



ΠΑΝΕΠΙΣΤΗΜΙΟ ΠΕΛΟΠΟΝΝΗΣΟΥ

Σχολή Θετικών Επιστημών και Τεχνολογίας

Τμήμα Επιστήμης και Τεχνολογίας Υπολογιστών

ΔΙΠΛΩΜΑΤΙΚΗ ΕΡΓΑΣΙΑ

**ΣΥΣΤΗΜΑ ΔΙΑΧΕΙΡΙΣΗΣ ΚΑΙ ΟΠΤΙΚΟΠΟΙΗΣΗΣ
ΠΡΟΣΩΠΙΚΗΣ ΠΛΗΡΟΦΟΡΙΑΣ ΒΑΣΙΖΟΜΕΝΟ ΣΕ ΟΝΤΟΛΟΓΙΕΣ**

ΡΟΜΠΑ ΕΥΓΕΝΙΑ

**Επιβλέποντες: Λέπουρας Γεώργιος
Βασιλάκης Κωνσταντίνος
Τρυφωνόπουλος Χρήστος**

ΤΡΙΠΟΛΗ, 2012



University of Peloponnese
Faculty of Science and Technology
Department of Computer Science and Technology

THESIS

PERSONAL INFORMATION MANAGEMENT AND VISUALIZATION
ONTOLOGY-BASED SYSTEM

ROMPA EVGENIA

Supervisors: George Lepouras
Costas Vassilakis
Christos Tryfonopoulos

TRIPOLIS, 2011

*Dedicated to my parents,
for their unconditional love,
support and encouragement.*

ACKNOWLEDGEMENTS

There are many individuals who contributed to the production of this thesis through their moral support, advice or participation. I would like to express my most sincere gratitude to them all.

First of all, I would like to express my appreciation to my advisory committee: Dr. George Lepouras, Dr. Costas Vassilakis, and Dr. Christos Tryfonopoulos. Thank you for your guidance, patience, careful supervision and encouragement throughout the years. It has been both a privilege and a pleasure to have experienced the opportunity to be taught by such extraordinary people. I sincerely thank you for being the sort of supervisor every student needs - supportive, enthusiastic and inspiring.

I would also like to extend my gratitude to the University of Peloponnese. Through the support offered by granting, I have been able to obtain the Master of Science and travel to remarkable conferences in order to present my research in progress.

Lastly, would like to thank my family for their understanding, encouragement and support. They have always been the greatest support through the whole process, encouraging me to keep going when frustration and other problems drained my motivation. Their support not only emotionally if not financially as well as their love, help and encouragement cannot be thanks enough.

March 2012

Evgenia Rompa

SUMMARY

Users nowadays need to manage large amounts of information, including documents, e-mails, contacts, multimedia and so forth. To facilitate the users' tasks regarding the organization, maintenance and retrieval of this information, a number of semantics-based methods have emerged, employing mainly ontologies as an underlying infrastructure for organizing and querying the information base. However this is not sufficient as users need tools that would exploit the potential offered by the ontology to help them perform their information management and retrieval tasks. Mind maps can offer valuable assistance to users in this context, by graphically representing the information, focusing on concepts related to the search keywords entered by the user, and allowing the user to navigate through the semantic links between concepts and information items.

Within this thesis, we present OntoFM, a personal ontology-based file manager, offering a mind map-oriented visualization to support user navigation within the personal information base. Browsing aids are complemented with search mechanisms, while the user is also offered with the potential to establish links between ontology concepts and files-directories, offering thus a fully-functional personal information management tool.

Keywords: Ontology, Mind maps, Protégé, Plug-in, Personal Information Management

ΠΕΡΙΛΗΨΗ

Στις μέρες μας οι χρήστες έχουν την ανάγκη να διαχειριστούν μεγάλες ποσότητες πληροφοριών, συμπεριλαμβανομένων εγγράφων, e-mail, επαφών, πολυμέσων και άλλων. Για τη διευκόλυνση των διεργασιών των χρηστών σχετικά με την οργάνωση, τη συντήρηση και την ανάκτηση των εν λόγω πληροφοριών, ένα πλήθος σημασιολογικά βασιζόμενων μεθόδων προέκυψε, χρησιμοποιώντας κυρίως οντολογίες ως θεμέλιο για την οργάνωση και την αναζήτηση της βάσης πληροφοριών. Ωστόσο αυτό δεν αρκεί, καθώς οι χρήστες χρειάζονται εργαλεία που θα αξιοποιήσουν τις δυνατότητες που προσφέρονται από την οντολογία, για να τους βοηθήσει στη διαχείριση πληροφοριών και τη διαδικασία ανάκτησης. Οι χάρτες σκέψης μπορούν να προσφέρουν πολύτιμη βοήθεια στους χρήστες, με τη γραφική αναπαράσταση πληροφοριών, εστιάζοντας σε έννοιες που σχετίζονται με τις λέξεις-κλειδιά που εισάγει ο χρήστης, και επιτρέποντας την πλοήγηση μέσω των σημασιολογικών σχέσεων μεταξύ εννοιών και πληροφοριακών στοιχείων.

Σε αυτή τη διατριβή παρουσιάζουμε τον OntoFM, ένα διαχειριστή αρχείων βασιζόμενο σε προσωπική οντολογία, που προσφέρει γραφική αναπαράσταση με τη μορφή χάρτη σκέψης (mind map) υποστηρίζοντας κατ' αυτόν τον τρόπο την πλοήγηση του χρήστη στην προσωπική του βάση πληροφοριών. Οι προσφερόμενοι μηχανισμοί περιήγησης εμπλουτίζονται με μηχανισμούς αναζήτησης, ενώ υποστηρίζεται η δυνατότητα δημιουργίας δεσμών μεταξύ των εννοιών της οντολογίας και αρχείων ή καταλόγων, προσφέροντας έτσι ένα πλήρως λειτουργικό εργαλείο διαχείρισης προσωπικών πληροφοριών. Η παρούσα διατριβή επεκτείνει προηγούμενη προσέγγιση που αφορά την αξιοποίηση οντολογιών στην ανάκτηση προσωπικής πληροφορίας εισάγοντας μία απεικόνιση με τη μορφή χάρτη σκέψης, χρησιμοποιώντας μια μηχανή ευρετηρίασης και αναζήτησης πλήρους κειμένου και προσφέροντας τη δυνατότητα ενημέρωσης της προσωπικής οντολογίας.

Ο διαχειριστής αρχείων OntoFM έχει κατασκευαστεί σύμφωνα με μία αρχιτεκτονική πολλών επιπέδων έχοντας ως υπόβαθρο τον χώρο αποθήκευσης πληροφοριών που αποτελείται από το σύστημα αρχείων και την προσωπική οντολογία,

η οποία περιέχει τις έννοιες καθώς και τις συσχετίσεις μεταξύ αυτών των εννοιών και των αντικειμένων (αρχείων ή καταλόγων) που βρίσκονται στο σύστημα αρχείων. Για τη διαχείριση και αποθήκευση της οντολογίας χρησιμοποιείται το ανοικτού κώδικα σύστημα επεξεργασίας οντολογιών και βάσης γνώσεων Protégé. Στο αμέσως ανώτερο επίπεδο αρχιτεκτονικής, βρίσκεται το επίπεδο που αφορά λειτουργίες του OntoFM όπως είναι (i) η αναζήτηση πληροφοριών, που αφορά την εύρεση στοιχείων του συστήματος αρχείων, (ii) η διασύνδεση πληροφοριών, που επιτρέπει στο χρήστη τη σύνδεση επιλεγμένων στοιχείων με έννοιες της οντολογίας, και (iii) η προσκόμιση και ενημέρωση δεδομένων από το χώρο αποθήκευση πληροφοριών. Για την αποτελεσματική ευρετηρίαση και ανάκτηση αρχείων και στοιχείων οντολογίας χρησιμοποιείται το Lucene, μία βιβλιοθήκη ανοικτού κώδικα που προσφέρει υψηλής απόδοσης μηχανές αναζήτησης πλήρους κειμένου. Στο ανώτατο στρώμα της αρχιτεκτονικής του OntoFM βρίσκεται η διεπαφή χρήστη η οποία συλλέγει τα δεδομένα εισόδου του χρήστη και εμφανίζει τα αντίστοιχα αποτελέσματα ανάλογα με την λειτουργία που εκτελέσθηκε. Η διεπαφή χρήστη περιλαμβάνει και την οπτικοποίηση της προσωπικής οντολογίας, η οποία βασίζεται στο Ontograf plug-in του Protégé, το οποίο για τις ανάγκες του OntoFM έχει επεκταθεί προκειμένου να υποστηρίξει την πρόσθετη λειτουργικότητα του διαχειριστή αρχείων και να απεικονίσει την οντολογία με μία μορφή που να παραπέμπει σε χάρτες σκέψης. Αυτός ο τρόπος απεικόνισης καθώς και η αλληλεπίδραση του χρήστη, επιτρέπει στους χρήστες το χειρισμό της οντολογίας χωρίς την απαίτηση απόκτησης λεπτομερούς τεχνικής γνώσης για τη διαχείριση οντολογιών. Με αυτό τον τρόπο, ο OntoFM προσφέρει σημασιολογική εκφραστικότητα αξιοποιώντας τους χάρτες σκέψης για τον εμπλουτισμό και την παρουσίαση του χώρου πληροφοριών του χρήστη.

Βασικές λειτουργίες που υποστηρίζονται από όλους τους περιηγητές αρχείων είναι αυτές της περιήγησης και της αναζήτησης σε αρχεία και καταλόγους. Ο OntoFM ως διαχειριστής αρχείων προσφέρει τις δύο αυτές βασικές λειτουργίες εμπλουτισμένες με σημασιολογικό περιεχόμενο. Ο χρήστης έχει τη δυνατότητα να περιηγηθεί στους φυσικούς καταλόγους και τα αρχεία του προσωπικού του υπολογιστή μέσω μιας ιεραρχίας φυσικών καταλόγων που διατίθεται μέσα από τη διεπαφή χρήστη, καθώς και στις έννοιες που συνδέονται με τον εκάστοτε επιλεγμένο κατάλογο. Επίσης, δίνεται στο χρήστη η δυνατότητα της ταυτόχρονης εμφάνισης περιεχομένων πολλαπλών καταλόγων με μία καινοτόμα λειτουργία που του επιτρέπει την πολλαπλή επιλογή

καταλόγων. Η λειτουργία αυτή λειτουργεί και ως φίλτρο αποτελεσμάτων στην περίπτωση όπου ο χρήστης έχει εκτελέσει μία αναζήτηση και θέλει να περιορίσει τα αποτελέσματα, διατηρώντας εμφανή μόνο αυτά που έχουν θέση σε επιλεγμένους καταλόγους. Εκτός από την περιήγηση σε καταλόγους, ο χρήστης έχει και τη δυνατότητα να περιηγηθεί σε έννοιες και στιγμιότυπα της προσωπικής οντολογίας μέσω του πλαισίου οπτικοποίησης οντολογίας. Κάθε έννοια ή στιγμιότυπο αναπαρίσταται με κόμβους οι οποίοι αναπτύσσονται ή συμπύσσονται στο διπλό click του χρήστη, ενώ κάθε συσχέτιση αναπαρίσταται με ένα βέλος που δείχνει τη φορά της συσχέτισης και συνδέει τους δύο κόμβους που εμπλέκονται σε αυτή. Επιπλέον, ο χρήστης μπορεί να ελέγξει την εμφάνιση του γραφήματος της οντολογίας αποκρύπτοντας ή εμφανίζοντας τύπους κόμβων ή συσχετίσεων ή αλλάζοντας τη θέση κόμβων ή γενικότερα τη διάταξη εμφάνισης.

