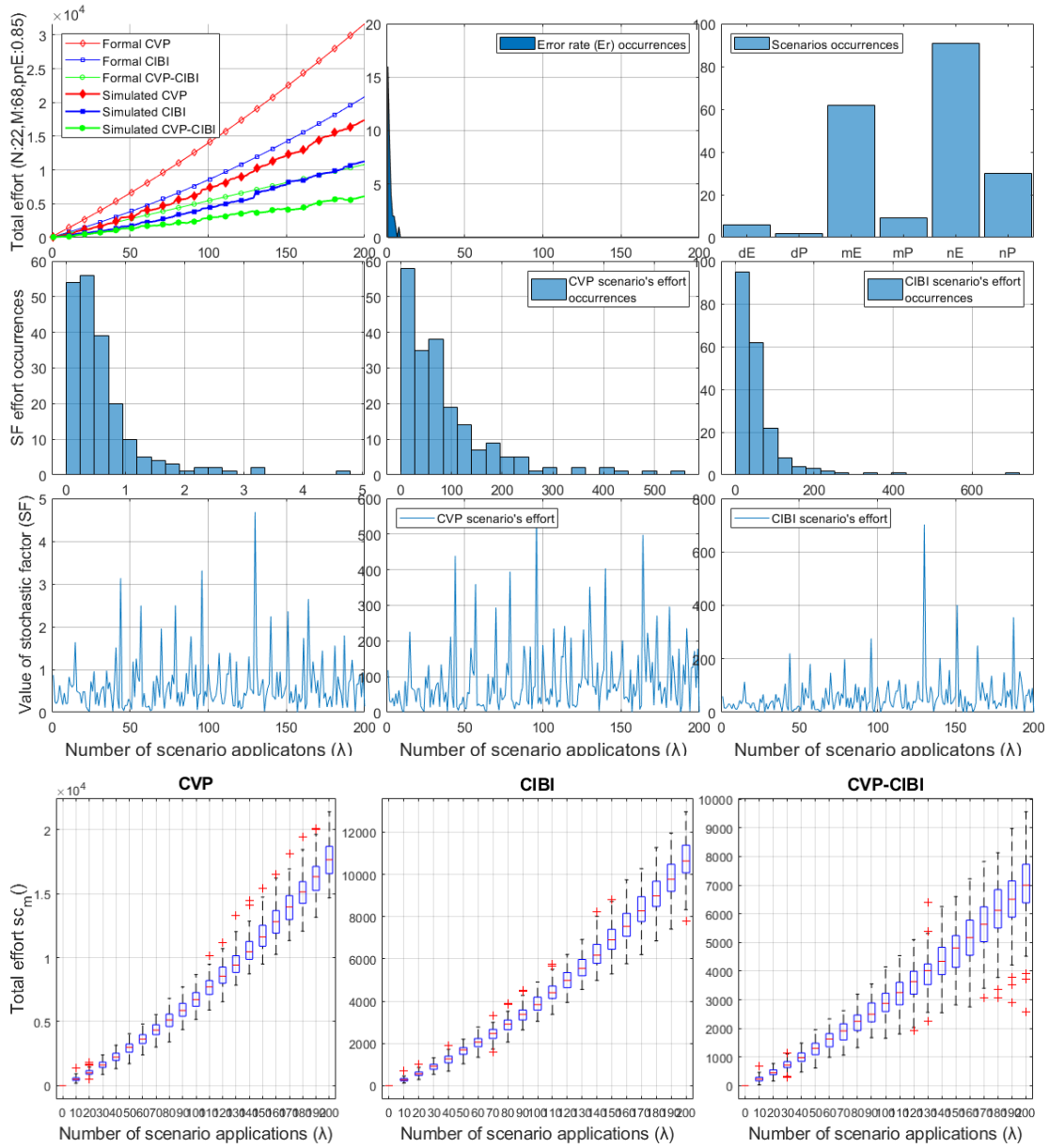


Figure 0-25: Sample instance N. 033 (N=22, M=68,  $p_{nE}=0.85$ ,  $p_{nP}=0.15$ )

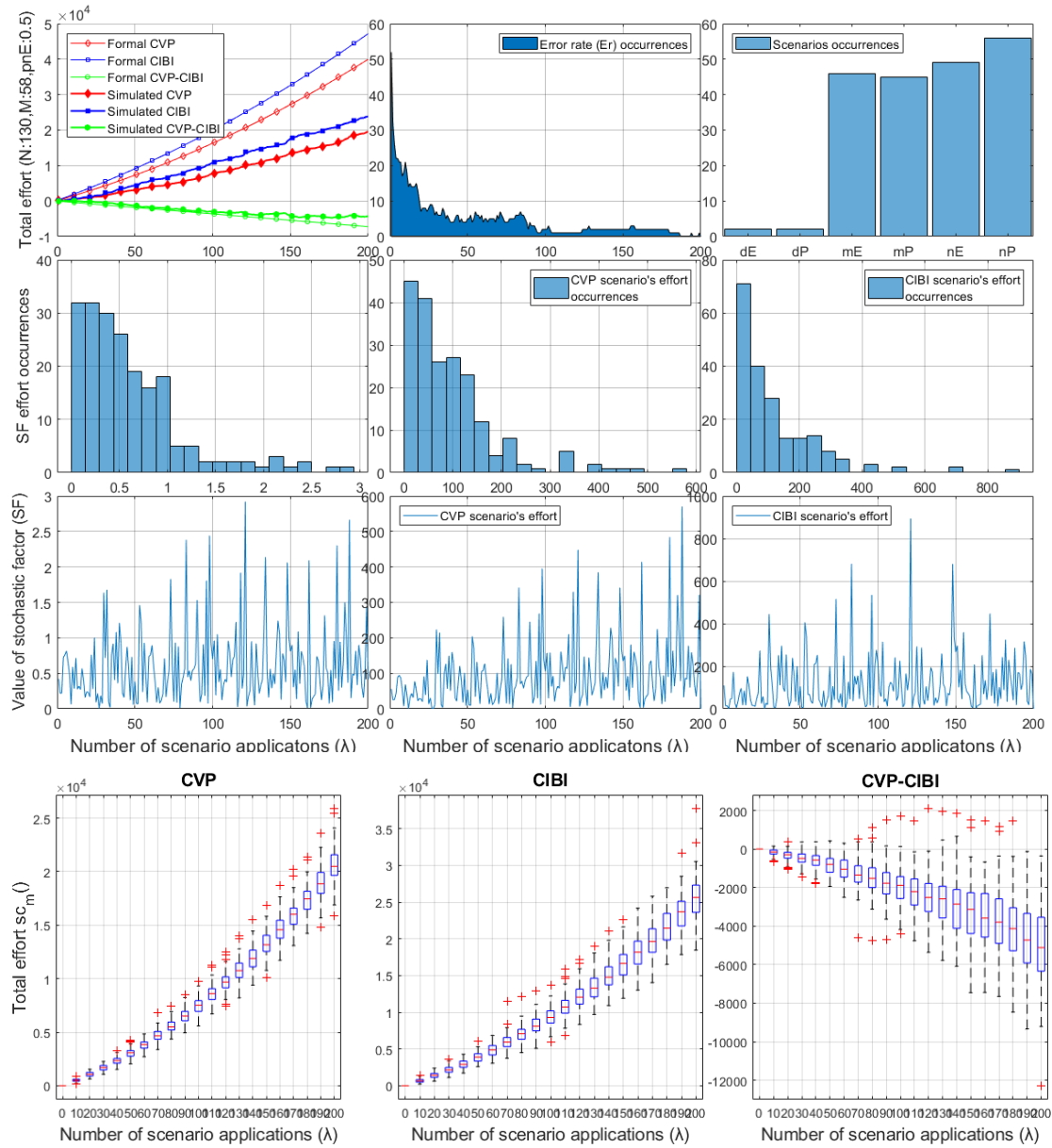


Figure 0-26: Sample instance N. 034 (N=130, M=58,  $p_{nE}=0.50$ ,  $p_{nP}=0.50$ )

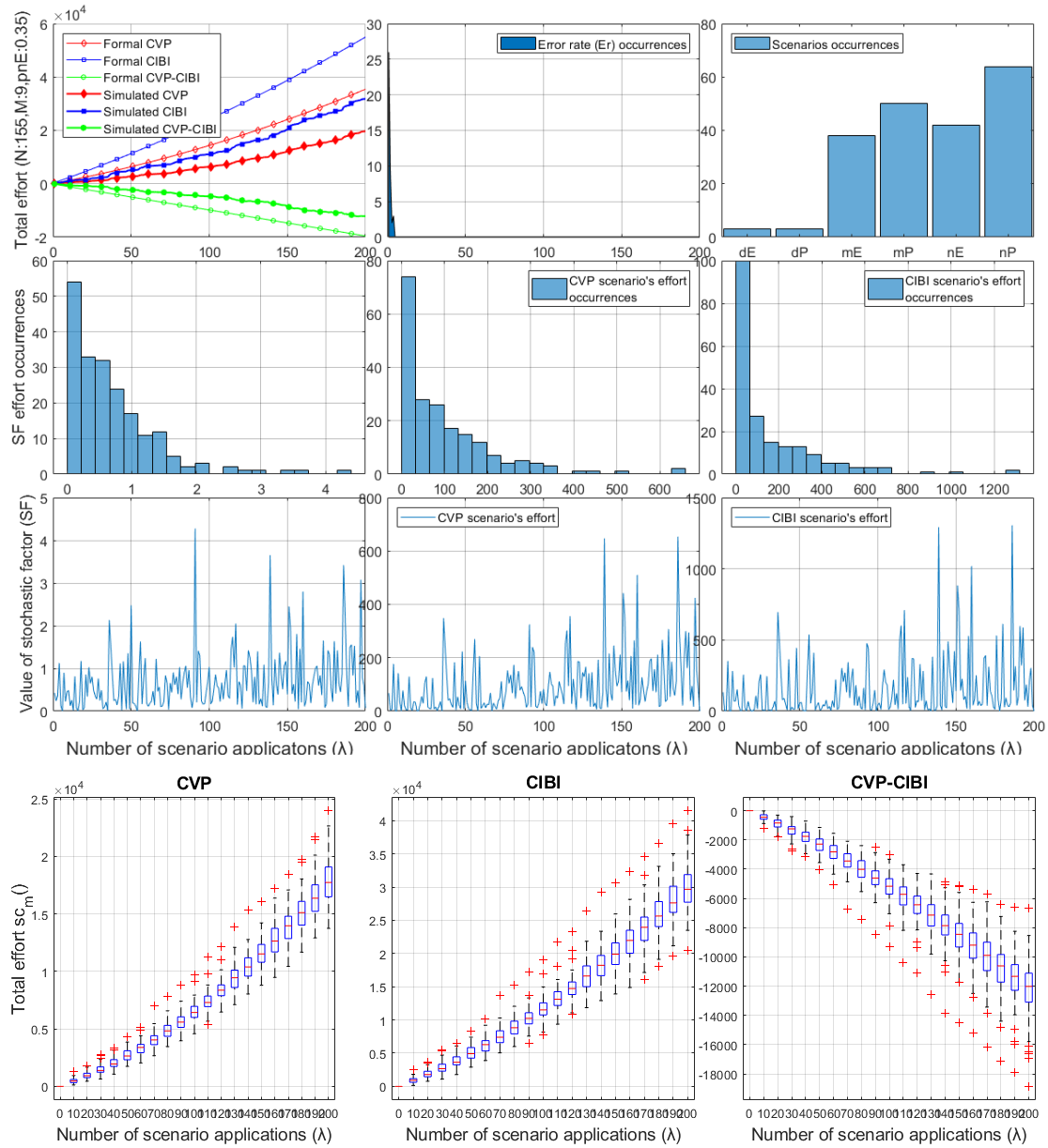


Figure 0-27: Sample instance N. 035 (N=155, M=9,  $p_{nE}=0.35$ ,  $p_{nP}=0.65$ )

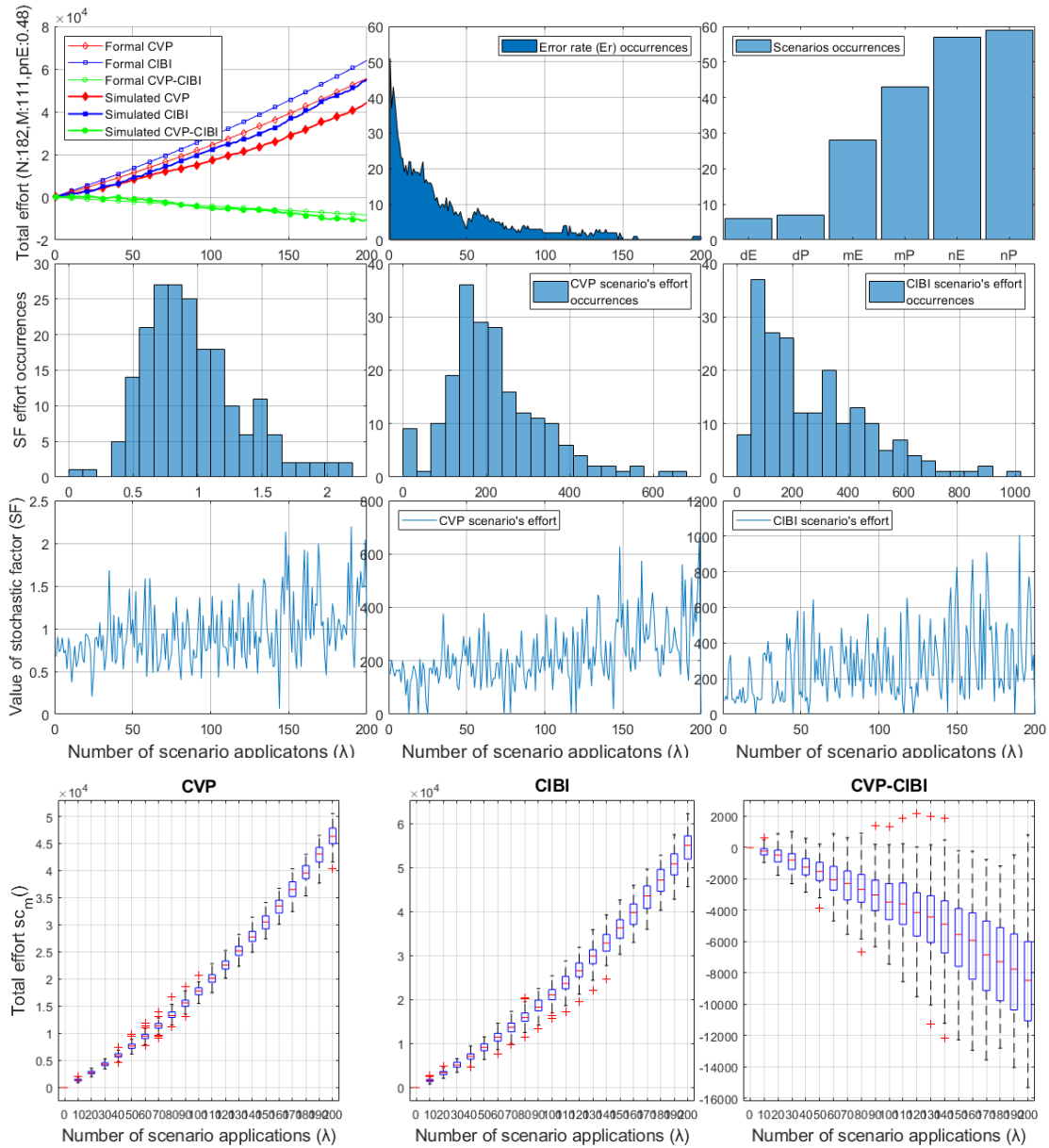


Figure 0-28: Sample instance N. 036 ( $N=182$ ,  $M=111$ ,  $p_{nE}=0.48$ ,  $p_{nP}=0.52$ )

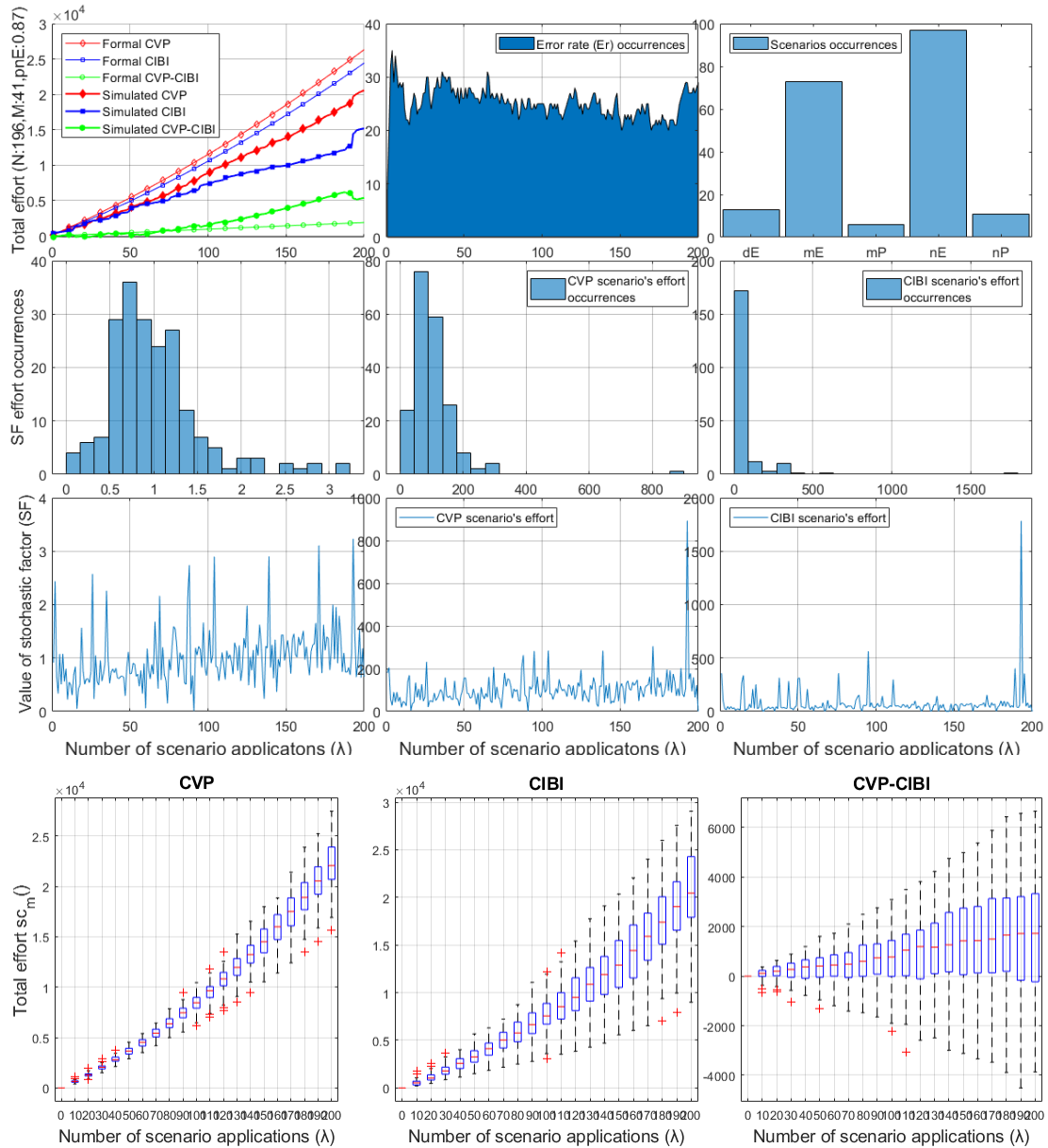


Figure 0-29: Sample instance N. 037 (N=196, M=41,  $p_{nE}=0.87$ ,  $p_{nP}=0.13$ )

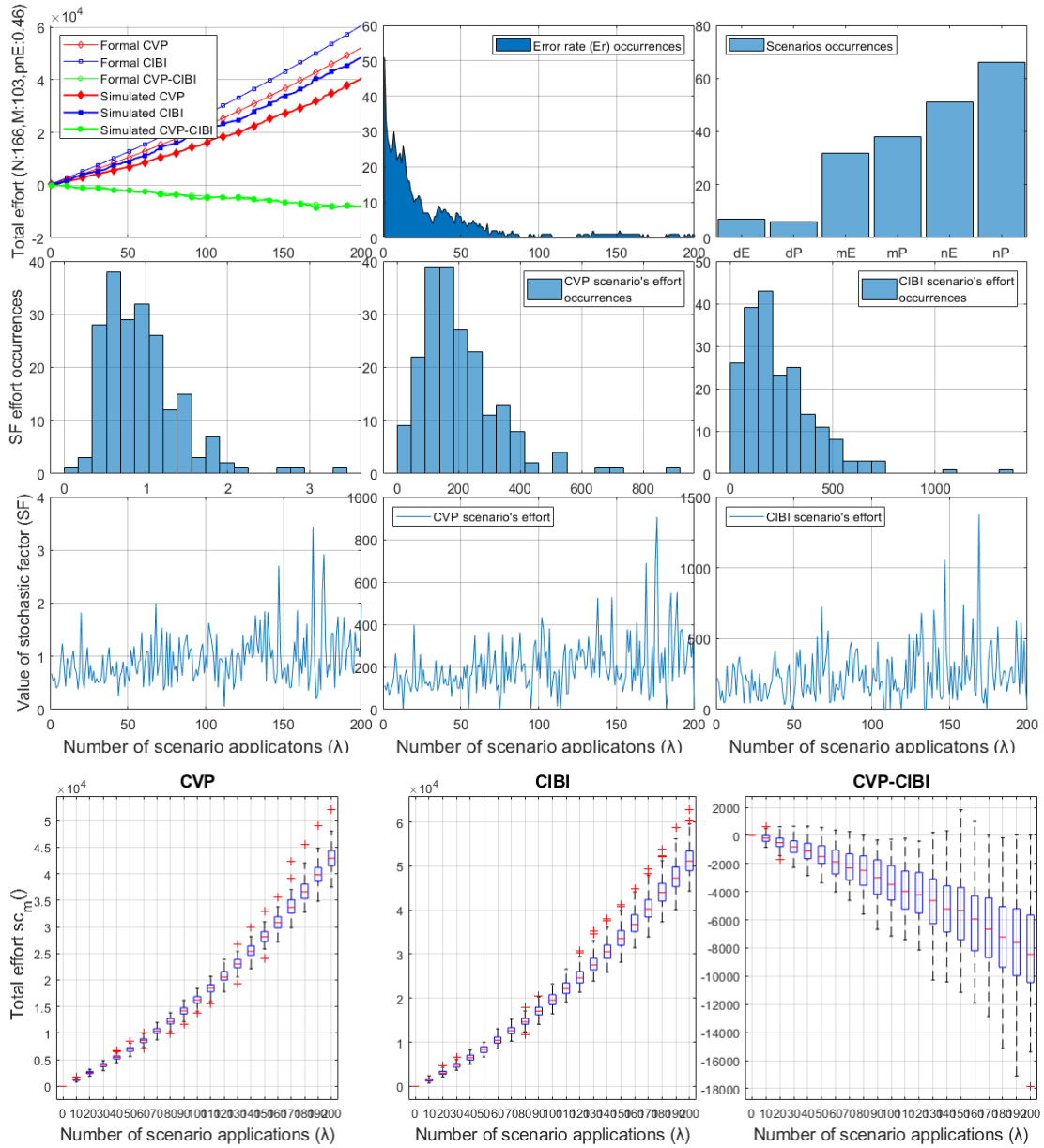


Figure 0-30: Sample instance N. 038 (N=166, M=103,  $p_{nE}=0.46$ ,  $p_{nP}=0.54$ )



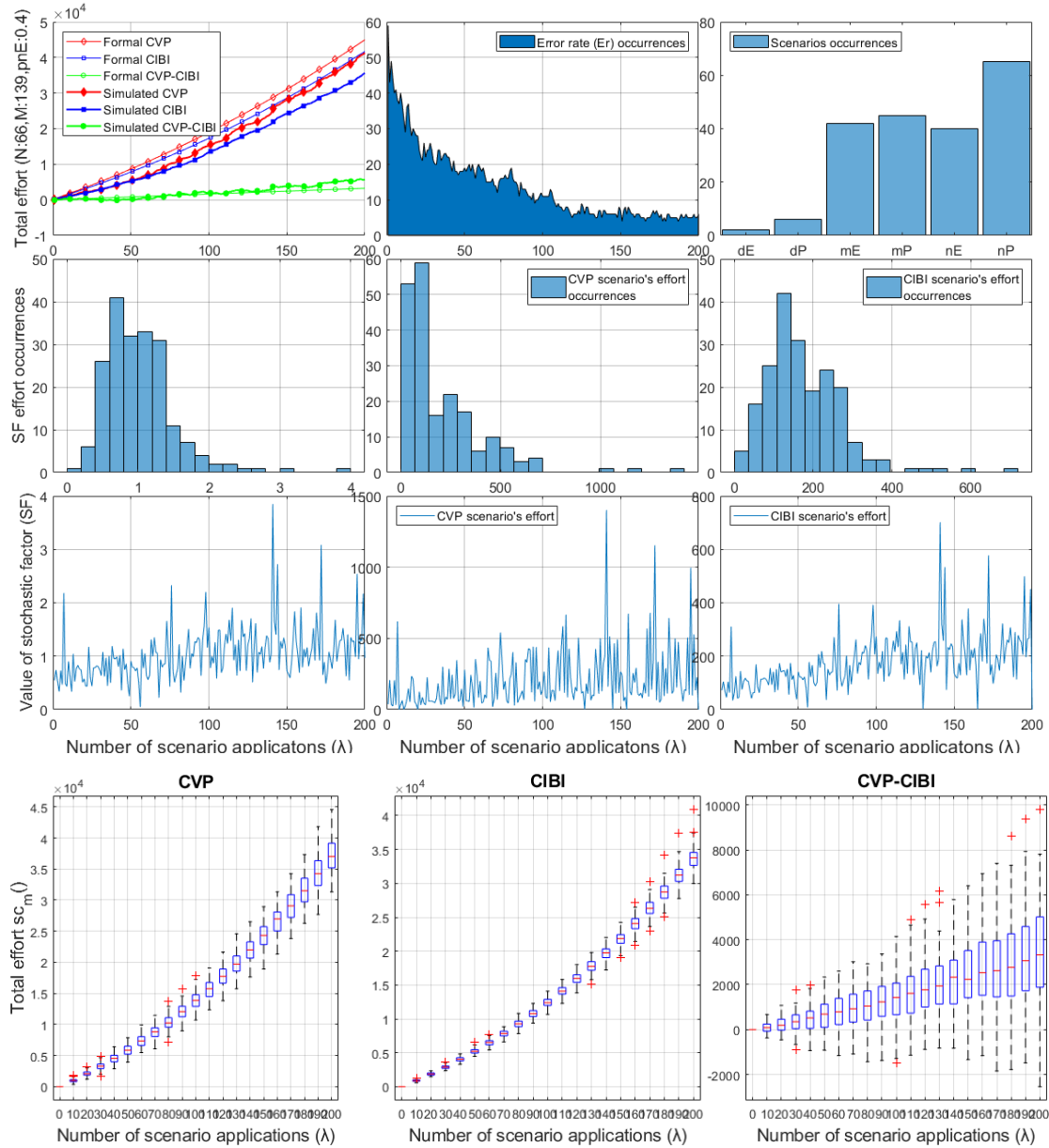


Figure 0-31: Sample instance N. 039 (N=66, M=139,  $p_{nE}=0.40$ ,  $p_{nP}=0.60$ )

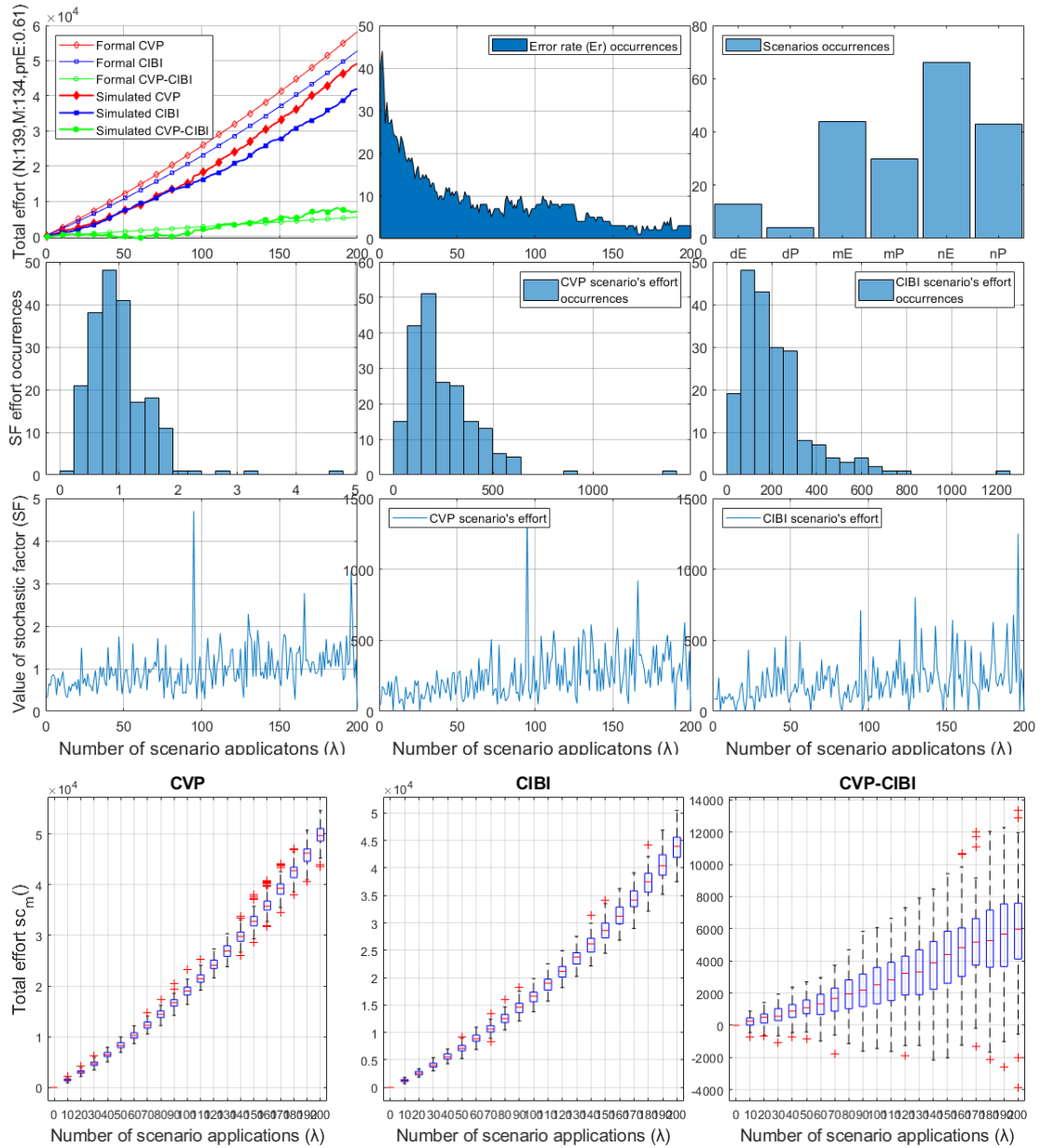


Figure 0-32: Sample instance N. 040 (N=139, M=134,  $p_{nE}=0.61$ ,  $p_{nP}=0.39$ )