Στην περίπτωση όπου ο χρήστης αναζητά κάποιο αρχείο, ο OnτοFM προσφέρει κατάλληλους μηχανισμούς αναζήτησης εμπλουτισμένους με σημασιολογικό περιεχόμενο κάνοντας τη λειτουργία της αναζήτησης πιο αποδοτική. Ανάλογα με τις πληροφορίες που είναι διαθέσιμες στο χρήστη για το ζητούμενο αρχείο, η λειτουργία της αναζήτησης μπορεί να κατηγοριοποιηθεί σε (i) *τέλεια αναζήτηση*, όπου ο χρήστης γνωρίζει το όνομα και τη θέση του αρχείου, και (ii) *περιορισμένη / ελλιπή / μερική αναζήτηση*, όπου ο χρήστης έχει ελλιπείς πληροφορίες σχετικά με το ζητούμενο αρχείο. Η περίπτωση της τέλει αναζήτησης, μπορεί κατά μία έννοια να αντιστοιχισθεί με την λειτουργία της περιήγησης καθώς ο χρήστης μπορεί να περιηγηθεί στη γνωστή τοποθεσία και στη συνέχεια να βρει το αρχείο με το γνωστό όνομα. Στην περίπτωση που ο χρήστης εκτελεί μία μερική αναζήτηση, οι διαθέσιμες πληροφορίες μπορεί να αφορούν είτε (i) την *τοποθεσία του αρχείου*, αλλά όχι το όνομά του, είτε (ii) το *όνομα του αρχείου*, αλλά όχι τη θέση του, ή τέλος (iii) κάποια *μεταδεδομένα* που μπορεί να αφορούν μία ημερομηνία (δημιουργίας/προσπέλασης κλπ.), κάποιες λέξεις-κλειδιά από το περιεχόμενο, τον συγγραφέα κλπ. Αξίζει να σημειωθεί ότι το βασικό πρόβλημα των κοινών περιηγητών αρχείων προκύπτει στην περίπτωση της ελλιπούς αναζήτησης έχοντας γνωστά κάποια μεταδεδομένα, καθώς το πλήθος των αποτελεσμάτων μπορεί να είναι πολύ μεγάλο και να μην περιέχει αρχεία που θεωρεί ο χρήστης σχετικά με αυτό που αναζητά. Αυτό ίσως να έγκειται στο γεγονός ότι πολλά αρχεία ταιριάζουν με τα κριτήρια αναζήτησης που εισήγαγε ο χρήστης, τα οποία πιθανώς να είναι λανθασμένα, αλλά και στην πιθανότητα το αρχείο που αναζητά να μην έχει διαθέσιμα μεταδεδομένα.

Με την εισαγωγή της σημασιολογικής αναζήτησης, χρησιμοποιώντας μία προσωπική οντολογία ως σημασιολογικό χώρο πληροφοριών, ο διαχειριστής αρχείων OntoFM αμβλύνει αυτό το πρόβλημα και αποδεικνύεται ένα πολύτιμο εργαλείο της διαχείρισης προσωπικών πληροφοριών.

Στην πρώτη περίπτωση της ελλιπούς αναζήτησης όπου μόνο η θέση του αρχείου είναι γνωστή, ο χρήστης μπορεί να μεταβεί στην τοποθεσία και να περιηγηθεί στα περιεχόμενα του καταλόγου προκειμένου να εντοπίσει το εν λόγω αρχείο, ενώ στη δεύτερη και τρίτη περίπτωση ο χρήστης μπορεί να καταφύγει στο μηχανισμό αναζήτησης που προσφέρει ο διαχειριστής αρχείων OntoFM. Αφού ο χρήστης εισάγει τους όρους αναζήτησης που θεωρεί κατάλληλους και εκτελέσει τη λειτουργία της αναζήτησης, η διεπαφή χρήστη εμφανίζει τα αποτελέσματα καθώς και την οπτικοποίηση μέρους της οντολογίας με τη μορφή χάρτη σκέψης. Τα αποτελέσματα εμφανίζονται με φθίνουσα σειρά σχετικότητας που προκύπτει από τον υπολογισμό σχετικότητας των αρχείων με τις λέξεις-κλειδιά που εισήγαγε ο χρήστης, χρησιμοποιώντας το μοντέλο διανυσματικού χώρου (vector space model).

Ο διαχειριστής OntoFM, ως ένα πλήρες εργαλείο διαχείρισης προσωπικής πληροφορίας, επιτρέπει τη διαχείριση της προσωπικής οντολογίας μέσω της διεπαφής χρήστη που προσφέρει. Οι βασικές λειτουργίες επεξεργασίας της οντολογίας αφορούν στη διασύνδεση αρχείων ή καταλόγων με υπάρχουσες έννοιες ή στιγμιότυπα της οντολογίας. Η συσχέτιση μπορεί να υλοποιηθεί με δύο τρόπους: είτε (i) *οπτικά*, με σύρσιμο και εναπόθεση ενός αρχείου από τη λίστα αποτελεσμάτων πάνω σε ένα κόμβο της οπτικοποίησης της οντολογίας, ή (ii) με τη *χρήση μενού* που εμφανίζεται στο δεξί click πάνω σε επιλεγμένο αρχείο από τη λίστα αποτελεσμάτων. Στην πρώτη περίπτωση του συρσίματος και της εναπόθεσης αρχείου πάνω σε κόμβο, εμφανίζεται ένα αναδυόμενο παράθυρο που περιέχει όλες τις πιθανές συσχετίσεις μεταξύ του αρχείου και του στιγμιότυπου που αναπαριστά ο κόμβος, δίνοντας έτσι τη δυνατότητα στο χρήστη να επιλέξει το είδος της συσχέτισης που θέλει να προσθέσει. Αν ο κόμβος αναπαριστά έννοια αντί για στιγμιότυπο, εμφανίζεται στο χρήστη ένα ενημερωτικό μήνυμα που αναφέρει ότι η προσθήκη της νέας συσχέτισης ήταν επιτυχής. Στην περίπτωση της χρήσης μενού, εμφανίζεται ένα διαφορετικό αναδυόμενο παράθυρο που δίνει τη δυνατότητα στο χρήστη να επιλέξει το είδος της συσχέτισης που θέλει να προσθέσει, καθώς και το αντικείμενο της συσχέτισης αυτής από μία λίστα που περιέχει

τα πιθανά αντικείμενα και ενημερώνεται αυτόματα ανάλογα με τη συσχέτιση που έχει επιλεγεί. Η συσχέτιση αρχείου με έννοια ή στιγμιότυπο με χρήση μενού, επιτρέπει στο χρήστη και τη διαγραφή μιας υπάρχουσας συσχέτισης (που δεν είναι σχέση συστήματος) με τη βοήθεια ενός πίνακα που περιέχει όλες τις υπάρχουσες συσχετίσεις για το συγκεκριμένο αρχείο που επιλέχτηκε.

Στόχος του OntoFM είναι να γίνει μία επέκταση του προσωπικού υπολογιστή διευκολύνοντας τη διαχείριση προσωπικής πληροφορίας με την παροχή μηχανισμών περιήγησης και αναζήτησης εμπλουτισμένους με σημασιολογικό περιεχόμενο. Παρόλο που ο διαχειριστής αρχείων OntoFM είναι λειτουργικός, υπάρχει η προοπτική της βελτιστοποίησης αλλά και της επέκτασης της λειτουργικότητάς του καθώς και η διεξαγωγή μελετών αξιολόγησης σχετικά με τα στοιχεία διεπαφής χρήστη και τις λειτουργίες που προσφέρει.

Λέξεις - Κλειδιά: Οντολογία, Χάρτες σκέψης, Protégé, Plug-in, Προσωπική διαχείριση πληροφοριών

TABLE OF CONTENTS

I. ACKNOWLEDGEMENTS.....	III
II. SUMMARY	IV
III. ΠΕΡΙΛΗΨΗ	V
IV. TABLE OF CONTENTS.....	X
V. TABLE OF FIGURES	XII
VI. TABLE OF TABLES.....	XIII
1 INTRODUCTION	1
1.1 Objectives and Motivation.....	2
1.2 Thesis Structure.....	3
2 THEORETICAL FRAMEWORK	5
2.1 Personal Information Management	6
2.2 Graphic Representations of Ideas and Information.....	9
2.2.1 Concept Maps.....	9
2.2.2 Mind Maps.....	11
2.2.3 Concept Maps Versus Mind Maps.....	12
2.3 Software Platforms and Languages.....	14
2.3.1 The Protégé Ontology Editor and Knowledge Acquisition System.....	14
2.3.2 Lucene Information Retrieval Library.....	16
2.3.3 Web Ontology Language (OWL).....	18
3 RELATED WORK	19
3.1 Personal Information Management Related Systems.....	20
3.1.1 MIT Semantic File System (MIT-SFS).....	20
3.1.2 Semantic, Deep Archival File System (SEDAR)	21
3.1.3 GoNTogle.....	22
3.1.4 NEPOMUK.....	25
3.1.5 Semex.....	27

3.1.6	Haystack	29
3.1.7	SPONGE	29
3.2	Related Ontology Visualizations.....	30
3.2.1	NavigOWL	31
3.2.2	OWLViz.....	31
3.2.3	OntoGraf	32
3.3	A brief comparison.....	33
3.3.1	Comparing Related Personal Information Systems to OntoFM.....	34
3.3.2	Comparing Related Visualizations to OntoFM Visualization.....	35
3.3.3	Comparing Results.....	36
4	PREVIOUS EFFORTS	38
4.1	Earlier implementation of OntoFM	39
4.2	Problems and Solutions.....	40
5	DEVELOPMENT PROCESS	42
5.1	Assumptions.....	43
5.1.1	General Assumptions.....	43
5.1.2	Visualization Assumptions.....	44
5.2	Specifications	46
5.3	Architecture.....	47
5.4	Implementation	49
5.4.1	Platform and tools.....	49
5.4.2	Libraries.....	49
6	ONTOFM PLUG-IN PRESENTATION	51
6.1	Overview	52
6.2	Functionality	53
6.2.1	Browsing through Catalogues and Ontology Items	53
6.2.2	Search and Locate Files	56
6.2.3	Advanced Search.....	59
6.2.4	Relate File to Items	61
7	FUTURE WORK	66
VII.	REFERENCES.....	XIV

TABLE OF FIGURES

Figure 2-1. Activities inside and outside PIM by (Boardman, 2004).....	7
Figure 2-2. PIM activities viewed as an effort to establish, use, and maintain a mapping between needs and information by (Jones, 2008).....	8
Figure 2-3. Activities inside and outside PIM by (Kljun, Dix, & Solina, November 2009).....	9
Figure 2-4. A concept map example concerning concept maps.....	10
Figure 2-5. A mind map example concerning mind maps.....	11
Figure 2-6. The Protégé architecture as described in (Gennari, et al., 2003).....	15
Figure 3-1. GoNTogle automatic annotation: System suggests the class H.2_DATABASE_MANAGEMENT as the most appropriate class for the current document.....	24
Figure 3-2. GoNTogle hybrid search: Find documents relevant to XML annotated with the class H.2_DATABASE_MANAGEMENT.....	25
Figure 3-3. NEPOMUK architecture.....	26
Figure 3-4. A sample screenshot from browsing the Semex database.....	28
Figure 3-5. Image stump from SPONGE's Search gadget.....	30
Figure 3-6. Image stump from the NavigOwl interface.....	31
Figure 3-7. Part of (Pizza ontology) with the OWLViz visualization.....	32
Figure 3-8. Part of the ontology described in (Katifori, et al., 2008) and (Golemati, Katifori, Vassilakis, Lepouras, & Halatsis, 2007) drawn by the OntoGraf plug-in.....	33
Figure 4-1. OntoFM screen with captions - Inputting multiple keywords. Entered terms appear in the related concepts pane along with directly connected concept instances.	39
Figure 5-1. OntoFM architecture scheme.....	48
Figure 6-1. OntoFM initial screen with captions.....	52
Figure 6-2. Browsing the file system hierarchy. Contents of the selected folder "University" at the center and corresponding ontology visualization on the right.....	54
Figure 6-3. Browsing the personal ontology - Visualizing all relations to "Ontology Visualization Methods - A Survey" after browsing to the "University" folder.....	55
Figure 6-4. Ontology visualization options - Show specific node types.....	56
Figure 6-5. Ontology visualization options - Show specific arc types.....	56
Figure 6-6. Autosuggestion feature.....	57
Figure 6-7. Search for "ontology" relevant items in multiple selected folders.....	58
Figure 6-8. Search for "ontology iswc" relevant items in multiple selected folders.....	59
Figure 6-9. Multiple folders selection - "University" and "2011 Files" have been both selected to view their contents.....	60
Figure 6-10. Relate by right clicking file to concept/instance dialog with captions.....	62
Figure 6-11. Candidate relations for "OntoFM A Personal Ontology-based File Manager for the Desktop" file.....	63
Figure 6-12. Creating new relation - "OntoFM A Personal Ontology-based File Manager for the Desktop" file "acceptedIn" "ISWC 2011" "Conference/Journal".....	63
Figure 6-13. Relate by drag and drop file to concept/instance dialog.....	65

TABLE OF TABLES

Table 2-1. Summary of the differences between concept and mind mapping.....	13
Table 3-1. File formats scanned automatically by Semex.....	29
Table 3-2. Summary table concerning the differences among MIT SFS, GoNTogle, NEPOMUK and, OntoFM (✓=supported, ✗= not supported, ?=not mentioned)	35
Table 3-3. Summary table concerning the differences among NavigOWL, OWLViz, OntoGraf and, OntoFM ontology visualizations (✓=supported, ✗= not supported)	36

1

INTRODUCTION

PREVIEW

- Motivation
- Objectives
- Thesis overview

This chapter gives a short introduction to this thesis. It describes the motivation, the background, and the goal of this thesis. A thesis overview is included.