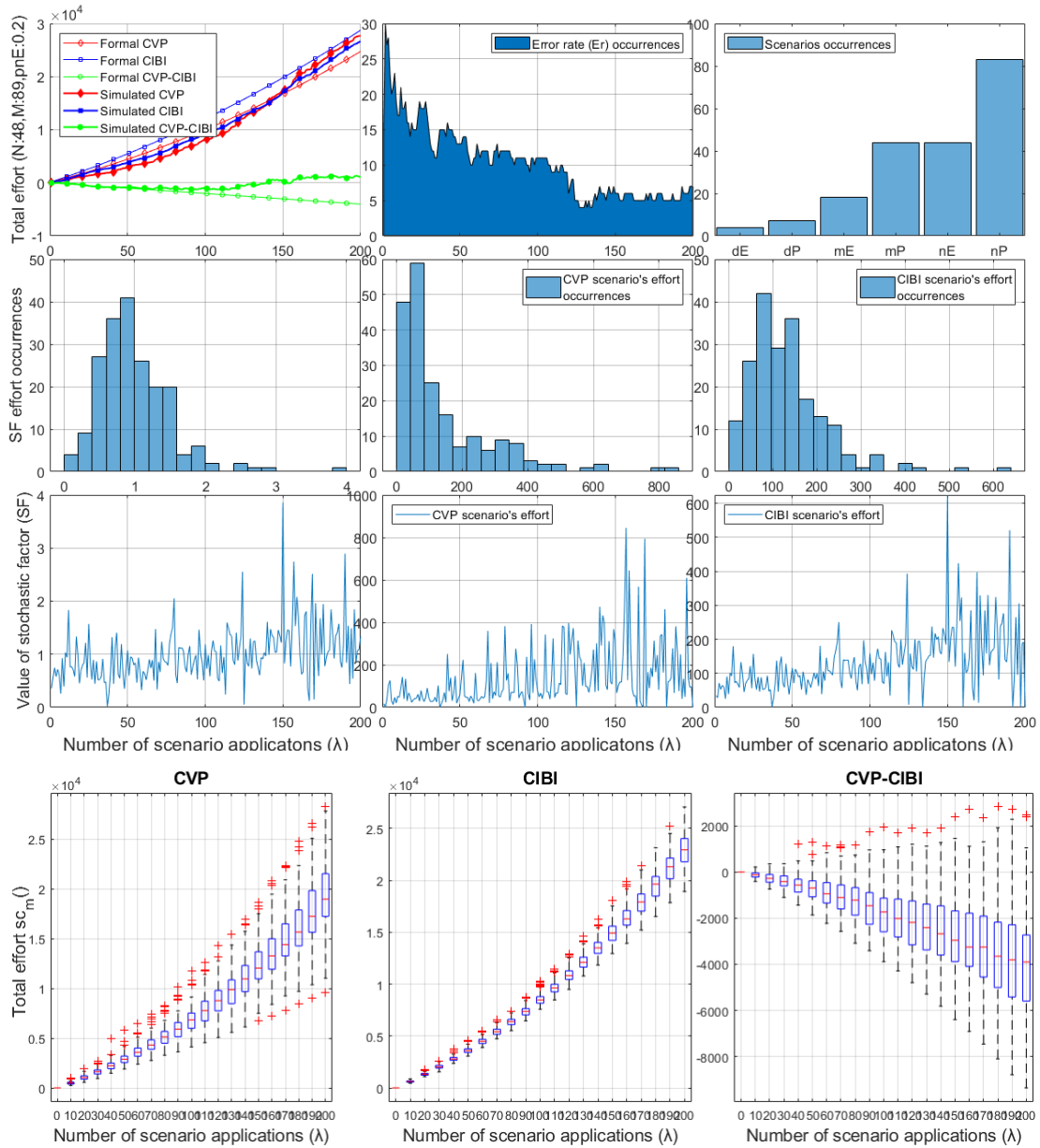


Figure 0-33: Sample instance N. 041 ( $N=48$ ,  $M=89$ ,  $p_{nE}=0.20$ ,  $p_{nP}=0.80$ )

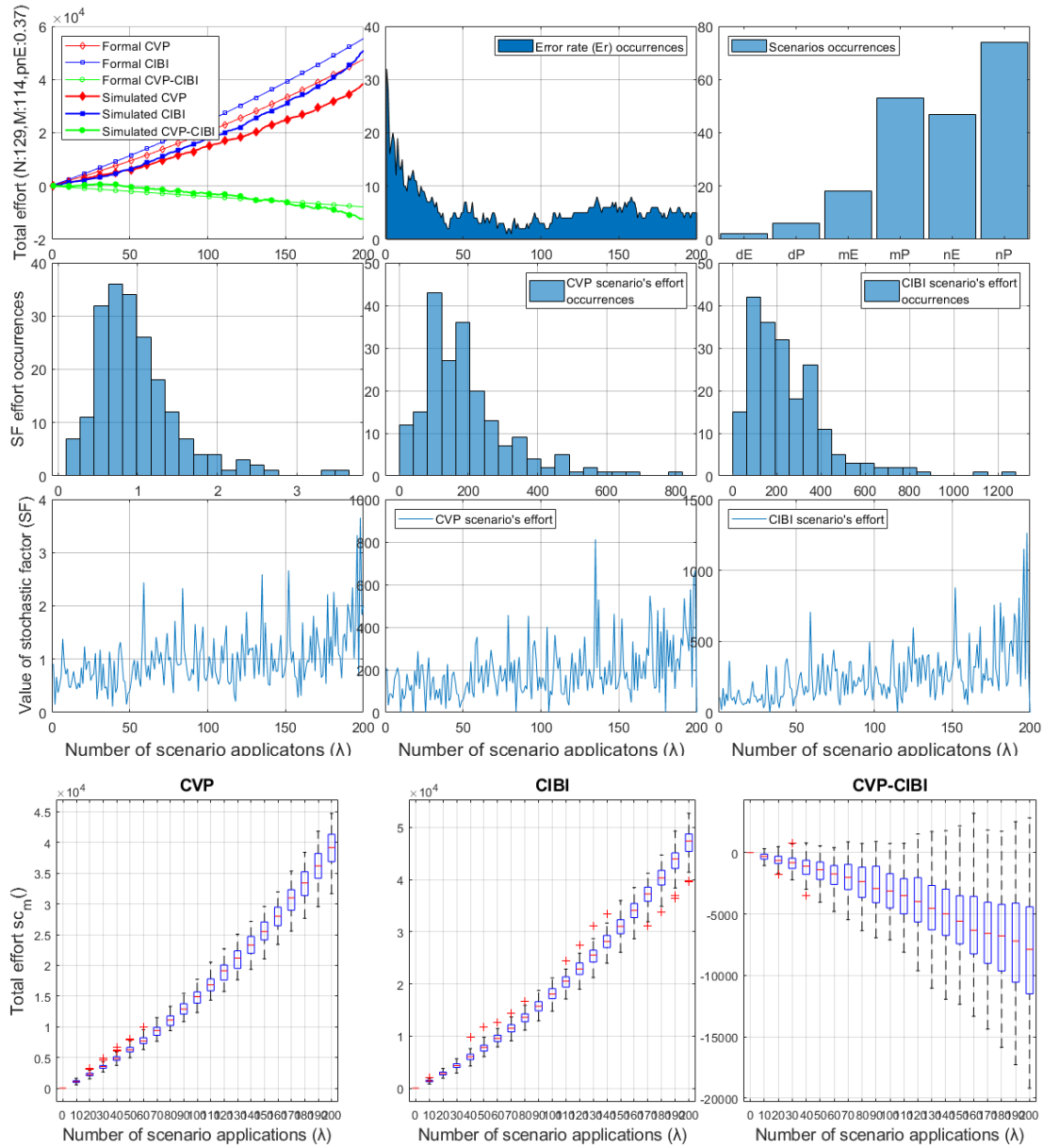


Figure 0-34: Sample instance N. 042 ( $N=129$ ,  $M=114$ ,  $p_{nE}=0.37$ ,  $p_{nP}=0.63$ )

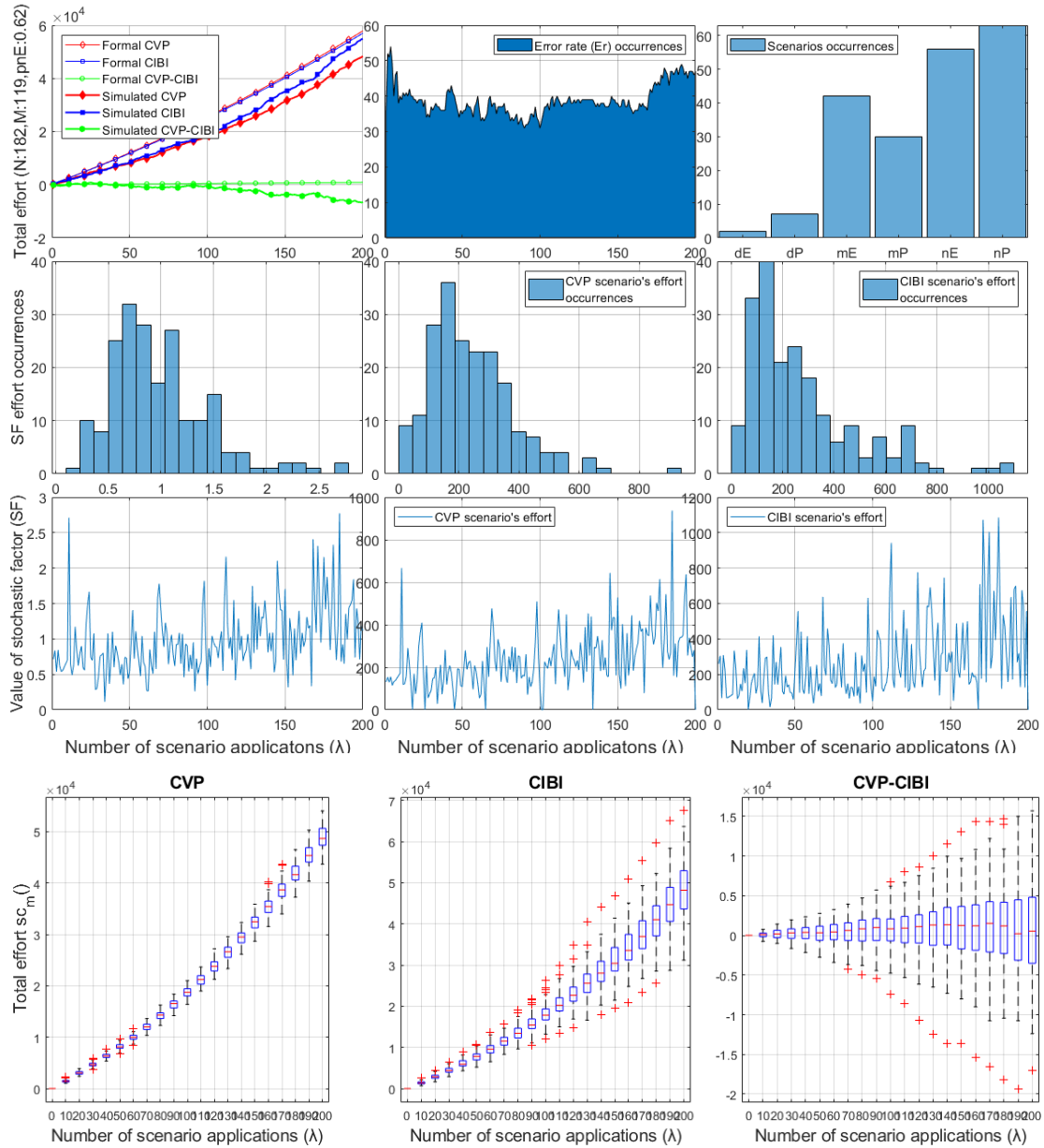


Figure 0-35: Sample instance N. 043 (N=182, M=119,  $p_{nE}=0.62$ ,  $p_{nP}=0.38$ )

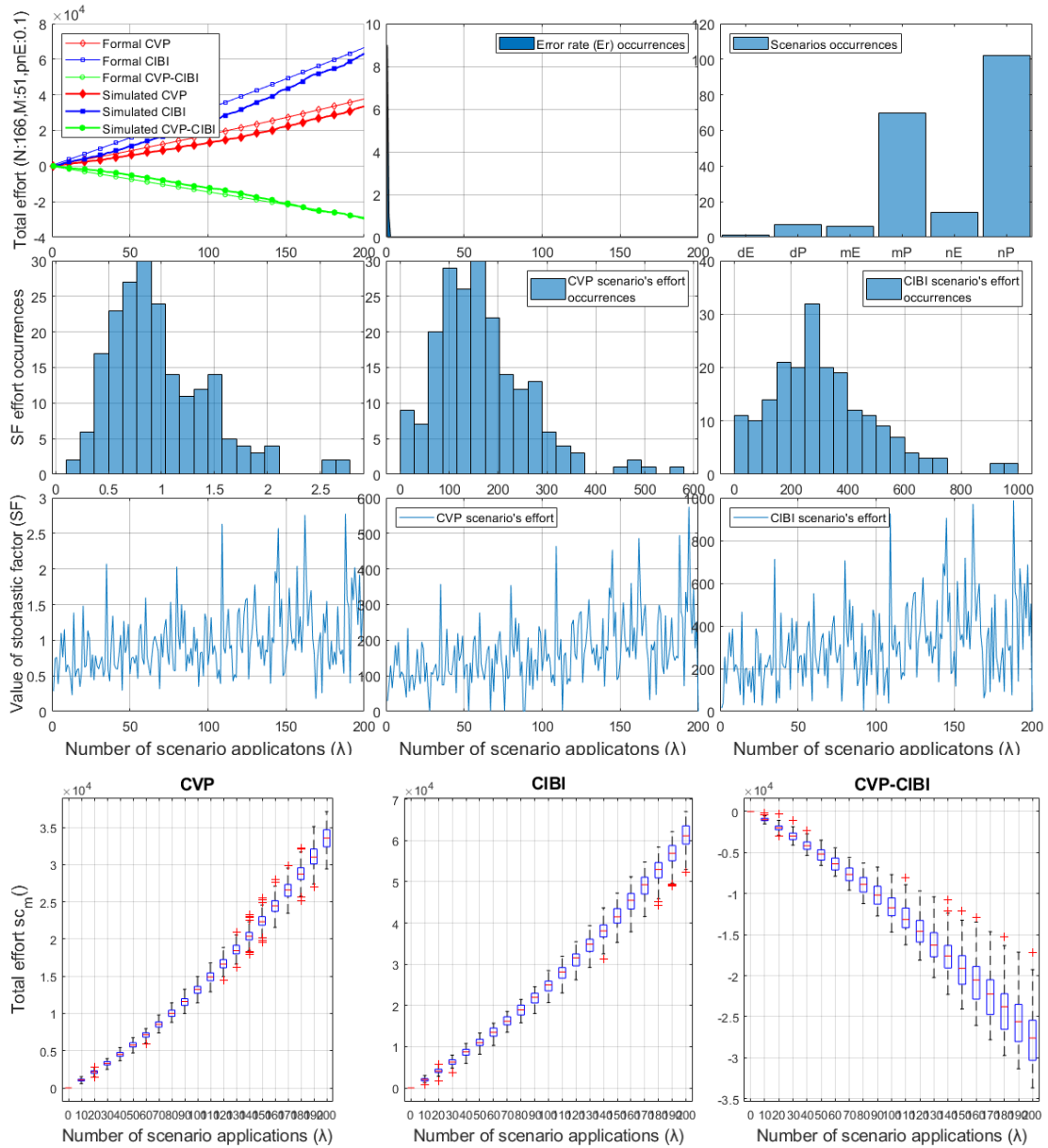


Figure 0-36: Sample instance N. 045 (N=166, M=51,  $p_{nE}=0.10$ ,  $p_{nP}=0.90$ )

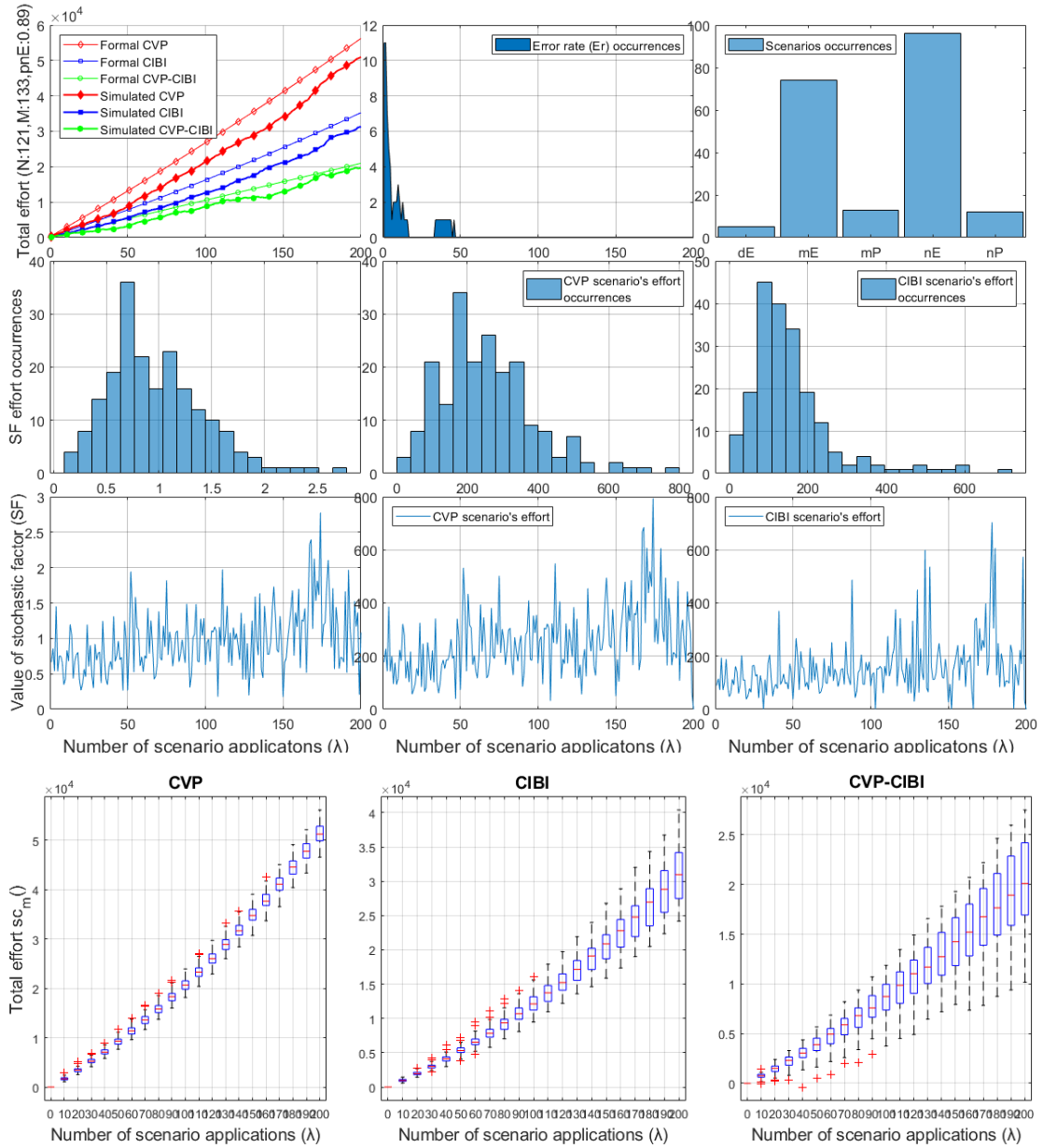


Figure 0-37: Sample instance N. 046 ( $N=121$ ,  $M=133$ ,  $p_{nE}=0.89$ ,  $p_{nP}=0.11$ )

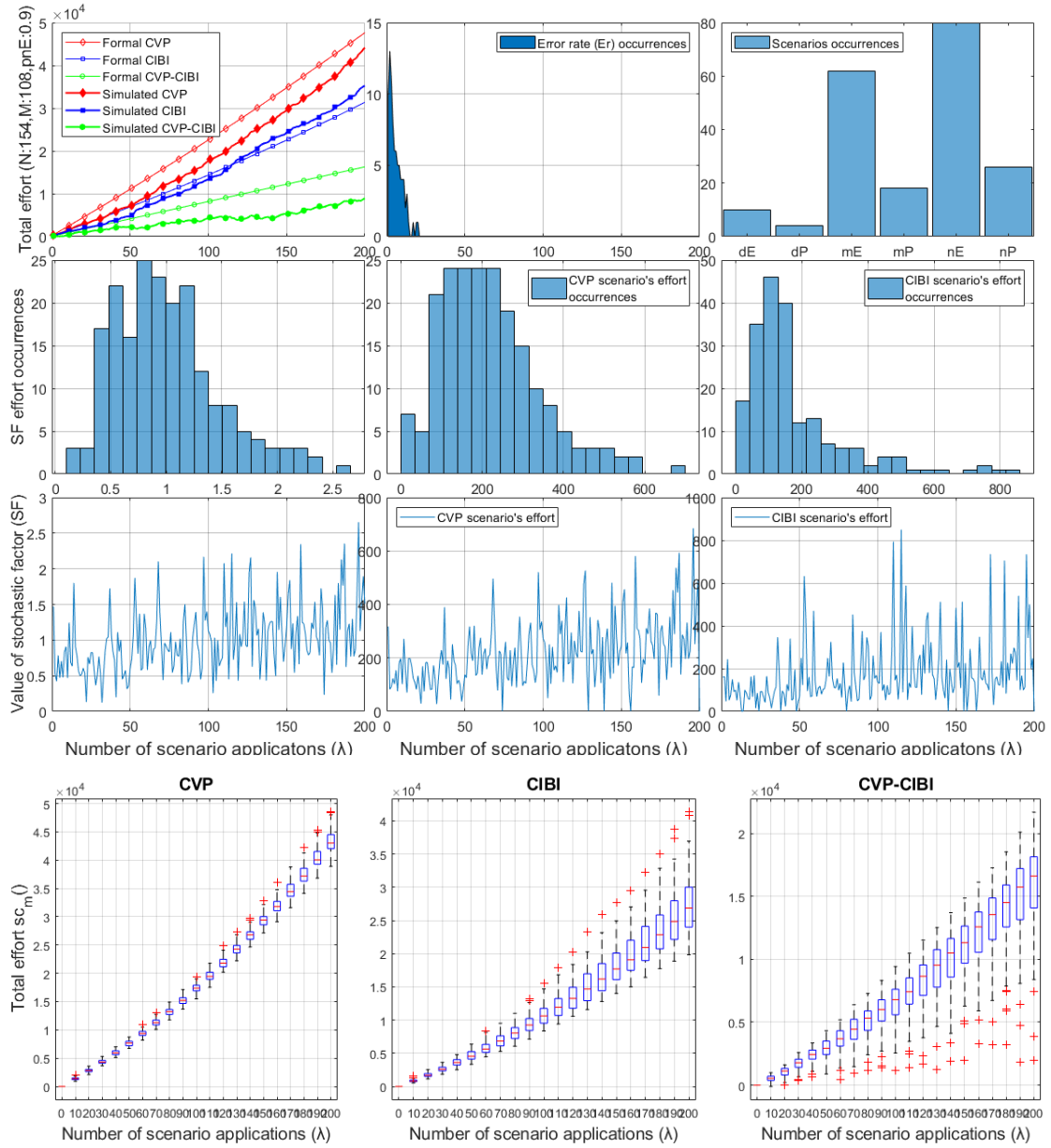


Figure 0-38: Sample instance N. 047 (N=154, M=108,  $p_{nE}=0.90$ ,  $p_{nP}=0.10$ )



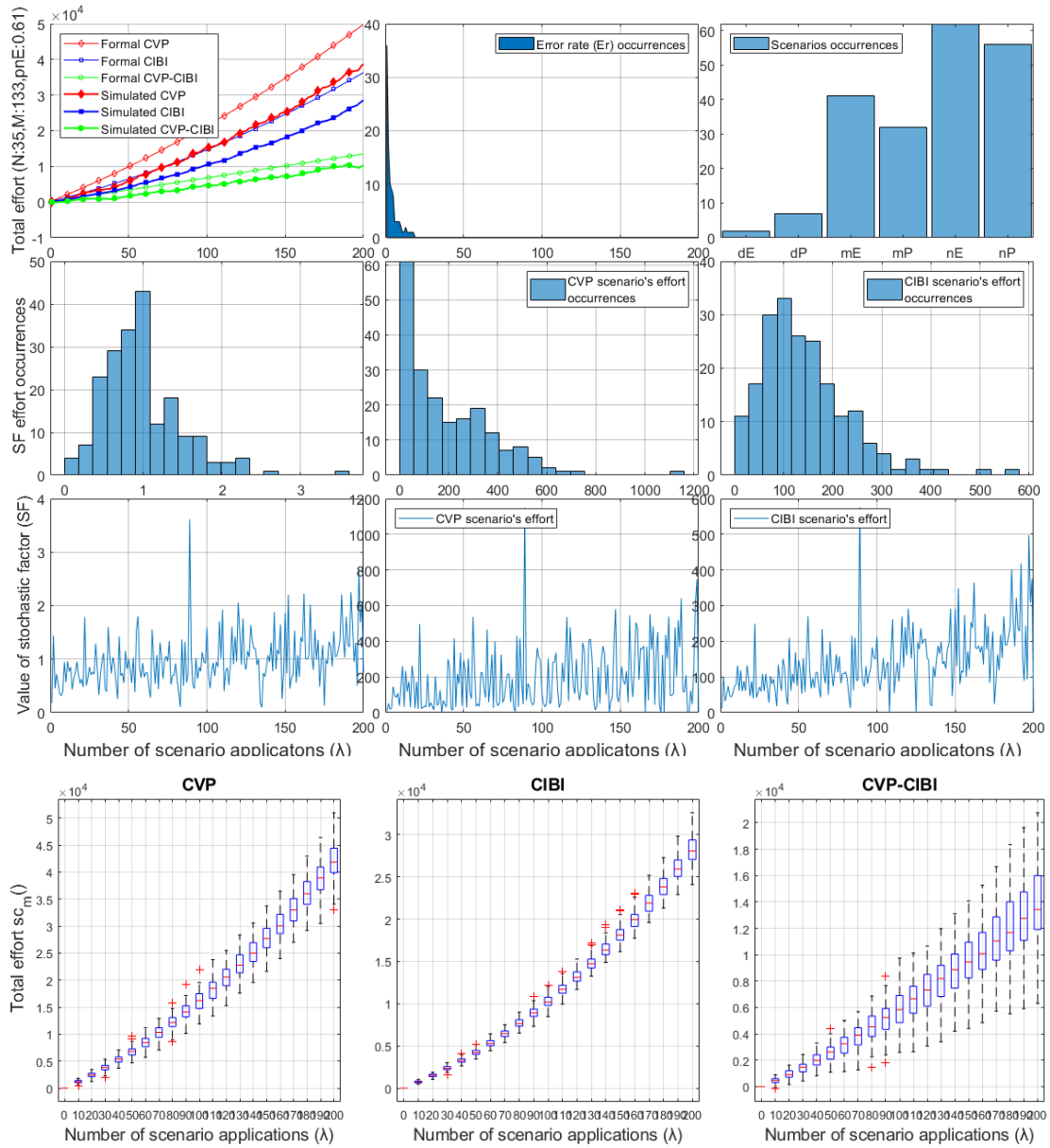


Figure 0-39: Sample instance N. 048 ( $N=35$ ,  $M=133$ ,  $p_{nE}=0.61$ ,  $p_{nP}=0.39$ )

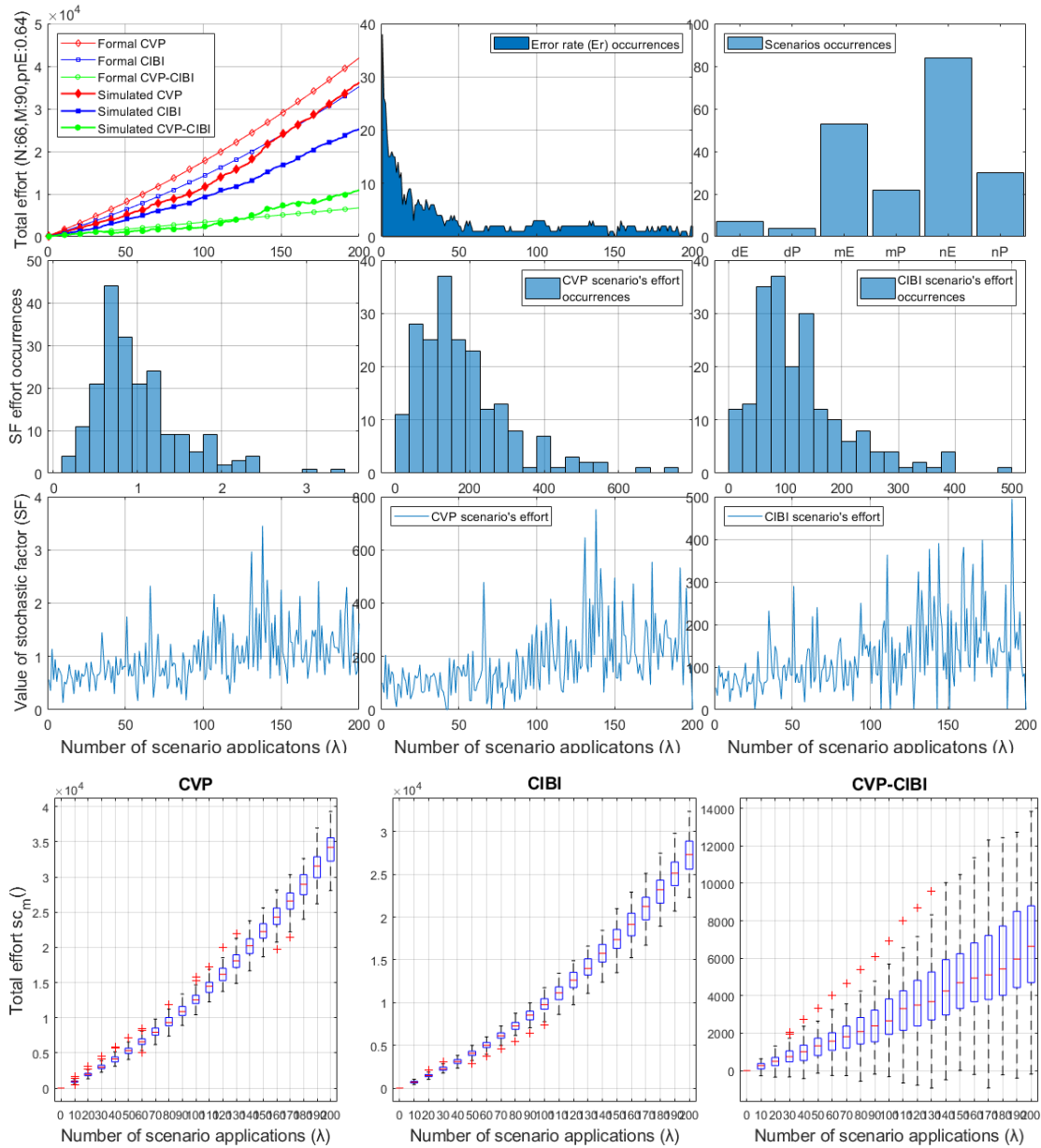


Figure 0-40: Sample instance N. 049 (N=66, M=90,  $p_{nE}=0.64$ ,  $p_{nP}=0.36$ )