1.1 OBJECTIVES AND MOTIVATION

If we live to be 70 years old, we get roughly 613,200 hours to live. That's 365 days, times 70 years, times 24 hours. This is the time we have to enjoy, to love, to weep, to learn and to cry. The quest for 'the good life' for the meaning, for fulfillment, and purpose must fit into this average number of hours. But we spend a lot of our time doing other things, things we have little choice in, and so we do not pay much attention to. One of these things is retrieving information we already found but cannot remember their location. Assuming we spend 5 minutes each day on finding information we need, we waste approximately 2,129 hours of our living time trying to retrieve the needed information. This number gets even bigger as the amount of time we spend on searching increases. A common everyday example that reflects these statistics is trying to locate a paper in our personal information space by knowing only its author or the name of a conference in which it was submitted or accepted. The question which arises is "How should I organize my information so that I can find information items needed for later use and repeated re-use easily?". Keeping found information found is an essential challenge of personal information management (PIM).

According to (Jones, 2008) personal information management is not only about finding, keeping, organizing, and maintaining information but also managing privacy and the flow of information. A more formal definition refers to personal information management as both the practice and the study of the activities a person performs in order to acquire or create, store, organize, maintain, retrieve, use, and distribute the information needed to meet life's many roles and responsibilities. Personal information management places special emphasis on the organization and maintenance of personal information collections in which information items, such as paper documents, electronic documents, email messages, web references, etc. are stored for later use and repeated re-use.

However, (Jones, Bruce, & Dumais, 2001) have reported that personal information management is frequently a chore, as it is poorly supported by existing technology and many users effort to handle, classify and retrieve the accumulating information. A wide concern is that everyday needs provoke personal information

management problems that have repercussions to work productivity. To name some of them, computer users are being exposed to more and more information. Increased storage capacity allows them to collect this information, leading to more management overheads. Not to forget that, many users are managing information in more than one places (multiple desktop computers, laptops, tablets, and mobile phones).

(Katifori, Poggi, Scannapieco, Catarci, & Ioannidis, 2005) have proposed personal ontologies as a means to support personal information management with the semantic management of a user's information. Assuming a personal ontology system is in use, new tools have to be developed to exploit the enhanced capabilities of the user interface. To this end, a number of semantics-based methods have emerged employing personal ontologies as an underlying infrastructure for organizing and querying the personal information space.

This thesis deals with all these mentioned facts, aspects and questions. It attempts to explore, analyze and describe the area of personal information management and existing facilities and tools. The centerpiece of this effort is the OntoFM plug-in for (Protégé), developed in the context of this thesis with regard to requirements and needs of common users. OntoFM aims to become an extension of the personal computer, allowing the management of personal files based on semantic content. The kernel of the OntoFM plug-in is an existing personal ontology which corresponds to the user's personal information space. This ontology is exploited to complement the traditional hierarchy-based management of personal files with the potential to navigate and search within the personal information space through the concepts and relationships within the ontology.

1.2 THESIS STRUCTURE

The following chapter of this thesis, Chapter 2, consists of a literature review. A number of academic papers have been reviewed to give an introductory overview of personal information management, concept and mind mapping techniques, Protégé functionality and Lucene indexing and searching capabilities. Chapter 3 is about overview of the systems and visualizations related to our work and comparing OntoFM with these related approaches. Continuing with Chapter 4, which is a presentation of

previous efforts concerning the OntoFM plug-in, followed by a list of problems occurred and solutions proposed to overwhelm them. Chapter 5 analyzes the development process of the OntoFM plug-in by introducing assumptions and specifications and giving a prototype architecture and description about its components. In chapter 6, an overview outline and a functionality list are described using image stumps to illustrate the plug-in's functionality and to better explain the features provided. Finally, chapter 7 gives the summary of the thesis and reports suggestions for future work.

2

THEORETICAL FRAMEWORK

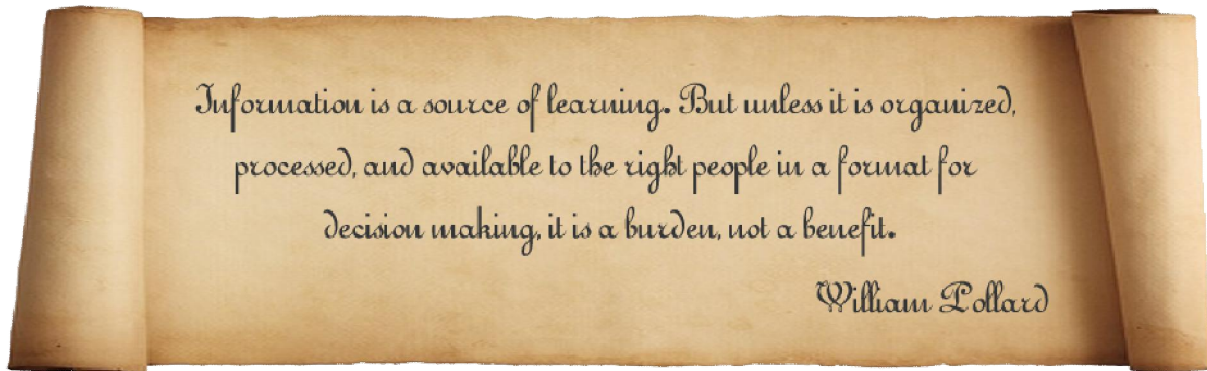
PREVIEW

- Personal Information Management (PIM)
- Graphic Representations of Ideas and Information
 - Concept Maps, Mind maps
- Software Platforms and Languages
 - Protégé, Lucene
 - OWL

The following chapter presents a more formal introduction of the theoretical and technological basis of this thesis. Included is a brief review of personal information management, graphic representations of ideas and information, and software platforms and languages concerning this thesis.

In this chapter, a brief overview about personal information management and graphic representations of ideas and information, such as concept maps and mind maps is given. Later, Protégé software platform, Lucene information retrieval library and OWL language are discussed.

2.1 PERSONAL INFORMATION MANAGEMENT



In a perfect, ideal world, we would always have the right information in the right place, in the right form, and of sufficient completeness and quality to meet our current need. However, in the real world, we do not always find the right information in the right place, or the information arrives too late to be useful, or even worse, information is never found.

Personal Information Management (PIM) intends to make the world, or even better in this case, our workspace a better place for finding, keeping, organizing and maintaining information. PIM is about the use of information to manage precious resources such as our time, make good decisions and carry out successfully everyday tasks. From time to time, various researchers have tried to dismember PIM activities in order to understand how PIM is performed.

(Barreau, 1995) views a PIM system as a file hierarchy centered system. He divided PIM in 5 sub-activities as *acquisition*, *organization and storage*, *maintenance*, *retrieval* and *output*. To start with acquisition, it is about defining, labeling and grouping the information we decided to be included in our information space. After acquisition the next sub-activity is organization and storage which includes classifying, naming, grouping and placing information in the right place for later retrieval. These two sub-

activities are determinant; however they would have no value if there was not the third sub-activity, maintenance. Maintaining information systems is an important function as the out-of-date information is updated by moving or deleting information from the information space so that it corresponds to each current state. Moving to the next sub-activity, retrieval takes over finding information for re-use and finally, output is about visualizing the information space based on users' needs and objectives.

(Boardman, 2004) based his classification on (Barreau, 1995) describing PIM as a four sub-activity procedure. As shown in Figure 2-1, he kept the first four Barreau's sub-activities and rejected the output activity as he believes that visualizing is done by computers. His division focuses on a computer as a set of several PIM systems that allow a user to manage a collection of personal information.

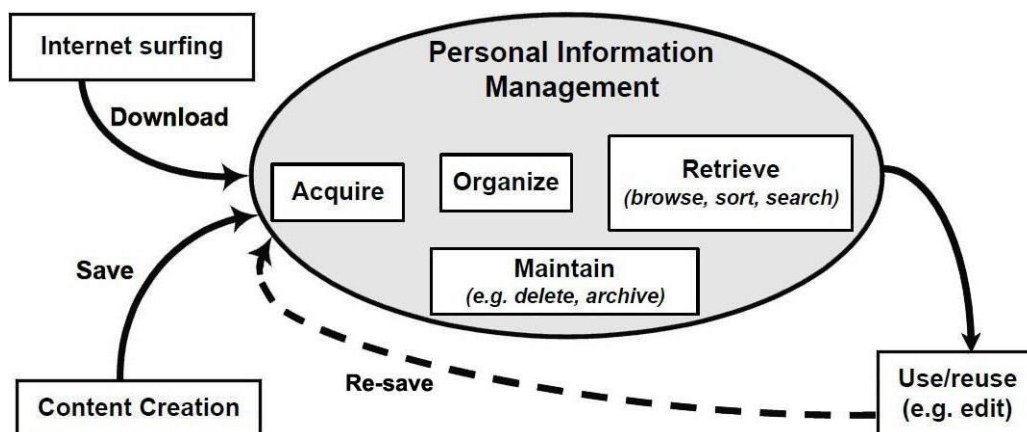


Figure 2-1. Activities inside and outside PIM by (Boardman, 2004)

(Jones & Teevan, 2007) had a different, more general than Boardman's and Barreau's prospective of PIM, which includes the whole users' personal information space. As shown in Figure 2-2, they divided all PIM activities in three groups as *keeping*, *finding and re-finding* and, *meta-level activities*. Keeping activities are about deciding about the future needs and future availability for a single information; finding and re-finding activities are conducted by the user's needs for information; and last but not least, meta-level activities include maintaining and organizing the personal information collections within the personal information space, managing privacy, evaluating personal space of information, making sense of information and information distribution.