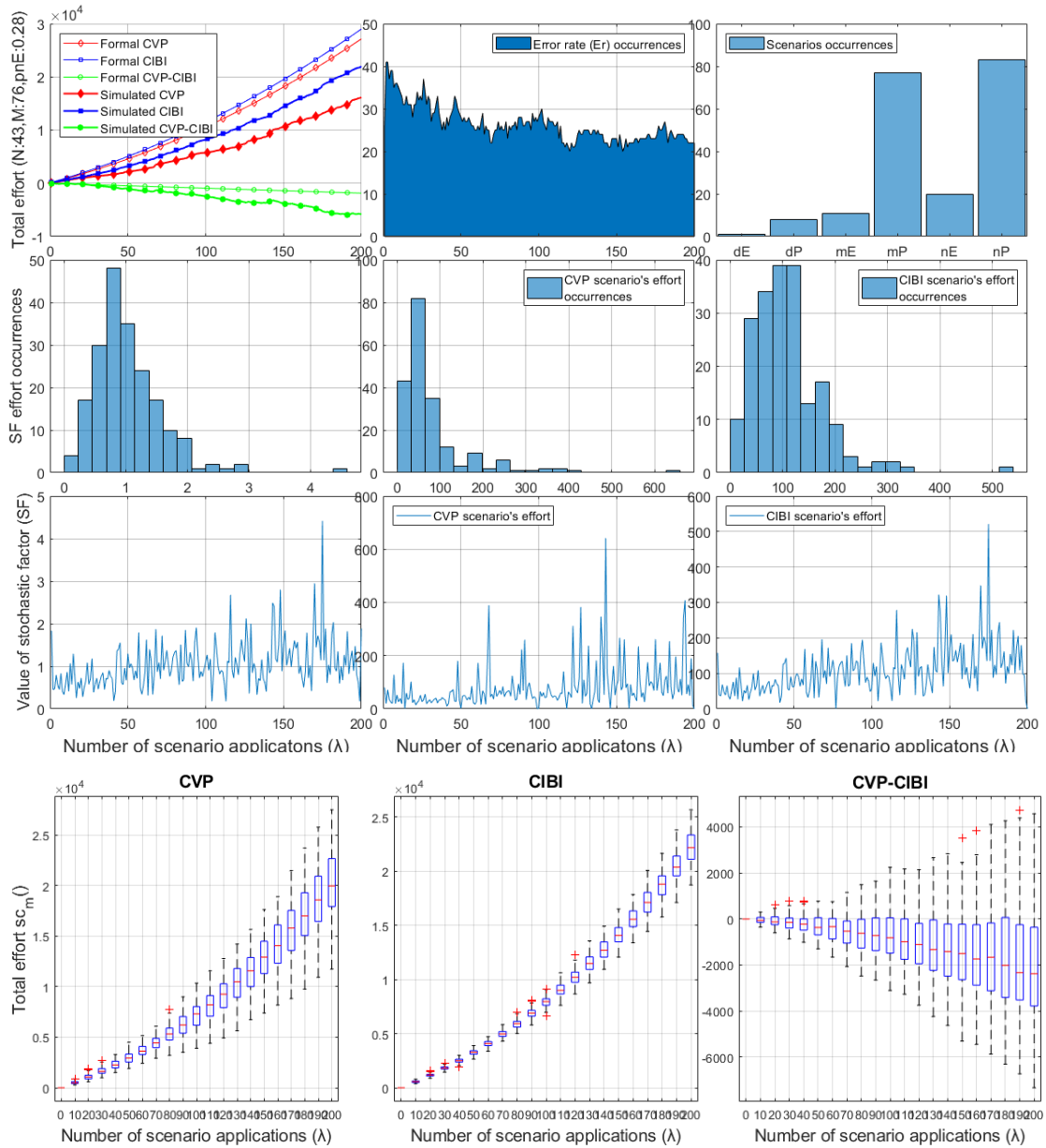


Figure 0-41: Sample instance N. 050 ( $N=43$ ,  $M=76$ ,  $p_{nE}=0.28$ ,  $p_{nP}=0.72$ )

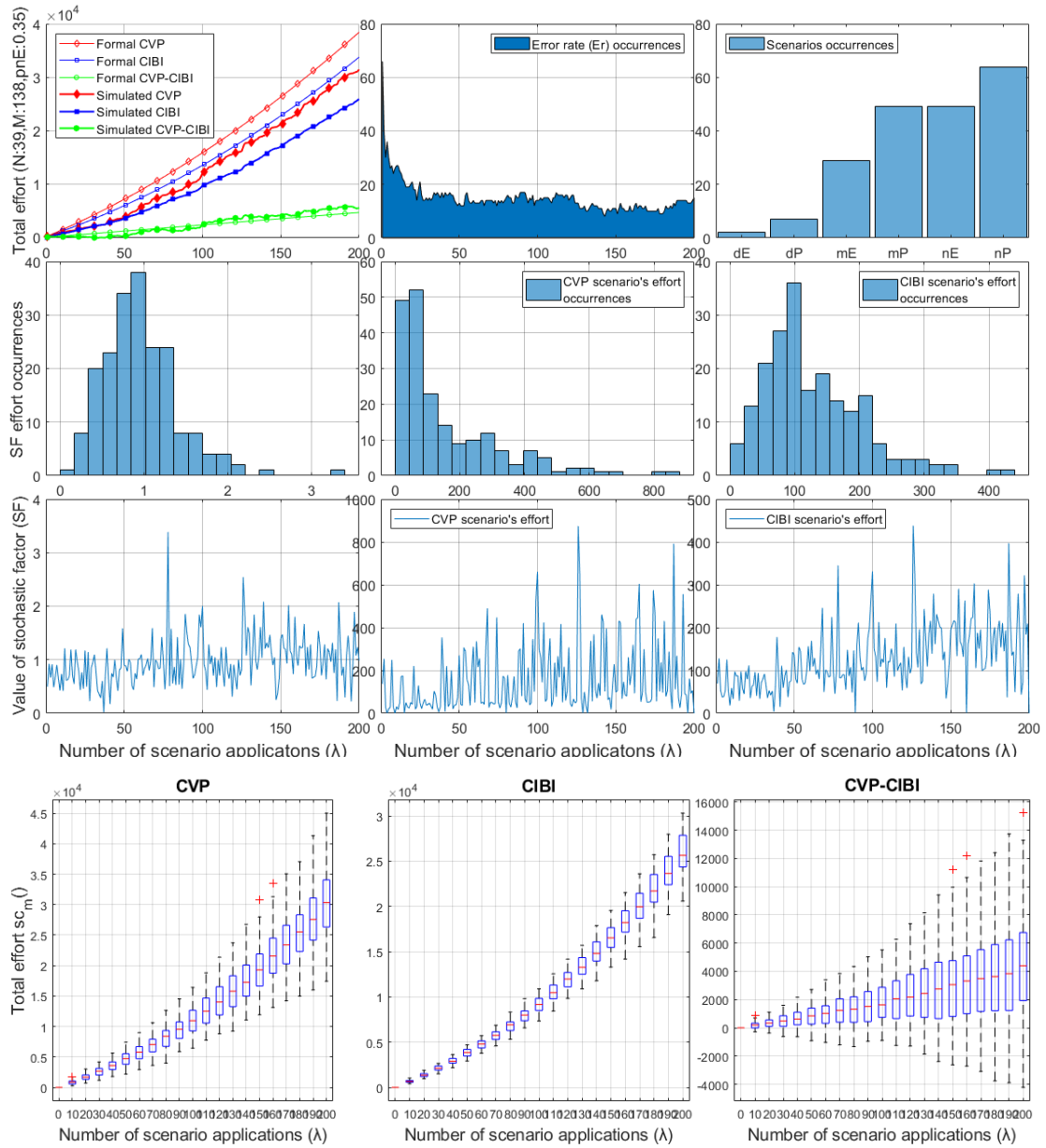


Figure 0-42: Sample instance N. 051 (N=39, M=138,  $p_{nE}=0.35$ ,  $p_{nP}=0.65$ )

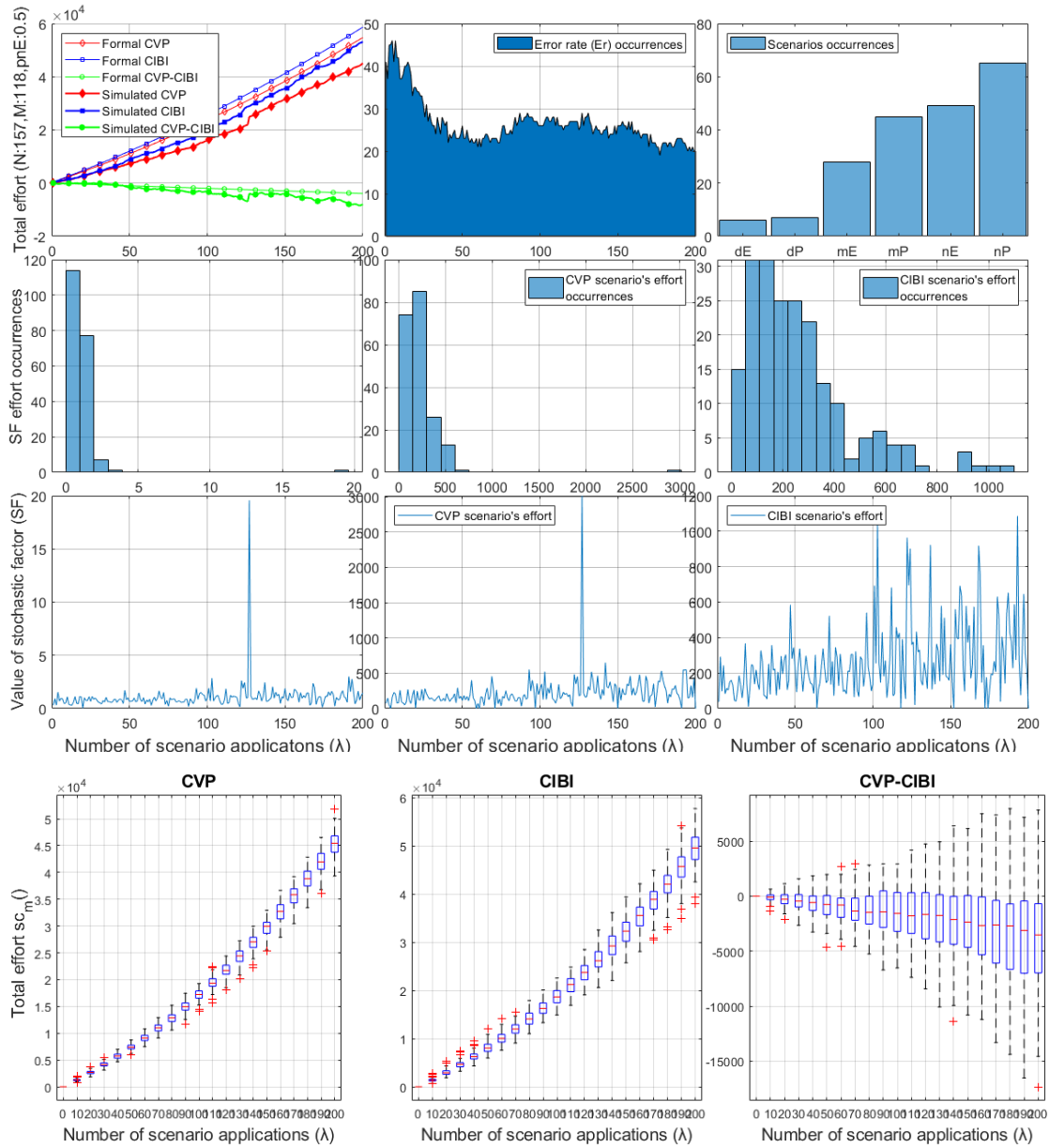


Figure 0-43: Sample instance N. 052 (N=157, M=118,  $p_{nE}=0.50$ ,  $p_{nP}=0.50$ )

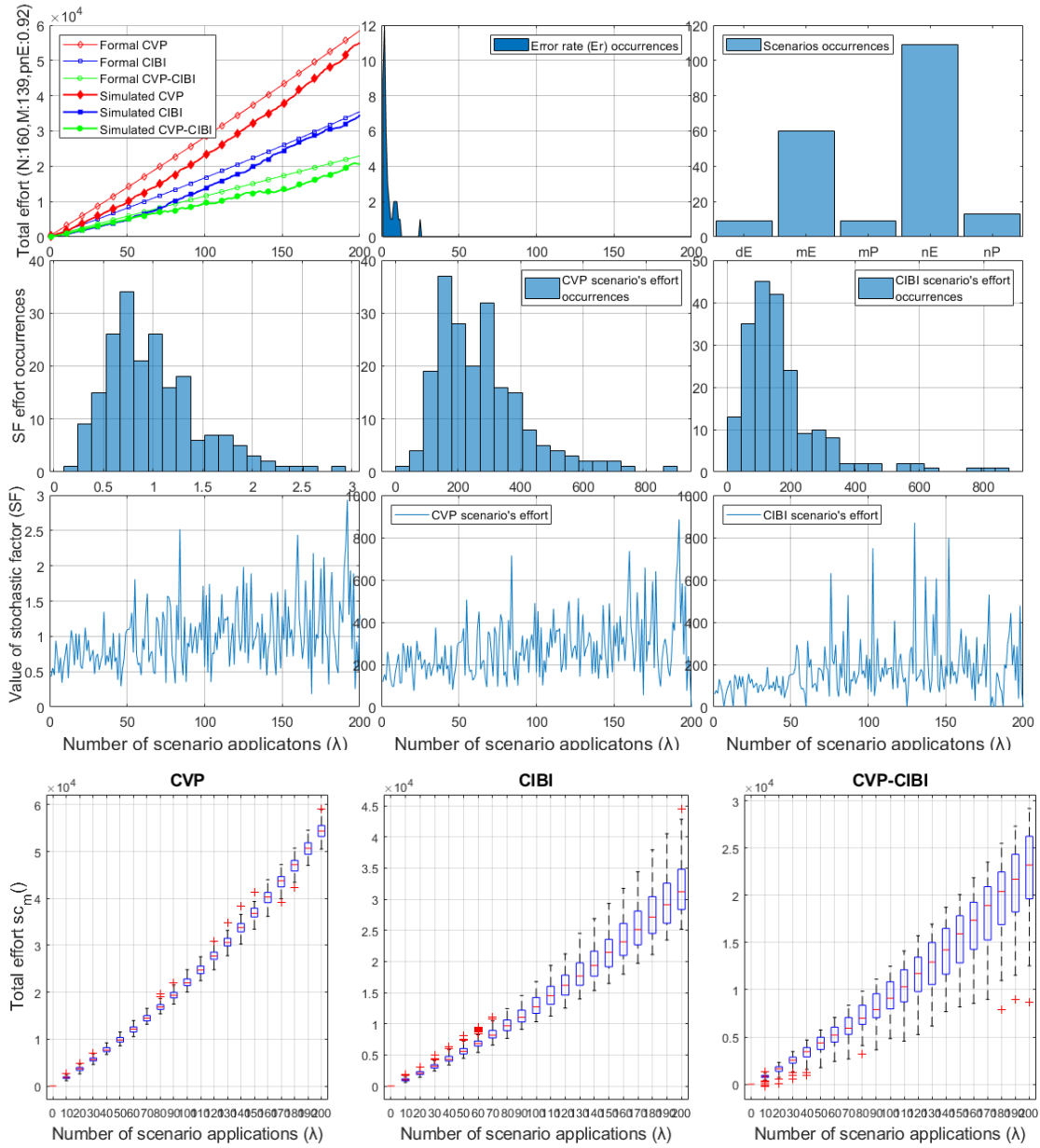


Figure 0-44: Sample instance N. 053 (N=160, M=139,  $p_{nE}=0.92$ ,  $p_{nP}=0.08$ )

## Publications

- C. Karanikolas, G. Dimitroulakos, and K. Masselos, “Early Evaluation of Implementation Alternatives of Composite Data Structures Toward Maintainability”, *ACM Transactions on Software Engineering and Methodology (TOSEM)*, vol. 26, no. 2, p. 8:1-8:44, Oct. 2017, doi: 10.1145/3132731
- C. Karanikolas, G. Dimitroulakos, and K. Masselos, “Simulating Software Evolution to Evaluate the Reliability of Early Decision-Making among Design Alternatives towards Maintainability”, *ACM Transactions on Software Engineering and Methodology (TOSEM)*, p. 1-46, 2022, doi: 10.1145/3569931, *under press (just accepted)*
- C. Karanikolas, G. Dimitroulakos, and K. Masselos, “Object Oriented Software Design Space Exploration for Maintainability based on Evolution Modeling”, *under peer-review in journal*

## Bibliography

- Ahn, Y., Suh, J., Kim, S., & Kim, H. (2003). The Software Maintenance Project Effort Estimation Model Based on Function Points. *Journal of Software Maintenance*, 15(2), 71–85. <http://dx.doi.org/10.1002/smr.269>
- Aho, A. V., Lam, M. S., Sethi, R., & Ullman, J. D. (2006). *Compilers: Principles, Techniques, and Tools* (2nd ed.). Boston: Addison Wesley.
- Alexandrescu, A. (2001). *Modern C++ Design: Generic Programming and Design Patterns Applied*. Boston, MA, USA: Addison-Wesley Longman Publishing Co., Inc.
- Aloysius, A., & Arockiam, L. (2012). Coupling Complexity Metric: A Cognitive Approach. *International Journal of Information Technology and Computer Science*, 4(9), 29–35. <http://dx.doi.org/10.5815/ijitcs.2012.09.04>
- Aloysius, A., & Arockiam, L. (2013). Maintenance effort prediction model using cognitive complexity metrics. *International Journal of Advanced Research in Computer Science and Software Engineering*, 3(11), 1599–1608.
- Antoniol, G., Casazza, G., Di Penta, M., & Merlo, E. (2001). Modeling clones evolution through time series. *Proceedings IEEE International Conference on Software Maintenance. ICSM 2001*, 273–280. <https://doi.org/10.1109/ICSM.2001.972740>
- Antoniol, Giuliano, Fiutem, R., & Cristoforetti, L. (1998). Using metrics to identify design patterns in Object-Oriented software. *International Software Metrics Symposium, Proceedings*, 23–33. Scopus. <http://dx.doi.org/10.1109/METRIC.1998.731224>
- Antoniol, Giuliano, Lokan, C., Caldiera, G., & Fiutem, R. (1999). A Function Point-Like Measure for Object-Oriented Software. *Empirical Software Engineering*, 4(3), 263–287. <https://doi.org/10.1023/A:1009834811663>
- Araújo, M. A. P., Monteiro, V. F., & Travassos, G. H. (2012). Towards a model to support in silico studies of software evolution. *Proceedings of the 2012 ACM-IEEE International Symposium on Empirical Software Engineering and Measurement*, 281–289. <https://doi.org/10.1145/2372251.2372303>
- Arbuckle, T. (2011). Studying Software Evolution Using Artefacts' Shared Information Content. *Sci. Comput. Program.*, 76(12), 1078–1097. <http://dx.doi.org/10.1016/j.scico.2010.11.005>
- Aroonvatanaporn, P., Sinthop, C., & Boehm, B. (2010). *Reducing estimation uncertainty with continuous assessment: Tracking the “cone of uncertainty.”* 337–340. Scopus. <https://doi.org/10.1145/1858996.1859065>
- Aversano, L., Canfora, G., Cerulo, L., Del Grosso, C., & Di Penta, M. (2007). An Empirical Study on the Evolution of Design Patterns. *Proceedings of the the 6th Joint Meeting of the European Software Engineering Conference and the ACM SIGSOFT Symposium on The Foundations of Software Engineering*, 385–394. New York, NY, USA: ACM. <http://dx.doi.org/10.1145/1287624.1287680>
- Aversano, L., Cerulo, L., & Di Penta, M. (2009). Relationship between design patterns defects and crosscutting concern scattering degree: An empirical study. *IET Software*, 3(5), 395–409. <http://dx.doi.org/10.1049/iet-sen.2008.0105>
- Baker, A. L., Bieman, J. M., Fenton, N., Gustafson, D. A., Melton, A., & Whitty, R. (1990). A Philosophy for Software Measurement. *Journal of System and Software*, 12(3), 277–281. [http://dx.doi.org/10.1016/0164-1212\(90\)90050-V](http://dx.doi.org/10.1016/0164-1212(90)90050-V)
- Bakota, T., Hegedűs, P., Körtvélyesi, P., Ferenc, R., & Gyimóthy, T. (2011). A probabilistic software quality model. *2011 27th IEEE International Conference on Software Maintenance (ICSM)*, 243–252. <http://dx.doi.org/10.1109/ICSM.2011.6080791>
- Bakota, T., Hegedűs, Peter., Ladányi, G., Körtvélyesi, P., Ferenc, R., & Gyimóthy, T. (2012). A cost model based on software maintainability. *2012 28th IEEE International Conference on Software Maintenance (ICSM)*, 316–325. <http://dx.doi.org/10.1109/ICSM.2012.6405288>
- Baldwin, D. (2003). A compiler for teaching about compilers. *SIGCSE '03 Proceedings of the 34th SIGCSE Technical Symposium on Computer Science Education*, 220–223. Reno, Nevada, USA: ACM. Scopus. <http://dx.doi.org/10.1145/611892.611974>
- Bandi, R. K., Vaishnavi, V. K., & Turk, D. E. (2003). Predicting maintenance performance using object-oriented design complexity metrics. *IEEE Transactions on Software Engineering*, 29(1), 77–87. <http://dx.doi.org/10.1109/TSE.2003.1166590>
- Bansiya, J., & Davis, C. G. (2002). A Hierarchical Model for Object-Oriented Design Quality Assessment. *IEEE Trans. Softw. Eng.*, 28(1), 4–17. <http://dx.doi.org/10.1109/32.979986>



- Barros, M. D. O., Werner, C. M. L., & Travassos, G. H. (2004). Supporting Risks in Software Project Management. *J. Syst. Softw.*, 70(1–2), 21–35. [https://doi.org/10.1016/S0164-1212\(02\)00155-3](https://doi.org/10.1016/S0164-1212(02)00155-3)
- Barry, E. J., Kemerer, C. F., & Slaughter, S. A. (2007). How software process automation affects software evolution: A longitudinal empirical analysis. *Journal of Software Maintenance and Evolution: Research and Practice*, 19(1), 1–31. <https://doi.org/10.1002/smr.342>
- Bartosz, W., & Pawel, M. (2010). Hierarchical Model for Evaluating Software Design Quality. *E-Informatica Software Engineering Journal*, 4(1), 21–30.
- Basili, V. R., & Rombach, H. D. (1988). The TAME project: Towards improvement-oriented software environments. *IEEE Transactions on Software Engineering*, 14(6), 758–773. <https://doi.org/10.1109/32.6156>
- Basili, Victor R., Briand, L. C., & Melo, W. L. (1996). A Validation of Object-Oriented Design Metrics As Quality Indicators. *IEEE Transactions on Software Engineering*, 22(10), 751–761. Scopus. <http://dx.doi.org/10.1109/32.544352>
- Bass, L., Clements, P., & Kazman, R. (2012). *Software Architecture in Practice* (3rd Edition). Upper Saddle River, NJ: Addison-Wesley Professional.
- Bengtsson, P., & Bosch, J. (1999). Architecture level prediction of software maintenance. *Proceedings of the Third European Conference on Software Maintenance and Reengineering (Cat. No. PR00090)*, 139–147. <https://doi.org/10.1109/CSMR.1999.756691>
- Berenson, M. L., Levine, D. M., & Timothy, K. C. (2012). *Basic Business Statistics, Concepts and Applications* (12 edition). Pearson.
- Bernardi, M. L., & Di Lucca, G. A. (2010). Model-driven detection of Design Patterns. *2010 IEEE International Conference on Software Maintenance (ICSM)*, 1–5. <http://dx.doi.org/10.1109/ICSM.2010.5609740>
- Bhattacharya, R. N., & Waymire, E. C. (2009). *Stochastic Processes with Applications* (Siam Classics ed. edition). Philadelphia: Society for Industrial and Applied Mathematics.
- Bidve, V. S., & Sarasu, P. (2016). Tool for Measuring Coupling in Object-Oriented Java Software. *International Journal of Engineering and Technology*, 8(2), 812–820.
- Bieman, J. M., Jain, D., & Yang, H. J. (2001). OO design patterns, design structure, and program changes: An industrial case study. *IEEE International Conference on Software Maintenance, 2001. Proceedings*, 580–589. <http://dx.doi.org/10.1109/ICSM.2001.972775>
- Boehm, B. (2008). Making a Difference in the Software Century. *Computer*, 41(3), 32–38. <https://doi.org/10.1109/MC.2008.91>
- Boehm, B., Clark, B., Horowitz, E., Westland, C., Madachy, R., & Selby, R. (1995). Cost models for future software life cycle processes: COCOMO 2.0. *Annals of Software Engineering*, 1(1), 57–94. <https://doi.org/10.1007/BF02249046>
- Boehm, B. W. (1984). Software Engineering Economics. *IEEE Transactions on Software Engineering*, SE-10(1), 4–21. <https://doi.org/10.1109/TSE.1984.5010193>
- Boehm, Barry W., Clark, Horowitz, Brown, Reifer, Chulani, ... Steece, B. (2000). *Software Cost Estimation with Cocomo II* (1st ed.). Upper Saddle River, NJ, USA: Prentice Hall PTR.
- Bosch, J., & Bengtsson, P. (2001). Assessing optimal software architecture maintainability. *Proceedings Fifth European Conference on Software Maintenance and Reengineering*, 168–175. <https://doi.org/10.1109/.2001.914981>
- Briand, L. C., Daly, J., Porter, V., & Wüst, J. (1998). Comprehensive empirical validation of design measures for object-oriented systems. *International Software Metrics Symposium, Proceedings*, 246–257. Scopus. <http://dx.doi.org/10.1109/METRIC.1998.731251>
- Briand, L. C., Daly, J. W., & Wüst, J. K. (1998). A Unified Framework for Cohesion Measurement in Object-Oriented Systems. *Empirical Software Engineering*, 3(1), 65–117. Scopus. <http://dx.doi.org/10.1023/A:1009783721306>
- Briand, L. C., Daly, J. W., & Wüst, J. K. (1999). A unified framework for coupling measurement in object-oriented systems. *IEEE Transactions on Software Engineering*, 25(1), 91–121. <http://dx.doi.org/10.1109/32.748920>
- Briand, L. C., Melo, W. L., & Wust, J. (2002). Assessing the applicability of fault-proneness models across object-oriented software projects. *IEEE Transactions on Software Engineering*, 28(7), 706–720. <https://doi.org/10.1109/TSE.2002.1019484>

- Briand, L. C., Wüst, J. K., Ikonovovski, S. V., & Lounis, H. (1999). Investigating quality factors in object-oriented designs: An industrial case study. *Proceedings - International Conference on Software Engineering*, 345–354. Scopus. <http://dx.doi.org/10.1145/302405.302654>
- Briand, L., Emam, K. E., & Morasca, S. (1996). On the application of measurement theory in software engineering. *Empirical Software Engineering*, 1(1), 61–88. <https://doi.org/10.1007/BF00125812>
- Caldiera, G., Antoniol, G., Fiutem, R., & Lokan, C. (1998). Definition and experimental evaluation of function points for object-oriented systems. *Proceedings Fifth International Software Metrics Symposium. Metrics (Cat. No.98TB100262)*, 167–178. <https://doi.org/10.1109/METRIC.1998.731242>
- Canfora, G., Cerulo, L., Di Penta, M. D., & Pacilio, F. (2010). An Exploratory Study of Factors Influencing Change Entropy. *International Conference on Program Comprehension*, 134–143. Los Alamitos, CA, USA: IEEE Computer Society. <http://dx.doi.org/10.1109/ICPC.2010.32>
- Chapin, N., Hale, J. E., Khan, K. Md., Ramil, J. F., & Tan, W.-G. (2001). Types of software evolution and software maintenance. *Journal of Software Maintenance and Evolution: Research and Practice*, 13(1), 3–30. <http://dx.doi.org/10.1002/smr.220>
- Chidamber, S. R., & Kemerer, C. F. (1994). A metrics suite for object oriented design. *IEEE Transactions on Software Engineering*, 20(6), 476–493. <http://dx.doi.org/10.1109/32.295895>
- Clarke, J., Dolado, J. J., Harman, M., Hierons, R., Jones, B., Lumkin, M., ... Shepperd, M. (2003). Reformulating software engineering as a search problem. *IEE Proceedings - Software*, 150(3), 161–175. <https://doi.org/10.1049/ip-sen:20030559>
- Coleman, D., Ash, D., Lowther, B., & Oman, P. (1994). Using Metrics to Evaluate Software System Maintainability. *Computer*, 27(8), 44–49. <http://dx.doi.org/10.1109/2.303623>
- Cook, C. R., & Roesch, A. (1994). Real-time Software Metrics. *J. Syst. Softw.*, 24(3), 223–237. [https://doi.org/10.1016/0164-1212\(94\)90065-5](https://doi.org/10.1016/0164-1212(94)90065-5)
- Cook, T. D., & Campbell, D. T. (1979). *Quasi-experimentation: Design & Analysis Issues for Field Settings*. Boston: Houghton Mifflin.
- Cooper, K., & Torczon, L. (2011). *Engineering a Compiler, Second Edition* (2 edition). Amsterdam ; Boston: Morgan Kaufmann.
- Cormen, T. H., Leiserson, C. E., Rivest, R. L., & Stein, C. (2009). *Introduction to Algorithms, 3rd Edition* (3rd ed.). Cambridge, Mass.: The MIT Press.
- Dagpinar, M., & Jahnke, J. H. (2003). Predicting maintainability with object-oriented metrics -an empirical comparison. *10th Working Conference on Reverse Engineering, 2003. WCRE 2003. Proceedings.*, 155–164. <http://dx.doi.org/10.1109/WCRE.2003.1287246>
- Dascalu, S., Hao, N., & Debnath, N. (2005). Design patterns automation with template library. *Proceedings of the Fifth IEEE International Symposium on Signal Processing and Information Technology, 2005*, 699–705. <http://dx.doi.org/10.1109/ISSPIT.2005.1577183>
- de Fraça, B. B. N., & Travassos, G. H. (2016). Experimentation with dynamic simulation models in software engineering: Planning and reporting guidelines. *Empirical Software Engineering*, 21(3), 1302–1345. Scopus. <https://doi.org/10.1007/s10664-015-9386-4>
- Demeyer, S., Mens, T., & Wermelinger, M. (2001). Towards a Software Evolution Benchmark. *Proceedings of the 4th International Workshop on Principles of Software Evolution*, 174–177. New York, NY, USA: ACM. <https://doi.org/10.1145/602461.602502>
- Dolado, J. J. (2001). On the problem of the software cost function. *Information and Software Technology*, 1(43), 61–72.
- Dubey, S. K., & Rana, A. (2011). Assessment of Maintainability Metrics for Object-oriented Software System. *SIGSOFT Softw. Eng. Notes*, 36(5), 1–7. <http://dx.doi.org/10.1145/2020976.2020983>
- Durrett, R. (2010). *Probability: Theory and Examples* (4th edition). Cambridge ; New York: Cambridge University Press.
- Eden, A. H., & Mens, T. (2006). Measuring software flexibility. *IEE Proceedings - Software*, 153(3), 113. <http://dx.doi.org/10.1049/ip-sen:20050045>
- Elrad, T., Filman, R. E., & Bader, A. (2001). Aspect-oriented Programming: Introduction. *Commun. ACM*, 44(10), 29–32. <http://dx.doi.org/10.1145/383845.383853>
- Eveleens, J. L., & Verhoef, C. (2009). Quantifying IT forecast quality. *Science of Computer Programming*, 74(11–12), 934–988. Scopus. <https://doi.org/10.1016/j.scico.2009.09.005>