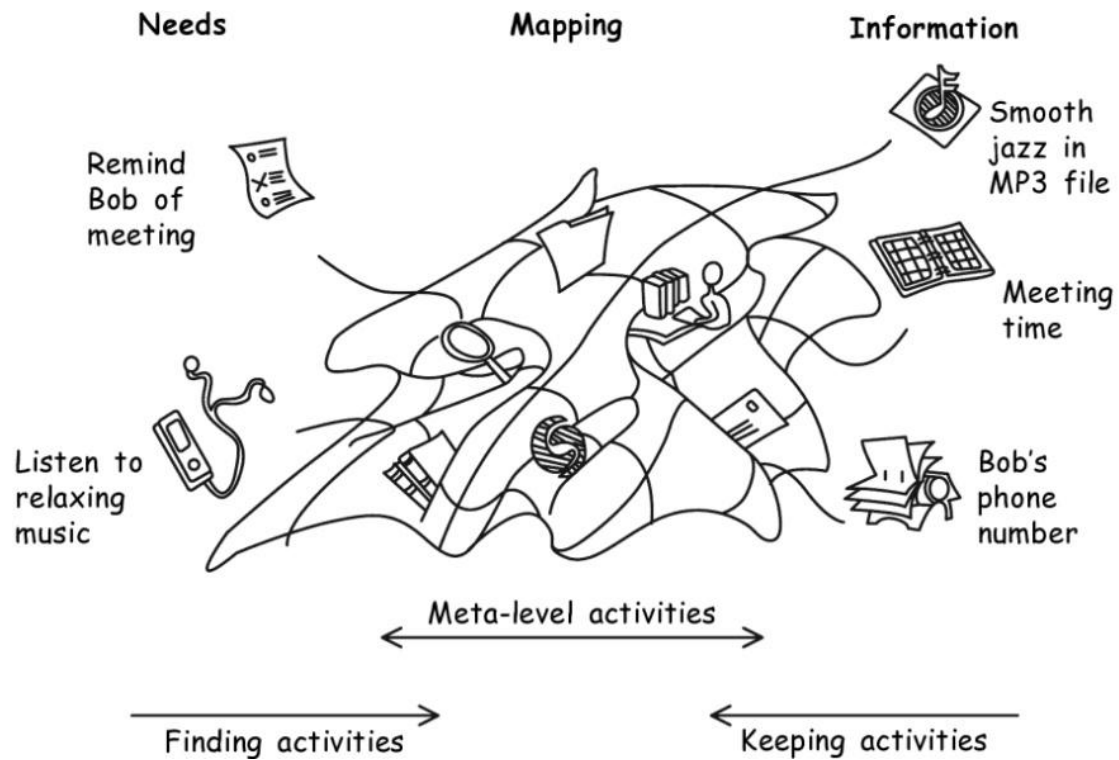


Figure 2-2. PIM activities viewed as an effort to establish, use, and maintain a mapping between needs and information by (Jones, 2008)

(Kljun, Dix, & Solina, November 2009) PIM framework is based on Boardman. It includes four PIM sub-activities (acquiring, organizing, maintaining and retrieving) but in authors' prospective these activities *overlap*. Overlapping is due to the fact that PIM activities rarely take place one after another. As shown in Figure 2-3, information organization overlaps with all other activities. Performing PIM activities can result in changes of the organizational structure of user's personal information space as such activities include some organization as well.

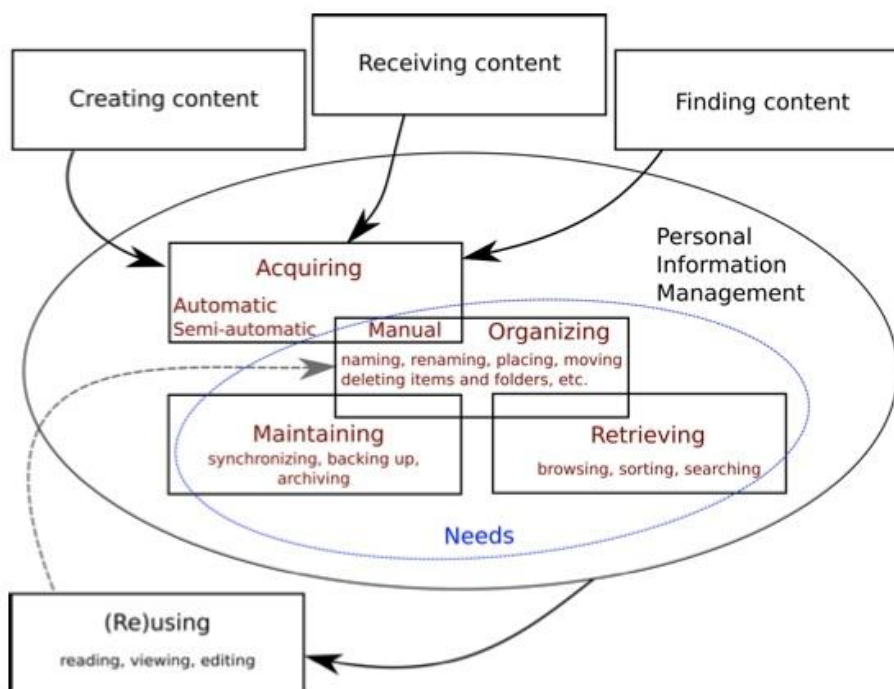


Figure 2-3. Activities inside and outside PIM by (KlJun, Dix, & Solina, November 2009)

These four approaches are not the only ones. Each person has a different estimation on how PIM activities should be divided. However, each one of them believes that PIM has great importance in managing precious resources, making good decisions and carrying out successfully everyday tasks.

2.2 GRAPHIC REPRESENTATIONS OF IDEAS AND INFORMATION

The value of graphics can hardly be underestimated. Graphs, charts and diagrams are like pictures: they can “speak a thousand words”. They are useful for expressing information clearly and simply, and they can be used as a visual-thinking tool. This subsection lists some graphic representations of ideas and information such as concept and mind maps.

2.2.1 CONCEPT MAPS

Concept maps are graphical tools for exploring knowledge and gathering and sharing information. They can be used to describe ideas about some topic in a pictorial

form. In a concept map, each word or phrase is connected to another and linked back to the original idea, word or phrase. Concept maps are a way to develop logical thinking and study skills by revealing connections and helping users see how individual ideas form a larger whole.

A concept map consists of nodes or cells that contain a concept, item or question and relationships between these concepts indicated by a connecting line with an arrow symbol denoting the linking direction of the two concepts. The arrow describes the direction of the relationship and reads like a sentence. Each linking line has a word or phrase, referred to as linking words or linking phrases, which specify the relationship between the two concepts. Figure 2-4 shows an example of a concept map that describes the structure of concept maps and illustrates the above characteristics.

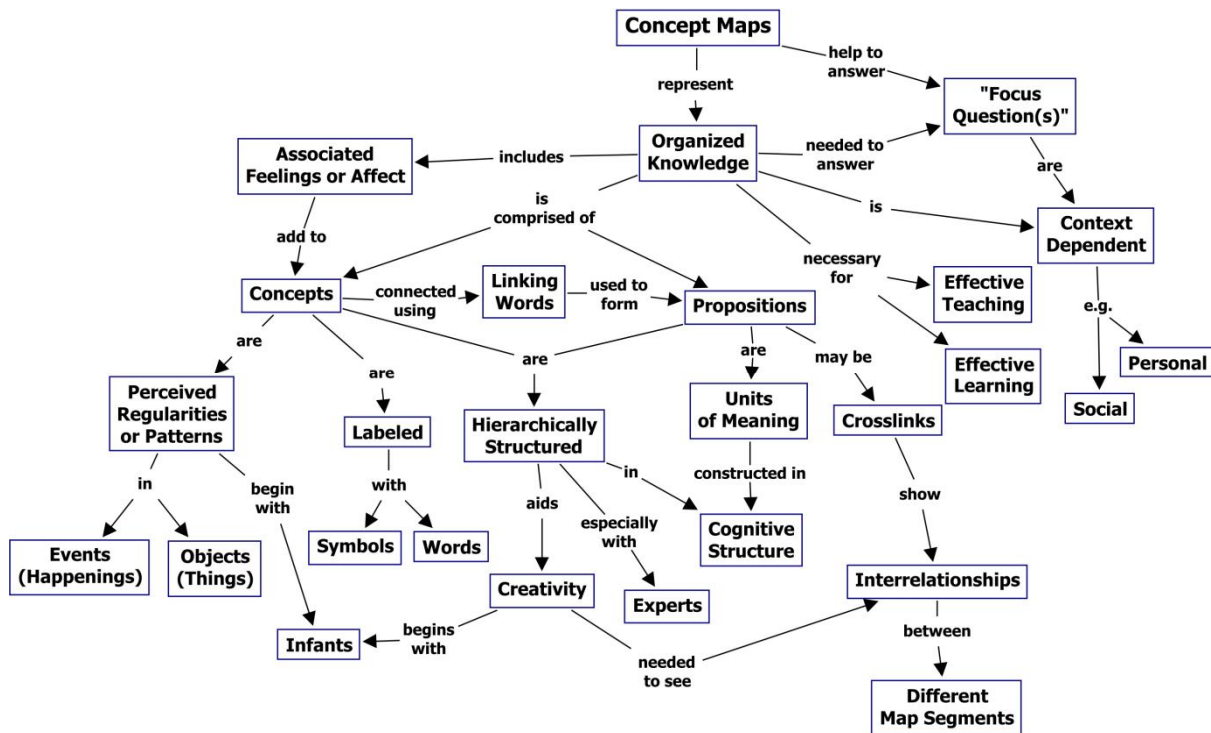


Figure 2-4. A concept map example concerning concept maps

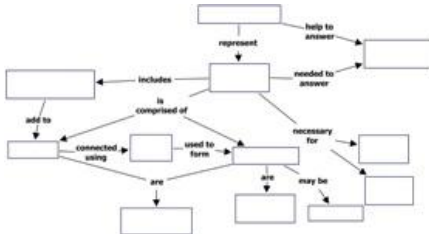
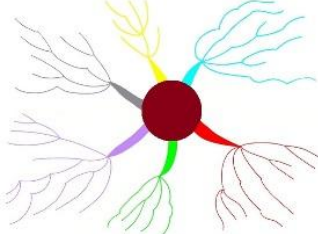
Concept maps have a hierarchical format with the most general concepts at the top and the more specific, less general concepts arranged hierarchically below. Apart from concepts, concept maps could also contain specific examples of events or objects that help to clarify the meaning of a given concept. Normally these are not included in ovals or boxes, since they are specific events or objects and do not represent concepts.

of the concepts, and are classified into groupings, branches, or areas, aiming to represent semantic or other connections between portions of information. Even though the tree format is a hierarchical structure, the radial arrangement disrupts the prioritizing of concepts. (Buzan, 1991) claimed that by presenting these connections in a radial, graphical, non-linear manner, encourages brainstorming.

Mind mapping software can be used to organize large amounts of information, combining spatial organization, dynamic hierarchical structuring and node folding. The mind mapping concept can be extended by allowing individuals to map more than thoughts and ideas with information on computers and the internet, like documents, images and e-mails.

2.2.3 CONCEPT MAPS VERSUS MIND MAPS

Concept maps can be contrasted to mind maps. Although the two visualization techniques are both very useful for visualizing and understanding ideas and information, they are not exactly the same. Table 2-1 summarizes the differences between the two forms of mapping as mentioned in (Davies, 2011) and (Eppler, 2006).

	Concept Maps	Mind Maps
Thumbnail Representation		
Purpose	Relations between concepts	Associations between ideas, topics or things
Structure	Hierarchical Tree-like	Non-linear Organic Radial
Level of Abstraction	Medium generality	High generality

Nodes	Boxes or Bubbles with text	Pictures Words Diagrams
Linking devices	Arrow	Lines Line Thicknesses Colors Shading
Linking words	Relational phrases (i.e. "related to", "is author of")	Associative words (i.e. "use", "links")
Language register and "granularity"	Medium	Loose
Reading direction	Top-down	Center-out
Extensibility	Limited	Open
Memorability	Low	Medium to High
Understandability	High	Low

Table 2-1. Summary of the differences between concept and mind mapping

Starting from the basic difference, concept maps help visualizing the relationships between different concepts within a central focus, while mind maps associate words and ideas with a central governing key word or concept. Concept maps can take a variety of forms ranging from hierarchical, to non-hierarchical, allowing even recursion and permit spatial node and arc arrangement. On the other hand, mind maps are predominantly bifurcating and clustering is implemented by color or line thickness. As far as precision and formality are concerned, the above statements lead to the conclusion that concept maps are more structured and formal contrariwise to mind maps which are less formal and structured. Finally, concept maps lead to a deeper understanding, but they are not very easy to remember. However, mind maps are not as understandable as concept maps but they are easier to remember.

2.3 SOFTWARE PLATFORMS AND LANGUAGES

This subsection is a brief review of software platforms, languages and development tools concerning this thesis. Protégé software platform, Lucene information retrieval library and OWL language are discussed.