- Fenton, N. E., & Pfleeger, S. L. (1998). *Software Metrics: A Rigorous and Practical Approach* (2nd ed.). Boston, MA, USA: PWS Publishing Co.
- Fenton, N., & Melton, A. (1990). Deriving Structurally Based Software Measures. *Journal of System and Software*, 12(3), 177–187. [http://dx.doi.org/10.1016/0164-1212\(90\)90038-N](http://dx.doi.org/10.1016/0164-1212(90)90038-N)
- Fenves, G., McKena, F., Scott, M., & Takahashi, Y. (2004). An object-oriented software environment for collaborative network simulation. *Proceedings of the 13th World Conference on Earthquake Engineering*. Vancouver, Canada.
- Filman, R. E., Elrad, T., Clarke, S., & Aksit, M. (2004). *Aspect-Oriented Software Development* (1st ed.). Harlow: Addison-Wesley Professional.
- Fioravanti, F., & Nesi, P. (2001). Estimation and Prediction Metrics for Adaptive Maintenance Effort of Object-Oriented Systems. *IEEE Trans. Softw. Eng.*, 27(12), 1062–1084. <http://dx.doi.org/10.1109/32.988708>
- Gall, H., Jazayeri, M., Klosch, R. R., & Trausmuth, G. (1997). Software evolution observations based on product release history. *1997 Proceedings International Conference on Software Maintenance*, 160–166. <https://doi.org/10.1109/ICSM.1997.624242>
- Gall, Harald, Hajek, K., & Jazayeri, M. (1998). Detection of logical coupling based on product release history. *Proceedings of the International Conference on Software Maintenance*, 190–198. Washington, DC, USA: IEEE Computer Society. <https://doi.org/10.1109/ICSM.1998.738508>
- Gamma, E., Helm, R., Johnson, R., & Vlissides, J. (1994). *Design Patterns: Elements of Reusable Object-Oriented Software*. Pearson Education.
- Gibbons, J. (2006). Design Patterns As Higher-order Datatype-generic Programs. *Proceedings of the 2006 ACM SIGPLAN Workshop on Generic Programming*, 1–12. New York, NY, USA: ACM. <https://doi.org/10.1145/1159861.1159863>
- Gibbons, J., & Oliveira, B. C. d. S. (2009). The essence of the Iterator pattern. *Journal of Functional Programming*, 19(3–4), 377–402. <http://dx.doi.org/10.1017/S0956796809007291>
- Glass, R. L. (2002). *Software Engineering: Facts and Fallacies*. Boston, MA, USA: Addison-Wesley Longman Publishing Co., Inc.
- Granja-Alvarez, J. C., & Barranco-García, M. J. (1997). A Method for Estimating Maintenance Cost in a Software Project: A Case Study. *Journal of Software Maintenance*, 9(3), 161–175. [http://dx.doi.org/10.1002/\(SICI\)1096-908X\(199705\)9:3<161::AID-SMR148>3.0.CO;2-8](http://dx.doi.org/10.1002/(SICI)1096-908X(199705)9:3<161::AID-SMR148>3.0.CO;2-8)
- Guderlei, R., Mayer, J., Schneckenburger, C., & Fleischer, F. (2007). *Testing randomized software by means of statistical hypothesis tests*. 46–54. Scopus. <https://doi.org/10.1145/1295074.1295084>
- Hannay, J., & Jørgensen, M. (2008). The Role of Deliberate Artificial Design Elements in Software Engineering Experiments. *IEEE Transactions on Software Engineering*, 34(2), 242–259. <https://doi.org/10.1109/TSE.2008.13>
- Hassan, A. E. (2009). Predicting Faults Using the Complexity of Code Changes. *Proceedings of the 31st International Conference on Software Engineering*, 78–88. Washington, DC, USA: IEEE Computer Society. <http://dx.doi.org/10.1109/ICSE.2009.5070510>
- Hayes, J. H., Patel, S. C., & Zhao, L. (2004). A metrics-based software maintenance effort model. *Eighth European Conference on Software Maintenance and Reengineering, 2004. CSMR 2004. Proceedings.*, 254–258. <http://dx.doi.org/10.1109/CSMR.2004.1281427>
- Hayes, J. H., & Zhao, L. (2005). Maintainability prediction: A regression analysis of measures of evolving systems. *21st IEEE International Conference on Software Maintenance (ICSM'05)*, 601–604. <http://dx.doi.org/10.1109/ICSM.2005.59>
- Heitlager, I., Kuipers, T., & Visser, J. (2007a). A Practical Model for Measuring Maintainability. *Proceedings of the 6th International Conference on Quality of Information and Communications Technology*, 30–39. Washington, DC, USA: IEEE Computer Society. <http://dx.doi.org/10.1109/QUATIC.2007.8>
- Heitlager, I., Kuipers, T., & Visser, J. (2007b). A Practical Model for Measuring Maintainability. *Proceedings of the 6th International Conference on Quality of Information and Communications Technology*, 30–39. Washington, DC, USA: IEEE Computer Society. <http://dx.doi.org/10.1109/QUATIC.2007.7>
- Heng, P. B. C., & Mackie, I. R. (2009). Using design patterns in object-oriented finite element programming. *Computers & Structures*, 87(15–16), 952–961. <http://dx.doi.org/10.1016/j.compstruc.2008.04.016>
- Hills, M., Klint, P., Van Der Storm, T., & Vinju, J. (2011). A Case of Visitor Versus Interpreter Pattern. *Proceedings of the 49th International Conference on Objects, Models, Components*,

- Patterns*, 228–243. Berlin, Heidelberg: Springer-Verlag. [http://dx.doi.org/10.1007/978-3-642-21952-8\\_17](http://dx.doi.org/10.1007/978-3-642-21952-8_17)
- ISO/IEC 25010. (2011). ISO/IEC 25010:2011—Systems and software engineering—Systems and software Quality Requirements and Evaluation (SQuaRE)—System and software quality models. ISO. Retrieved from [http://www.iso.org/iso/home/store/catalogue\\_tc/catalogue\\_detail.htm?csnumber=35733](http://www.iso.org/iso/home/store/catalogue_tc/catalogue_detail.htm?csnumber=35733)
- ISO/IEC 25023. (2016). ISO/IEC 25023:2016—Systems and software engineering—Systems and software Quality Requirements and Evaluation (SQuaRE)—Measurement of system and software product quality. ISO. Retrieved from [http://www.iso.org/iso/home/store/catalogue\\_tc/catalogue\\_detail.htm?csnumber=35747](http://www.iso.org/iso/home/store/catalogue_tc/catalogue_detail.htm?csnumber=35747)
- ISO/IEC/IEEE 24765. (2010). ISO/IEC/IEEE 24765:2010—Systems and software engineering – Vocabulary. *ISO/IEC/IEEE*, 1–418. <http://dx.doi.org/10.1109/IEEESTD.2010.5733835>
- Jabangwe, R., Börstler, J., Šmite, D., & Wohlin, C. (2015). Empirical evidence on the link between object-oriented measures and external quality attributes: A systematic literature review. *Empirical Software Engineering*, 20(3), 640–693. <https://doi.org/10.1007/s10664-013-9291-7>
- Jacobson, I. (1992). *Object Oriented Software Engineering: A Use Case Driven Approach* (1 edition). New York : Wokingham, Eng. ; Reading, Mass: Addison-Wesley Professional.
- Jahnke, J., & Zündorf, A. (1997). Rewriting poor Design Patterns by good Design Patterns. *Proc. ESEC/FSE '97 Workshop ObjectOriented Reeng.*
- Jaynes, E. T. (2003). *Probability Theory: The Logic of Science* (1 edition; G. L. Bretthorst, Ed.). Cambridge, UK ; New York, NY: Cambridge University Press.
- Jazayeri, M. (2002). On Architectural Stability and Evolution. *Reliable Software Technologies – Ada-Europe 2002*, 13–23. Springer, Berlin, Heidelberg. [https://doi.org/10.1007/3-540-48046-3\\_2](https://doi.org/10.1007/3-540-48046-3_2)
- Karanikolas, C., Dimitroulakos, G., & Masselos, K. (2017). Early Evaluation of Implementation Alternatives of Composite Data Structures Toward Maintainability. *ACM Trans. Softw. Eng. Methodol.*, 26(2), 8:1-8:44. <https://doi.org/10.1145/3132731>
- Karanikolas, C., Dimitroulakos, G., & Masselos, K. (2021). Experimental Results of Formal and Simulated Models' Outcomes for CIBIvsCVP General Problem. Retrieved from [http://www.chriskaranikolas.gr/CIBIvsCVP/Results\\_CIBI\\_CVP\\_7\\_SimulationState.xls](http://www.chriskaranikolas.gr/CIBIvsCVP/Results_CIBI_CVP_7_SimulationState.xls)
- Karanikolas, C., Dimitroulakos, G., & Masselos, K. (n.d.-a). Dynamic Formal comparison Model implementation of CIBIvsCVP General Problem. Retrieved from Open research domain website: <http://www.chriskaranikolas.gr/CIBIvsCVP/>
- Karanikolas, C., Dimitroulakos, G., & Masselos, K. (n.d.-b). MatLab macro of Modeling Framework implementation for CIBIvsCVP plus DP and MPvsAF general designing problem. Retrieved from Open research domain website: [http://www.chriskaranikolas.gr/CIBIvsCVP/MatLab\\_CIBIvsCVP\\_DP\\_MP\\_OP\\_General\\_Designing\\_Problems.rar](http://www.chriskaranikolas.gr/CIBIvsCVP/MatLab_CIBIvsCVP_DP_MP_OP_General_Designing_Problems.rar)
- Kayarvizhy, N., Kanmani, S., & Uthariaraj, V. R. (2013). High precision cohesion metric. *WSEAS Transactions on Information Science and Applications*, 10(3), 79–89.
- Keller, R. K., Schauer, R., Robitaille, S., & Pagé, P. (1999). Pattern-based Reverse-engineering of Design Components. *Proceedings of the 21st International Conference on Software Engineering*, 226–235. New York, NY, USA: ACM. <http://dx.doi.org/10.1145/302405.302622>
- Kelsen, P. (2004). A simple static model for understanding the dynamic behavior of programs. *Proceedings. 12th IEEE International Workshop on Program Comprehension, 2004.*, 46–51. <https://doi.org/10.1109/WPC.2004.1311046>
- Kemerer, C. F., & Slaughter, S. (1999). An empirical approach to studying software evolution. *IEEE Transactions on Software Engineering*, 25(4), 493–509. <https://doi.org/10.1109/32.799945>
- Krishnamurthi, S., Felleisen, M., & Friedman, D. P. (1998). Synthesizing Object-Oriented and Functional Design to Promote Re-Use. *Proceedings of the 12th European Conference on Object-Oriented Programming*, 91–113. London, UK, UK: Springer-Verlag. <http://dx.doi.org/10.1007/BFb0054088>
- Kruchten, P., Nord, R. L., & Ozkaya, I. (2012). Technical Debt: From Metaphor to Theory and Practice. *IEEE Software*, 29(6), 18–21. <https://doi.org/10.1109/MS.2012.167>
- Kumar, L., Krishna, A., & Rath, S. K. (2017). The impact of feature selection on maintainability prediction of service-oriented applications. *Service Oriented Computing and Applications*, 11(2), 137–161. <http://dx.doi.org/10.1007/s11761-016-0202-9>

- Land, R. (2002). Measurements of Software Maintainability. *Proceedings of ARTES Graduate Student Conference (Neither Reviewed nor Officially Published)*, ARTES.
- Langdon, W. B., Dolado, J., Sarro, F., & Harman, M. (2016). Exact Mean Absolute Error of Baseline Predictor, MARPO. *Inf. Softw. Technol.*, 73(C), 16–18. <https://doi.org/10.1016/j.infsof.2016.01.003>
- Larman, C. (2004). *Applying UML and Patterns: An Introduction to Object-Oriented Analysis and Design and Iterative Development* (3rd ed.). Harlow: Prentice Hall.
- Lavazza, L., & Agostini, A. (2005). Automated Measurement of UML Models: An open toolset approach. *Journal of Object Technology*, 4, 115–134. <http://dx.doi.org/10.5381/jot.2005.4.4.a2>
- Lehman, M. M., Perry, D. E., & Ramil, J. F. (1998). On evidence supporting the FEAST hypothesis and the laws of software evolution. *Proceedings Fifth International Software Metrics Symposium. Metrics (Cat. No.98TB100262)*, 84–88. <https://doi.org/10.1109/METRIC.1998.731229>
- Lehman, Meir M., & Ramil, J. F. (2002). Software Evolution and Software Evolution Processes. *Annals of Software Engineering*, 14(1–4), 275–309. <http://dx.doi.org/10.1023/A:1020557525901>
- Lehman, Meir M., Ramil, J. F., Wernick, P. D., Perry, D. E., & Turski, W. M. (1997). Metrics and laws of software evolution—the nineties view. *Proceedings Fourth International Software Metrics Symposium*, 20–32. IEEE Computer Society. <http://dx.doi.org/10.1109/METRIC.1997.637156>
- Li, W. (1998). Another Metric Suite for Object-oriented Programming. *Journal of Systems and Software*, 44(2), 155–162. [http://dx.doi.org/10.1016/S0164-1212\(98\)10052-3](http://dx.doi.org/10.1016/S0164-1212(98)10052-3)
- Li, W., & Henry, S. (1993). Object-oriented Metrics That Predict Maintainability. *Journal of Systems and Software*, 23(2), 111–122. [http://dx.doi.org/10.1016/0164-1212\(93\)90077-B](http://dx.doi.org/10.1016/0164-1212(93)90077-B)
- Liu, W., Tong, M., Wu, X., & Lee, G. (2003). Object-Oriented Modeling of Structural Analysis and Design with Application to Damping Device Configuration. *Journal of Computing in Civil Engineering*, 17(2), 113–122. [http://dx.doi.org/10.1061/\(ASCE\)0887-3801\(2003\)17:2\(113\)](http://dx.doi.org/10.1061/(ASCE)0887-3801(2003)17:2(113))
- Lorenz, M., & Kidd, J. (1994). *Object-Oriented Software Metrics* (1 edition). Englewood Cliffs, NJ: Prentice Hall.
- Mackie, R. I. (2002). *Object-Oriented Methods and Finite Element Analysis*. Kippen, Stirling, UK: Saxe-Coburg Publications.
- Meli, R. (1997). *Early and Extended Function Point: A new method for Function Points estimation*. 15–19. Arizona.
- Melton, A. C., Gustafson, D. A., Bieman, J. M., & Baker, A. L. (1990). A mathematical perspective for software measures research. *Software Engineering Journal*, 5(5), 246–254. <http://dx.doi.org/10.1049/sej.1990.0027>
- Mens, T., & Tourwe, T. (2001). A declarative evolution framework for object-oriented design patterns. *Proceedings IEEE International Conference on Software Maintenance. ICSM 2001*, 570–579. <https://doi.org/10.1109/ICSM.2001.972774>
- Mens, T., & Tourwe, T. (2004). A survey of software refactoring. *IEEE Transactions on Software Engineering*, 30(2), 126–139. <https://doi.org/10.1109/TSE.2004.1265817>
- Mens, Tom, & Demeyer, S. (2001). Future Trends in Software Evolution Metrics. *Proceedings of the 4th International Workshop on Principles of Software Evolution*, 83–86. New York, NY, USA: ACM. <http://dx.doi.org/10.1145/602461.602476>
- Mens, Tom, & Eden, A. H. (2005). On the Evolution Complexity of Design Patterns. *Electronic Notes in Theoretical Computer Science*, 127(3), 147–163. <http://dx.doi.org/10.1016/j.entcs.2004.08.041>
- Montgomery, D. C. (2012). *Design and Analysis of Experiments* (8th ed.). John Wiley & Sons, Incorporated.
- Müller, M., & Pfahl, D. (2008). Simulation Methods. In F. Shull, J. Singer, & D. I. K. Sjøberg (Eds.), *Guide to Advanced Empirical Software Engineering* (pp. 117–152). London: Springer London. [https://doi.org/10.1007/978-1-84800-044-5\\_5](https://doi.org/10.1007/978-1-84800-044-5_5)
- Musilek, P., Pedrycz, W., Nan Sun, & Succi, G. (2002). On the sensitivity of COCOMO II software cost estimation model. *Proceedings Eighth IEEE Symposium on Software Metrics*, 13–20. <https://doi.org/10.1109/METRIC.2002.1011321>