2.3.1 THE PROTÉGÉ ONTOLOGY EDITOR AND KNOWLEDGE ACQUISITION SYSTEM

Protégé is a free, open-source ontology editor/creator and knowledge-base framework, developed by the SMI (Stanford Medical Informatics) group at Stanford University. It is based on Java, is extensible, and provides a plug-and-play environment that makes it a flexible base for rapid prototyping and application development. The Protégé platform supports two main ways of modeling ontologies via the Protégé-Frames and Protégé-OWL editors. Ontologies can be exported into a variety of formats including RDF(S), OWL and, XML Schema.

Due to the fact that Protégé is the most widely-used ontology creation tool of the market with lots of support and a wealth of plug-ins, it was proposed for this thesis as a means to interact with the personal ontology. For this reason, the following subsections focus in Protégé architecture and knowledge model.

2.3.1.1 Architecture

(Gennari, et al., 2003) reviewed the development of Protégé and evaluated its adopted architecture, which is shown in Figure 2-6. On the top of the architecture, developers are allowed to add features they want or need by customizing the user interface or build a new one to interact directly with the knowledge model. This can be implemented by extending Protégé with plug-ins.

The plug-in architecture provides a flexible and powerful mechanism for extending Protégé in many ways. Plug-ins, usually *.jar files located below the «plugins» subdirectory, are loaded at Protégé startup. The system creates an instance of each plug-in, and arranges for the invocation of the appropriate plug-in methods as needed.

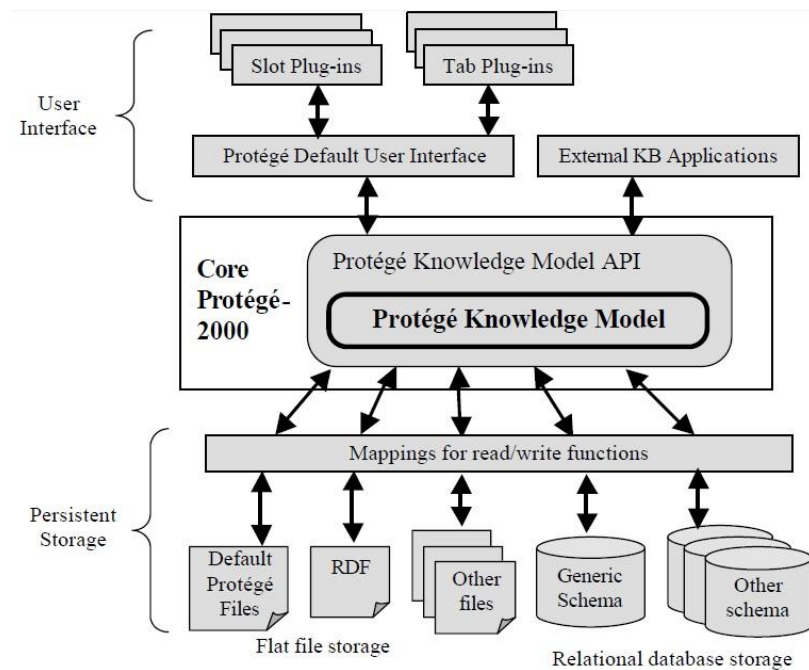


Figure 2-6. The Protégé architecture as described in (Gennari, et al., 2003)

There are several types of plug-ins, such as:

- *TabWidget plug-in*, used for functionality/application extensions to appear on a new tab in the Protégé user interface
- *SlotWidget plug-in*, for user interface extensions to the group of components that support slot value entry
- *KnowledgeBaseFactory plug-in*, etc.

2.3.1.2 Knowledge Model

Protégé's knowledge model is frame-based and OKBC compatible. (Noy, Ferguson, & Musen, 2000) describe four basic modeling components including *classes and instances*, *slots*, *facets*, and *axioms*. Starting with classes, which stand for the domain concepts and can be concrete or abstract, they can be organized in class hierarchies where multiple inheritances are permitted. Instances stand for a case of a concept, and each one inherits all the attributes and relations of the concept. Slots, which represent attributes or properties of the domain concepts, may have the same name but they stand for different attributes in different classes. Facets represent constraints of slots as they define restrictions and finally, axioms specify additional constraints.

2.3.2 LUCENE INFORMATION RETRIEVAL LIBRARY

Apache (Lucene) is a free and open source information retrieval software library with full text indexing and searching capabilities. While suitable for any application which requires full text indexing and searching capability, Lucene has been widely recognized for its utility in the implementation of internet search engines and local, single-site searching. At the core of Lucene's logical architecture is the idea of a document containing fields of text. This flexibility allows Lucene's API to be independent of the file format. Text from PDF, HTML, Microsoft Word, and OpenDocument documents, as well as many other (except images), can all be indexed as long as their textual information can be extracted.

2.3.2.1 Indexing

Lucene stores all of the necessary information in indexes in order to improve search efficiency. Each index contains sequences of documents, which themselves consist of sequences of fields. Inverted indexing technique is used for storing information as indexes list which documents contain each term. Fields may also be tokenized and placed into the index as individual tokens or placed into the index as a single literal term.

2.3.2.2 Searching

2.3.2.2.1 Querying

Lucene searches are implemented by user queries processed by a lexical analyzer, named *Lucene Query Parser*. Queries are made up of two primary components, *terms* and *operators*. Terms may consist of both individual terms as well as phrases which combine a group of terms enclosed by double quotation marks. In addition to basic terms, fielded data may be stored in a Lucene index. This feature is particularly useful as it allows users to search specific fields, such as a title or author. Among the advanced search features are not only field searches but also wildcard searches, ranges searches and boolean operators, such as "AND", "OR", "NOT", "+" and "-".

Starting with *wildcard searches*, the user can replace one or more symbols in a query term by changing them with the special wildcard characters "?" or "*". In case she

replaces with "?", the symbol may be substituted for a single character, while in case she replaces with "*", the symbol may be substituted for multiple characters. Fuzzy searches are conducted by applying a "~" symbol to the end of a search term. By doing so, terms similar in spelling to the original term will be found. Adding a value between 0 and 1 allows for the user to define the threshold level of similarity for terms returned by a fuzzy search.

Range searches, conducted by enclosing two field values in square or curly brackets, may be used to locate documents whose ranges lie within a certain upper and lower bound. Search terms may be further modified by a variety of means including wildcard and fuzzy searches.

Finally, *boolean operators* may be used to retrieve different search results using the same query terms depending on the operator used. Each of them has a different meaning. Starting with the "AND" operator, it may be inserted between terms or phrases to indicate that both should be located. Similarly, the "OR" operator indicates that either should be located. Inserting the "NOT" operator between two terms or phrases excludes the second term or phrase from the search results but includes the first. If the "+" operator precedes a term or phrase then it must be included in the results. However, if the "-" precedes a term or phrase then it must be excluded from the results. Finally, parentheses may be used to group terms, phrases, or field values.

2.3.2.2.2 Scoring

Lucene scoring uses a combination of the Vector Space Model (VSM) of Information Retrieval and the Boolean model to determine how relevant a given Document is to a user's query. In general, the idea behind the Vector Space Model is the more times a query term appears in a document relative to the number of times the term appears in all the documents in the collection, the more relevant that document is to the query. Lucene uses the Boolean model to first narrow down the documents that need to be scored based on the use of boolean logic in the Query specification. Finally, even if it adds some capabilities and refinements onto this model to support boolean and fuzzy searching, it essentially remains a Vector Space Model based system.

2.3.3 WEB ONTOLOGY LANGUAGE (OWL)

The Web Ontology Language (OWL, 2005) is a family of knowledge representation languages for authoring ontologies. Like Protégé, OWL is used to describe concepts. However, it also provides new facilities as it has a richer set of operators (intersection, union and negation). It is based on a different logical model which among other features provides a reasoner offering a checking whether or not all of the statements and definitions in the ontology are mutually consistent. Moreover, the reasoner helps recognizing which concepts fit under which definitions helping in this way to maintain the hierarchy correctly when dealing with cases where classes can have more than one parent.

2.3.3.1 OWL Components

As mentioned above, OWL ontologies are similar to Protégé frame based ontologies leading to the fact that they have similar components. However, the terminology used to describe these components is slightly different as an OWL ontology consists of *individuals*, *classes*, and *properties*.

Starting from individuals, which are also known as *instances*, they represent objects in the domain. Classes, which are also known as *concepts*, are groups of individuals which can be organized into a superclass-subclass hierarchy, also known as a *taxonomy*. Finally, properties represent relationships and can be either characterized as *object* or *datatype* properties. While object properties are relationships between two individuals, datatype properties describe relationships between an individual and data values.

To make OWL and Protégé component correspondence more clear, OWL ontology consists of individuals, properties, and classes, which roughly correspond to Protégé frames instances, slots and classes.

3

RELATED WORK

PREVIEW

- Personal Information Management systems
 - MIT SFS, GoNTogle, SEDAR, NEPOMUK, SEMEX, HAYSTACK, SPONGE
- Relevant ontology visualizations
 - NavigOwl, OWLViz, OntoGraf

This chapter is an overview of the systems related to our work. These systems were important while designing our personal information management system. Moreover, some ontology visualizations implemented as Protégé plug-ins compatible with Protégé-OWL 4.0 are presented. Finally, a brief comparison with these related approaches is included.

The following subsections include descriptions of systems and visualizations related to our work. Starting with 3.1 subsection, we present several personal information management systems such as MIT SFS, GoNTogle, SEDAR, NEPOMUK, SEMEX, HAYSTACK, and SPONGE while 3.2 subsection presents the three most well-known ontology visualizations implemented as Protégé plug-ins compatible with Protégé-OWL 4.0. Finally, 3.3 subsection summarizes and compares these related systems and visualizations to OntoFM's functionality and look.

3.1 PERSONAL INFORMATION MANAGEMENT RELATED SYSTEMS

Some related PIM systems will be described in this section. Instead of giving a full description of every system, we have described the systems exploring similar concepts, as the one implemented for this thesis.

3.1.1 MIT SEMANTIC FILE SYSTEM (MIT-SFS)

(Gifford, Jouvelot, Sheldon, & O'Toole, 1991) have developed a semantic file system which provides access to both file contents and metadata by extracting attributes using transducers through the file system.

Files stored are interpreted by transducers, which act like plug-ins and are file type specific, in order to produce a set of descriptive attributes. To make that more clear, a *transducer* is a filter that takes as input the file's contents and outputs the file's entities and their corresponding attributes. A transducer could be either (i) *simple*, by treating an input file as a single entity and use the file's unique words as attributes or (ii) *complex*, by performing type reconstruction on an input file, identifying each procedure as an independent entity and using attributes to record their reconstructed types. An *attribute* is a pair consisting of a *field*, which is a file property such as "author", "owner", "date", "type", etc., and a *value*, which may be a string or an integer. Each file may have many attributes with some of them having the same field name but different value. Finally, a unit of associative access is called *entity* and may consist of an entire file, an object within a file, or a directory.

Through querying the semantic file system the user can retrieve a set of files and/or directories containing the entities described in the query. *Queries* are boolean combinations of attributes, allowing the user to narrow down the number of retrieved results by inputting desired attributes' values. However, it is also possible to retrieve all values of a given field.

Queries are performed by using virtual directories, which are indistinguishable from ordinary directories. The query facilities of a semantic file system appear as virtual directories at each level of the directory tree. A *field virtual directory* is named by a field, and has one entry for each possible value of its corresponding field. Thus in `/sfs`, the virtual directory `/sfs/owner:` corresponds to the `owner:` field. The field virtual directory `/sfs/owner:` has one entry for each owner that has written a file in `/sfs`. For example:

```
% ls -F /sfs/owner:
jones/ root/ smith/
```

Value virtual directories are entries in a field virtual directory which have one entry for each entity described by a field-value pair. Thus the value virtual directory `/sfs/owner:/smith` contains entries for files in `/sfs` that are owned by Smith. Each entry is a symbolic link to the file. For example:

```
% ls -F /sfs/owner:/smith
bio.txt@ paper.tex@ prop.tex@
```

3.1.2 SEMANTIC, DEEP ARCHIVAL FILE SYSTEM (SEDAR)

(Mahalingam, Tang, & Xu, 2003) developed a similar system to MIT Semantic File System named Sedar. Sedar is a peer-to-peer archival file system offering semantic retrieval by using semantic vectors extracted from the file contents. Each time a file is modified and closed, Sedar creates a new instance of the file followed by a different version number. Even though, metadata is not versioned, a virtual snapshot using timestamps allows accessing the namespace arbitrary back in time. Moreover, a

semantic-based interface is provided allowing clients to locate files according to the semantics available.

Sedar consists of five main components:

- (i) a *NFS loop-back server*,
- (ii) a *Catalogue*, containing an index of the files based on vectors of file type specific features extracted from file contents, named *semantic vectors*,
- (iii) an *Extractor registry*, which is an external plug-in either deriving the semantic vectors in case the data types are known or deriving features using statistical analysis in case of unknown types,
- (iv) a *Sedar distributed storage (SDS)*, providing basic support for storing and retrieving files, directories and the catalogue, and finally
- (v) a *Sedar semantic utility*, offering semantic-based retrieval capabilities as it interacts with the file system to generate materialized views of query results in order users to access them as regular file system objects.

Sedar has a similar directory structure to a UNIX based system. Each directory entry contains the name and type of the object and a unique identifier called "Inode" that references to the object. Even though a directory Inode does not contain version information, a file Inode contains all file's versions. To condense storage utilization based on the semantic vector of a document, Sedar employs a novel technique called *semantic hashing*. Rather than storing the semantic vector for each file version, it uses a representative semantic vector for several files having close semantic vectors.

Sedar supports operations like the "create" operation, where the user may create a new file or directory or the "lookup" functionality which returns the Inode of the object and its latest version number. Finally, while the "read" operation is used for viewing a file, the "write" operation is implemented as soon as the file is closed in order to update the metadata and the semantic vector.

3.1.3 GoNTogle

(Giannopoulos, Bikakis, Dalamagas, & Sellis, 2010) developed a desktop search engine named GoNTogle, which provides advanced document annotation and search

facilities. GoNTogle allows the user to annotate several types of documents (*.doc, *.pdf, *.rtf, *.txt, *.odt) either manually or automatically by annotating suggestions based on textual similarity and previous document annotations. Moreover, it combines keyword and semantic-based searching, offering in this way advanced ontology query facilities to the users.

GoNTogle is built on top of Lucene and Protégé, using indexing, searching and visualization libraries. It consists of four basic components:

- (i) a *Semantic Annotation Component*, which consists of (a) a Document Viewer, (b) an Ontology Viewer and (c) an Annotation Editor and provides facilities regarding the semantic annotation of documents,
- (ii) an *Ontology Server Component*, that stores the semantic annotations of documents in the form of OWL ontology instances
- (iii) an *Indexing Component*, for indexing the documents using an inverted index, and
- (iv) a *Search Component*, that allows searching for documents using both textual (keyword search) and semantic (ontology search) information.

Through the application the user is able to index documents and annotate semantically either a whole document, or parts of its text, using concepts of a loaded ontology. Annotations are stored on the Ontology Server Component as instances that belong to one or more ontology classes. All the essential annotation information attached to those instances is stored as ontology properties. In case of an automatic annotation, the GoNTogle search engine suggests annotations using a weighted kNN classification that exploits user annotation history and textual information. The training data include document annotations extracted from manual annotations implemented by the user. Figure 3-1 shows an implemented hybrid search.

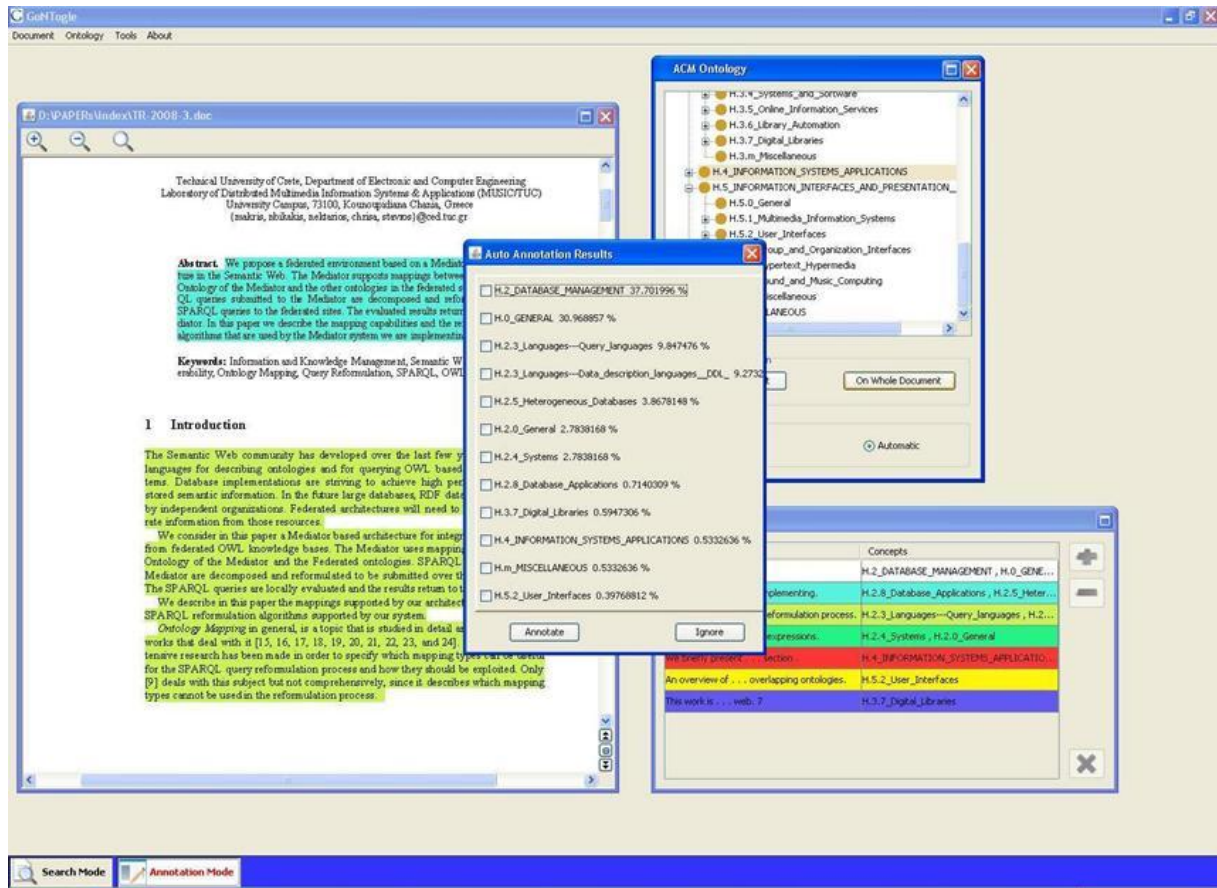


Figure 3-1. GoNTogle automatic annotation: System suggests the class H.2_DATABASE_MANAGEMENT as the most appropriate class for the current document.

GoNTogle's other primary functionality is searching either by inputting keywords and retrieving documents based on textual similarity, or by navigating through classes and retrieving documents semantically annotated to them. The user may also combine these types of searches, implementing a hybrid search by searching for documents using keywords and ontology classes and by determining whether the search will be the intersection or the union of the two searches. Figure 3-2 shows an implemented hybrid search. Moreover, advanced searching facilities may be used after implementing the search faculty such as finding related or similar documents, retrieving the next (subclasses) or *previous* (superclasses) *generations* and searching for documents not only in a selected class but also in its subclasses by implementing *proximity search*.

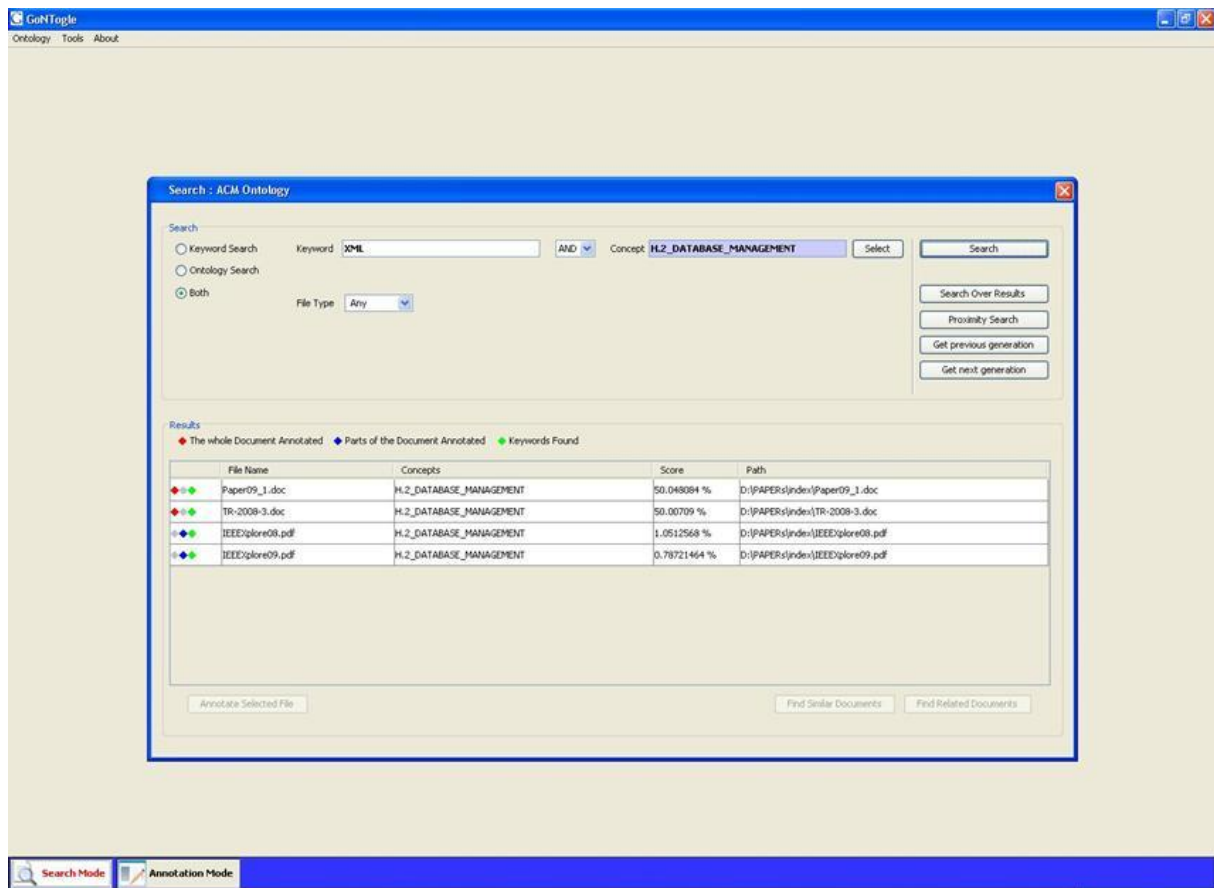


Figure 3-2. GoNTogle hybrid search: Find documents relevant to XML annotated with the class H.2_DATABASE_MANAGEMENT

3.1.4 NEPOMUK

(NEPOMUK) standing for Networked Environment for Personal, Ontology-based Management of Unified Knowledge, is an open-source software specification that is concerned with the development of a social semantic desktop that enriches and interconnects data from different desktop applications using semantic metadata stored as RDF. NEPOMUK aims to extend the personal computer into collaborative environment which supports both the personal information management and the sharing and exchange across social and organizational relations. Through developing methods, data structures and services it aims to organize the file system's information and analyze user's ideas in order to derive semantic information. NEPOMUK's goal is to empower individual knowledge for improving personal information space exploitation and to maintain fruitful communication and exchange within social networks.