- Neff, N. (1999). OO design in compiling an OO language. *SIGCSE Bulletin (Association for Computing Machinery, Special Interest Group on Computer Science Education)*, 31(1), 326–330. Scopus. <http://dx.doi.org/10.1145/384266.299798>
- Neff, N. (2004). Attribute Based Compiler Implemented Using Visitor Pattern. *Proceedings of the 35th SIGCSE Technical Symposium on Computer Science Education*, 130–134. New York, NY, USA: ACM. <http://dx.doi.org/10.1145/971300.971347>
- Offutt, J., Abdurazik, A., & Schach, S. R. (2008). Quantitatively Measuring Object-oriented Couplings. *Software Quality Journal*, 16(4), 489–512. <http://dx.doi.org/10.1007/s11219-008-9051-x>
- Oliveira, B. C. (2009). Modular Visitor Components. *Proceedings of the 23rd European Conference on ECOOP 2009 – Object-Oriented Programming*, 269–293. Berlin, Heidelberg: Springer-Verlag. [https://dx.doi.org/10.1007/978-3-642-03013-0\\_13](https://dx.doi.org/10.1007/978-3-642-03013-0_13)
- Oliveira, B. C. d. S., & Cook, W. R. (2012). Extensibility for the Masses: Practical Extensibility with Object Algebras. *Proceedings of the 26th European Conference on Object-Oriented Programming*, 2–27. Berlin, Heidelberg: Springer-Verlag. [http://dx.doi.org/10.1007/978-3-642-31057-7\\_2](http://dx.doi.org/10.1007/978-3-642-31057-7_2)
- Oliveira, B. C. d. S., Wang, M., & Gibbons, J. (2008). The Visitor Pattern As a Reusable, Generic, Type-safe Component. *Proceedings of the 23rd ACM SIGPLAN Conference on Object-Oriented Programming Systems Languages and Applications*, 439–456. New York, NY, USA: ACM. <http://dx.doi.org/10.1145/1449764.1449799>
- Oman, P., & Hagemester, J. (1994). Construction and testing of polynomials predicting software maintainability. *Journal of Systems and Software*, 24(3), 251–266. [http://dx.doi.org/10.1016/0164-1212\(94\)90067-1](http://dx.doi.org/10.1016/0164-1212(94)90067-1)
- Ostberg, J.-P., & Wagner, S. (2014). On Automatically Collectable Metrics for Software Maintainability Evaluation. *2014 Joint Conference of the International Workshop on Software Measurement and the International Conference on Software Process and Product Measurement*, 32–37. <https://doi.org/10.1109/IWSM.Mensura.2014.19>
- Paixao, M., Krinke, J., Han, D., Ragkhitwetsagul, C., & Harman, M. (2017). Are Developers Aware of the Architectural Impact of Their Changes? *Proceedings of the 32Nd IEEE/ACM International Conference on Automated Software Engineering*, 95–105. Piscataway, NJ, USA: IEEE Press. Retrieved from <http://dl.acm.org/citation.cfm?id=3155562.3155578>
- Palsberg, J., & Jay, C. B. (1998). Essence of the visitor pattern. *IEEE Computer Society's International Computer Software and Applications Conference (COMPSAC '98)*, 9–15. Washington, DC, USA: IEEE Computer Society. Scopus. <http://dx.doi.org/10.1109/COMPSAC.1998.716629>
- Parnas, D. L., & Curtis, B. (2009). Point/Counterpoint: Are rigorous experiments realistic for software engineering? *IEEE Software*, 26(6), 56–59. <https://doi.org/10.1109/MS.2009.184>
- Parnas, David Lorge. (1994). Software Aging. *Proceedings of the 16th International Conference on Software Engineering*, 279–287. Los Alamitos, CA, USA: IEEE Computer Society Press. Retrieved from <http://dl.acm.org/citation.cfm?id=257734.257788>
- Parr, T. (2013). *The Definitive ANTLR 4 Reference* (2nd ed.). Dallas, Texas: Pragmatic Bookshelf.
- Pressman, R. (2010). *Software Engineering: A Practitioner's Approach* (7th ed.). New York, NY, USA: McGraw-Hill, Inc.
- Pressman, R. S. (2001). *Software Engineering: A Practitioner's Approach* (5th ed.). McGraw-Hill Higher Education.
- Raja, U., Hale, D. P., & Hale, J. E. (2009). Modeling software evolution defects: A time series approach. *Journal of Software Maintenance and Evolution: Research and Practice*, 21(1), 49–71.
- Ramil, J. F., & Lehman, M. M. (2000). Metrics of software evolution as effort predictors—A case study. *Proceedings 2000 International Conference on Software Maintenance*, 163–172. Washington, DC, USA: IEEE Computer Society. <https://doi.org/10.1109/ICSM.2000.883036>
- Ramil, J. F., & Lehman, M. M. (2001). Defining and applying metrics in the context of continuing software evolution. *Proceedings of the 7th International Symposium on Software Metrics*, 199–209. Washington, DC, USA: IEEE Computer Society. <https://doi.org/10.1109/METRIC.2001.915529>
- Riaz, M., Mendes, E., & Tempero, E. (2009a). A systematic review of software maintainability prediction and metrics. *2009 3rd International Symposium on Empirical Software Engineering and Measurement*, 367–377. Washington, DC, USA: IEEE Computer Society. <http://dx.doi.org/10.1109/ESEM.2009.5314233>

- Riaz, M., Mendes, E., & Tempero, E. (2009b). A Systematic Review of Software Maintainability Prediction and Metrics. *Proceedings of the 2009 3rd International Symposium on Empirical Software Engineering and Measurement*, 367–377. Washington, DC, USA: IEEE Computer Society. <http://dx.doi.org/10.1109/ESEM.2009.5314233>
- Riehle, D. (2009). Design Pattern Density Defined. *Proceedings of the 24th ACM SIGPLAN Conference on Object Oriented Programming Systems Languages and Applications*, 469–480. New York, NY, USA: ACM. <https://doi.org/10.1145/1640089.1640125>
- Rizvi, S. W. A., & Khan, R. A. (2010). Maintainability Estimation Model for Object-Oriented Software in Design Phase (MEMOOD). *Journal of Computing*, 2(4). Retrieved from <http://arxiv.org/abs/1004.4447>
- Rubinstein, R. Y., & Kroese, D. P. (2016). *Simulation and the Monte Carlo Method* (3 edition). Wiley.
- Santos Oliveira, B. C. (2007). *Genericity, Extensibility and Type-Safety in the VISITOR Pattern*. (PhD Thesis, University of Oxford). University of Oxford. Retrieved from <http://www.cs.ox.ac.uk/people/bruno.oliveira/Thesis.pdf>
- Schildt, H. (2002). *C++: The Complete Reference, 4th Edition* (4th ed.). New York: McGraw-Hill Osborne Media.
- Shariat Yazdi, H., Angelis, L., Kehrer, T., & Kelter, U. (2016). A framework for capturing, statistically modeling and analyzing the evolution of software models. *Journal of Systems and Software*, 118(C), 176–207. <https://doi.org/10.1016/j.jss.2016.05.010>
- Shepperd, M., & MacDonell, S. (2012). Evaluating prediction systems in software project estimation. *Information and Software Technology*, 54(8), 820–827. <https://doi.org/10.1016/j.infsof.2011.12.008>
- Sommerville, I. (2010). *Software Engineering* (9th ed.). Boston: Addison-Wesley.
- Srinivasan, K. P., & Devi, N. T. (2014). A complete and comprehensive metrics suite for object-oriented design quality assessment. *International Journal of Software Engineering and Its Applications*, 8(2), 173–188. Scopus. <http://dx.doi.org/10.14257/ijseia.2014.8.2.17>
- Srivastava, B. (2004). A decision-support framework for component reuse and maintenance in software project management. *Eighth European Conference on Software Maintenance and Reengineering, 2004. CSMR 2004. Proceedings.*, 125–134. <https://doi.org/10.1109/CSMR.2004.1281413>
- Stewart, J. (2015). *Calculus, Early Transcendentals, International Metric Edition* (8 edition). Brooks Cole.
- Stol, K.-J., & Fitzgerald, B. (2018). The ABC of Software Engineering Research. *ACM Trans. Softw. Eng. Methodol.*, 27(3), 11:1-11:51. <https://doi.org/10.1145/3241743>
- Stol, K.-J., Goedicke, M., & Jacobson, I. (2016). Introduction to the special section—General Theories of Software Engineering: New advances and implications for research. *Information and Software Technology*, 70, 176–180. <https://doi.org/10.1016/j.infsof.2015.07.010>
- Stopford, B., & Counsell, S. (2008). A Framework for the Simulation of Structural Software Evolution. *ACM Trans. Model. Comput. Simul.*, 18(4), 17:1-17:36. <https://doi.org/10.1145/1391978.1391983>
- Subramanyam, K. R. (2013). *Financial Statement Analysis* (11th ed.). McGraw-Hill Publishing. Retrieved from <https://www.mheducation.com/highered/product/financial-statement-analysis-subramanyam/M9780078110962.html>
- Subramanyam, R., & Krishnan, M. S. (2003). Empirical analysis of CK metrics for object-oriented design complexity: Implications for software defects. *IEEE Transactions on Software Engineering*, 29(4), 297–310. <http://dx.doi.org/10.1109/TSE.2003.1191795>
- Torgersen, M. (2004). The Expression Problem Revisited—Four new solutions using generics. *Proceedings of the 18th European Conference on Object-Oriented Programming*, 123–146. Springer-Verlag. [http://dx.doi.org/10.1007/978-3-540-24851-4\\_6](http://dx.doi.org/10.1007/978-3-540-24851-4_6)
- Turver, R. J., & Munro, M. (1994). An early impact analysis technique for software maintenance. *Journal of Software Maintenance: Research and Practice*, 6(1), 35–52. Scopus. <https://doi.org/10.1002/smr.4360060104>
- VanDrunen, T., & Palsberg, J. (2004). Visitor-oriented Programming. *Proceedings of FOOL-11, 11th ACM SIGPLAN International Workshop on Foundations of Object-Oriented Languages*. Presented at the 11th ACM SIGPLAN International Workshop on Foundations of Object-Oriented Languages, New York, NY, USA. New York, NY, USA: ACM Press.

- Visser, J. (2001). Visitor combination and traversal control. *Proceedings of the Conference on Object-Oriented Programming Systems, Languages, and Applications, OOPSLA, 36*, 270–282. New York, NY, USA: ACM SIGPLAN. Scopus. <http://dx.doi.org/10.1145/504311.504302>
- Völter, M. (2003). A Catalog of Patterns for Program Generation. In K. Henney & D. Schütz (Eds.), *Proceedings of the 8th European Conference on Pattern Languages of Programms (EuroPLoP '2003), Irsee, Germany, June 25-29, 2003* (pp. 285–320). UVK - Universitätsverlag Konstanz.
- Wadler, P. (1998). *The expression problem*. Posted on the Java Genericity mailing list.
- Wang, Y., & Oliveira, B. C. d. S. (2016). The Expression Problem, Trivially! *Proceedings of the 15th International Conference on Modularity*, 37–41. New York, NY, USA: ACM. <https://dx.doi.org/10.1145/2889443.2889448>
- Williams, B. J., & Carver, J. C. (2010). Characterizing Software Architecture Changes: A Systematic Review. *Inf. Softw. Technol.*, 52(1), 31–51. <https://doi.org/10.1016/j.infsof.2009.07.002>
- Wohlin, C., Runeson, P., Höst, M., Ohlsson, M. C., Regnell, B., & Wesslén, A. (2012). *Experimentation in Software Engineering*. Springer Publishing Company. <https://doi.org/10.1007/978-3-642-29044-2>
- Woolf, M. (2016). Visualizing How Developers Rate Their Own Programming Skills. Retrieved May 12, 2018, from Maximaxir | Max Woolf's Blog website: <http://minimaxir.com/2016/07/stackoverflow/>
- Wu, H., Shi, L., Chen, C., Wang, Q., & Boehm, B. (2016). Maintenance Effort Estimation for Open Source Software: A Systematic Literature Review. *2016 IEEE International Conference on Software Maintenance and Evolution (ICSME)*, 32–43. <https://doi.org/10.1109/ICSME.2016.87>
- Xiao, L., Cai, Y., Kazman, R., Mo, R., & Feng, Q. (2016). Identifying and Quantifying Architectural Debt. *2016 IEEE/ACM 38th International Conference on Software Engineering (ICSE)*, 488–498. <https://doi.org/10.1145/2884781.2884822>
- Ye, Q. K. (1998). Orthogonal column latin hypercubes and their application in computer experiments. *Journal of the American Statistical Association*, 93(444), 1430–1439.
- Yuen, C. K. S. C. H. (1988). On analytic maintenance process data at the global and the detailed levels: A case study. *Proceedings. Conference on Software Maintenance, 1988.*, 248–255. <https://doi.org/10.1109/ICSM.1988.10170>
- Zdun, U., & Strembeck, M. (2009). Reusable architectural decisions for DSL design: Foundational decisions in DSL development. *Proceedings of EuroPLoP 2009 - 14th Annual European*. Presented at the Conference on Pattern Languages of Programming. Scopus. Retrieved from Scopus.
- Zeigler, B. P., Mittal, S., & Traore, M. K. (2018). MBSE with/out Simulation: State of the Art and Way Forward. *Systems*, 6(4), 40. <https://doi.org/10.3390/systems6040040>
- Zenger, M., & Odersky, M. (2005). Independently Extensible Solutions to the Expression Problem. *Workshop on Foundations of Object-Oriented Languages (FOOL)*. Long Beach, California (in conjunction with POPL).
- Zhang, J. (2008). The Establishment and Application of Effort Regression Equation. *2008 International Conference on Computer Science and Software Engineering*, 2, 11–14. <https://doi.org/10.1109/CSSE.2008.726>



**Author’s Statement:**

I hereby expressly declare that, according to the article 8 of Law 1559/1986, this dissertation is solely the product of my personal work, does not infringe any intellectual property, personality and personal data rights of third parties, does not contain works/contributions from third parties for which the permission of the authors/beneficiaries is required, is not the product of partial or total plagiarism, and that the sources used are limited to the literature references alone and meet the rules of scientific citations.