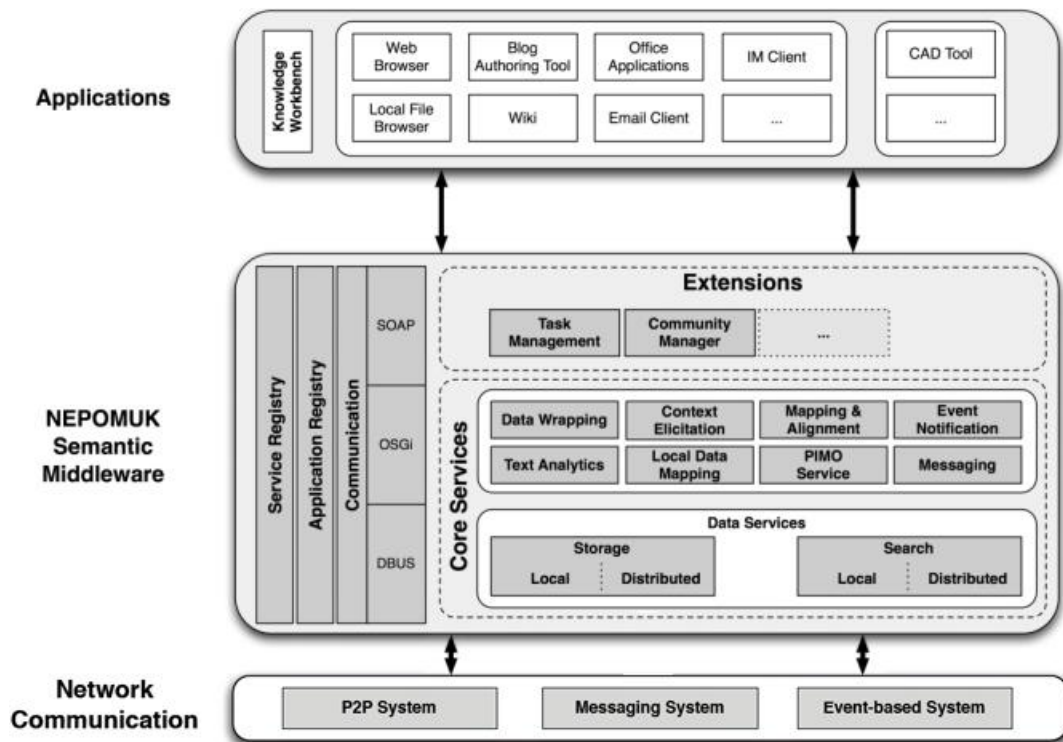


Figure 3-3. NEPOMUK architecture

As shown in Figure 3-3 above, NEPOMUK is organized into three main layers. Due to the fact the NEPOMUK Social Semantic Desktop is a peer-to-peer system consisting of individual desktops, a layer concerning the communication between peers is essential. This layer is named *Network Communication* layer and provides an Event-based System, which is responsible for distributing the events on between the NEPOMUK peers. Each event carries a RDF graph describing the cause of the event. Moreover, the Messaging System routes the messages to receiver and the Peer-to-Peer File Sharing System enables the shared information space. On top of the Network Communication Layer, the *Middleware* layer provides the core services of NEPOMUK. Middleware can be split into three main parts:

- (i) the *Service Registry*, which provides the means of publishing, unregistering and discovery of services,
- (ii) the *Core Services*, which represent the set of standardized services that are part of the NEPOMUK platform (also contains the Data Services which are important functionalities for storing and searching both data and metadata), and

- (iii) the *Extensions*, which are services that may be included in the NEPOMUK Middleware.

Finally, on top of the Middleware layer, the *Presentation* layer provides the NEPOMUK's user interface.

As far as NEPOMUK functionality is concerned, users are able to search amongst different sources or find relevant by querying. They may also access, manage and share resources even if they are offline. Moreover, NEPOMUK has a profiling functionality based on user's activity in order to support user's needs. Data analysis is also supported by providing keyword extraction as sometimes search results might need to be rearranged using sorting or grouping. Finally, at the social level, managing groups and users enhances social interaction and ease resource sharing. In this way, users can publish and subscribe to relevant stream of information.

3.1.5 SEMEX

(Semex) (SEMantic EXplorer), shown in Figure 3-4, offers a platform for the personal information management and provides a logical and integrated view of the personal information. It provides enriched desktop search experiences as it views the user's hard disk as a network consisting of instances and the associations between these instances. The main aim of this system is to enable browsing by creating an automatic association between data items on the desktop.

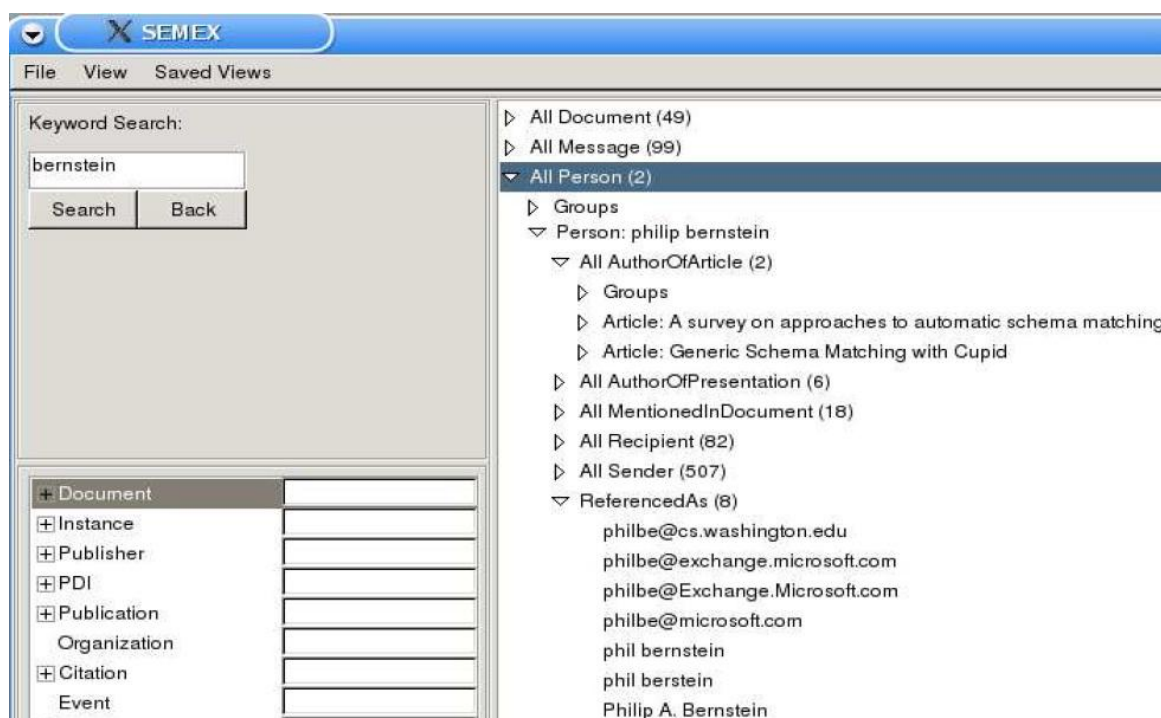


Figure 3-4. A sample screenshot from browsing the Semex database

As described in (Cai, Dong, Halevy, Liu, & Madhavan, 2005) and (Dong, Halevy, Nemes, Sigurdsson, & Domingos, 2004), Semex scans user's hard disk or specified directories and considering the file formats shown in Table 3-1 extracts one of the "Person", "Message", "Article", "Presentation", "Conference", "Journal", "Image", "Documents" or "Webpage" type of instance. Apart from automatic instance-and-association extraction, Semex offers reference reconciliation as it can recognize the person names and email addresses that refer to the same real-world person, and treat them as attributes of a single person instance. Also, when user poses a query it classifies the returned objects into their classes. Then, user can select a particular object instance to see detailed information, including its attribute values and associated instances. Other features concerning Semex are keyword and advanced search, ranking and lineage information.

Type of file	Filename extensions
Email	Outlook emails, Pine mails
LATEX file	.text, .bib, .bbl
MS office file	.doc, .ppt
Text file	.txt, .PDF, .ps, .html/.htm
Image	.gif, .jpg/.jpeg, .png, .tiff, .eps, .bmp

Table 3-1. File formats scanned automatically by Semex

3.1.6 HAYSTACK

(Haystack) is an extensible semantic web browser, that aims to a better organization, navigation, and retrieval, of both personal and shared information such as emails, web pages, documents, and calendars. It allows people to define new object attributes, which helps categorizing their personal information by creating collections and provides related material with user's work helping the user in this way to find effectively what she needs. Haystack is based on the idea that merely providing access to information is insufficient and needs to be aggregated in meaningful semantic ways. Therefore, applications must help users filter and select information that is relevant to the user's needs. Relevance depends on factors such as the kind of information being looked at, the task's context the user is performing, etc.

3.1.7 SPONGE

(SPONGE), standing for Semantic Personal Ontology-based Gadget, is a desktop client application that extends the core Social Semantic Desktop services by supporting personal and group information and knowledge management. It provides functionalities for searching by using keyword queries, annotating desktop resources by creating RDF triples and collaborating on group support systems with semantic capabilities.

Starting with the Search gadget, an RDF repository search is implemented as soon as the user starts typing keywords. The retrieved results enriched with ontology neighboring instances and followed by their annotations, are presented in a web browser page, arranged in a table of Sponge Notes, another gadget which will be

discussed below. Sponge extends desktop search by allowing the user to remotely access and retrieve resources from peer desktops. Sponge Notes are used for annotating desktop resources. Implemented as RDF triplets, Sponge Notes allow extended ontology editing or annotation removal. Finally, collaboration between users is supported by means of semantic workspaces by storing, organizing and sharing resources needed for the accomplishment of personal and collaborative tasks. Semantic workspaces can support either the personal work of individual users or the collaborative work of groups. Users can browse resources of each workspace, view detailed information for selected resources as well as annotate resources manually or by exploiting system-generated recommendations.



Figure 3-5. Image stump from SPONGE's Search gadget

3.2 RELATED ONTOLOGY VISUALIZATIONS

The ontology visualizations described below, are Protégé plug-ins compatible with Protégé-OWL 4.0. For this thesis implementation, a graphic ontology visualization which has a similar look to mind maps is also necessary. This visualization should include nodes visualized as oval or square boxes and relations represented as lines connecting nodes. In this subsection, we describe some visualization tools implemented as Protégé plug-ins which look like mind maps.

3.2.1 NAVIGOWL

Starting with (NavigOwl), a visualization tool specially designed to explore ontologies. It is enriched with graph layouts that can be applied to the ontology in order to understand its structure. NavigOwl features include zoom-able user interface, mouse events handling (pan, drag, mouse-over, etc.), node searching, changeable node visibility state, tool-tips and other. The NavigOwl interface is shown in Figure 3-6. Finally, NavigOwl supports only *.rdf and *.owl ontology formats.

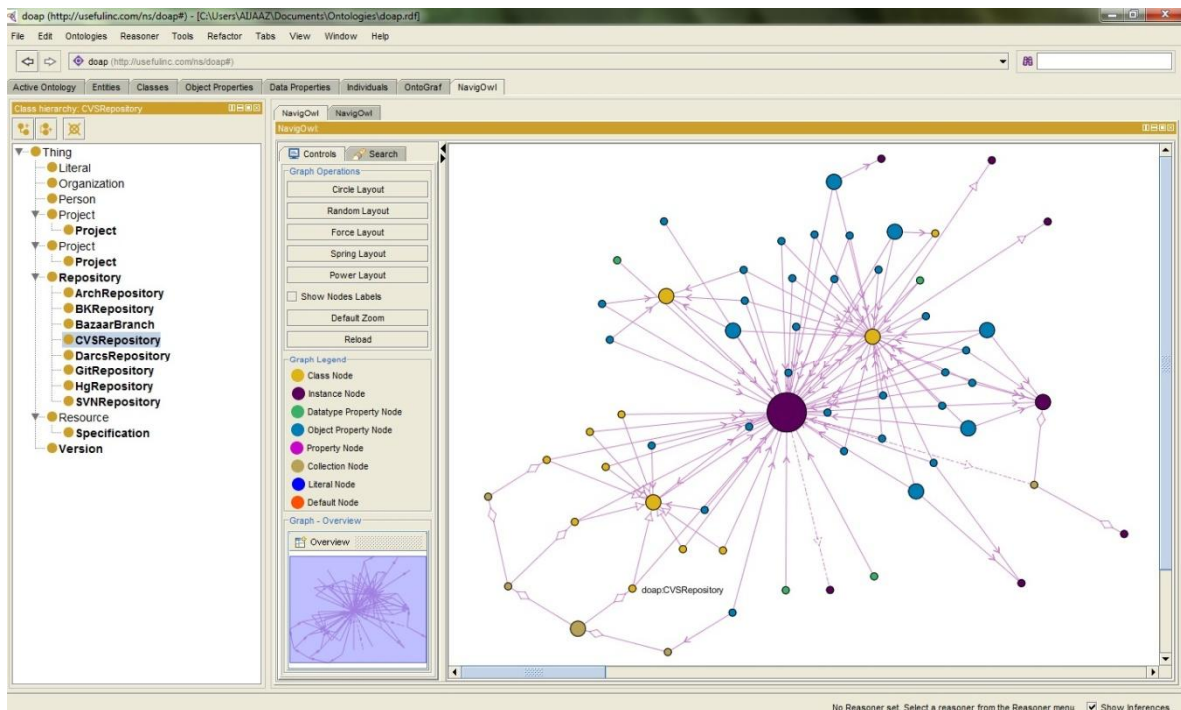


Figure 3-6. Image stump from the NavigOwl interface

3.2.2 OWLViz

(OWLViz) visualization plug-in is designed to be used with the Protégé-OWL plug-in and is based on (Graphviz) visualization. It visualizes class hierarchies of an OWL ontology as directed graph where the edges represent relationships between parents and children, allowing in this way the comparison of the asserted class hierarchy and the inferred class hierarchy. OWLViz uses the same color scheme as Protégé-OWL so that primitive and defined classes can be distinguished, computed changes to the class hierarchy can be clearly seen, and inconsistent concepts are

highlighted in red. Properties and individuals are not represented and nodes cannot be moved. However, the layout of the generated graph can be configured. Figure 3-7 shows the OWLViz visualization for a (Pizza ontology) part. The plug-in enables export of the graph as bitmap image, supporting formats like *.png, *.jpeg and *.svg. Finally, OWLViz is implemented as a view so that the subsumption graph can be shown on any tab.

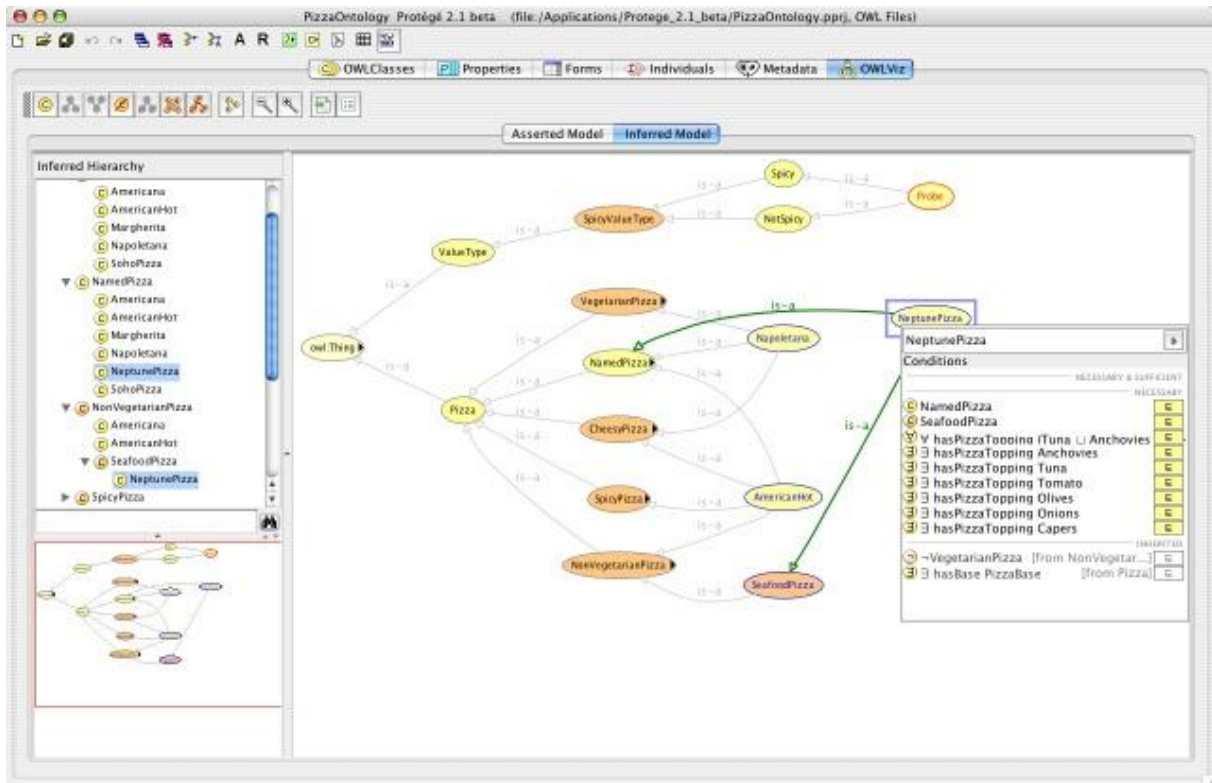


Figure 3-7. Part of (Pizza ontology) with the OWLViz visualization

3.2.3 ONTOGRAF

(OntoGraf) is another ontology graph visualization plug-in which has become a standard part of the Protégé-OWL editor. It has been developed in Stanford University, and makes use of the visualization library from the Protégé 3 plug-in (Jambalaya). OntoGraf supports interactively navigating the relationships of OWL ontologies. The OntoGraf Protégé plug-in is shown in Figure 3-8.

OntoGraf allows navigating through relationships of an OWL ontology. It visualizes classes and individuals (class instances) as nodes of a graph and relationships between them as edges. Hovering over the edges shows the type of the relationships

they correspond to, while terms can be expanded to provide incremental expansion of the graph. Various layouts are supported for automatically organizing the structure of the ontology, but also each node can be moved by the user. Edges and nodes can be filtered according to their types, to display only what is desired and nodes that become orphaned can be hidden. In this way, graph complexity is reduced allowing the user to navigate faster and easier. Finally, the user can take a bitmap image of the visualization or save the graph to a file to be loaded later.

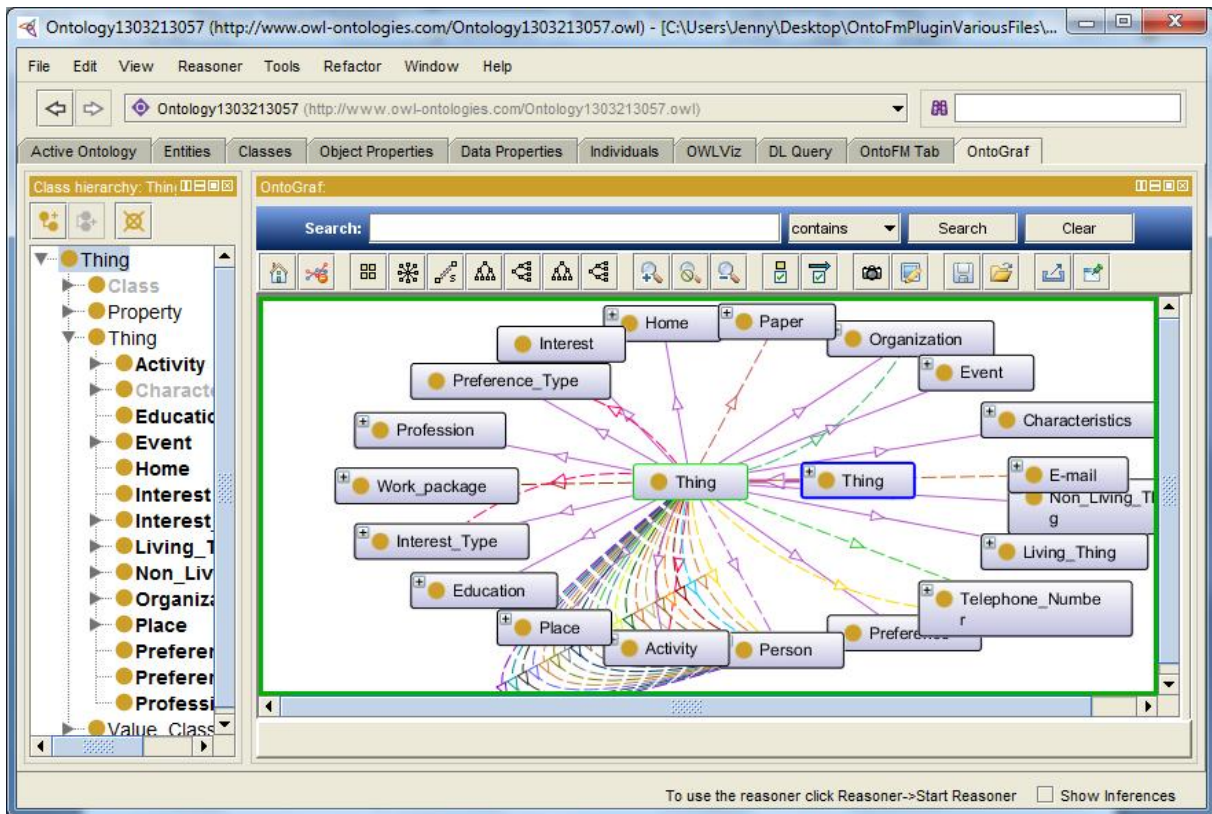


Figure 3-8. Part of the ontology described in (Katifori, et al., 2008) and (Golemati, Katifori, Vassilakis, Lepouras, & Halatsis, 2007) drawn by the OntoGraf plug-in

3.3 A BRIEF COMPARISON

The OntoFM personal ontology-based file manager implemented for this thesis, is a personal information management and retrieval system enriched with a mind map-oriented visualization to support user navigation within the personal information base. From the above related systems, two of them are well-known and one of them is close to the OntoFM search functionality. For this reason tables 3-2 and 3-3 summarize the main functionality each system (MIT Semantic File System, GoNTogle and, NEPOMUK) or

visualization offers to the user comparing to the OntoFM system and ontology visualization.

3.3.1 COMPARING RELATED PERSONAL INFORMATION SYSTEMS TO ONTOFM

As shown in Table 3-2, all systems allow the users to navigate through the ontology. Each system has a different way of approaching this functionality, either by a command-line like interface or by a tree showing the hierarchical relations. The OntoFM system offers this functionality graphically with a mind map-oriented visualization, offering in this way an easy understandable navigation method. The user is able to interact with the ontology visualization by expanding, collapsing or hiding nodes and apply filters or different layouts by using the ontology visualization toolbar offered. Moreover, unlike the other personal information management systems, OntoFM supports navigating through the file system catalogues and files by providing a common folder hierarchy pane, as employed by many file browsers, in order to keep not only the functionality but also user's familiarity. To end with the navigation functionality, OntoFM introduces an innovative support of selecting multiple folders in order to access their contents at once or limit the search results locations.

As far as annotating is concerned, all three related systems offer automatic annotation of files. Annotation is implemented either by transducers or semantic annotation components to either whole documents or parts of their text. However, only NEPOMUK supports any type of annotating of all types of files. Even though OntoFM offers a semi-automatic annotation, it also supports any type of annotating of all types of files.

Furthermore, all systems offer search functionality, with NEPOMUK and OntoFM having also available some advanced search options. Except for MIT Semantic File System, for which we have insufficient information, the rest three systems provide full-text searching, while apart from GoNTogle, the remaining systems have an auto-completion or autosuggestion feature, recommending to the user candidate search keywords or phrases.

	MIT SFS	GoNTogle	NEPOMUK	OntoFM
Navigation				
File System Navigation	?	✗	✗	✓
Multiple Folder Selection	✗	✗	✗	✓
Ontology Navigation	✓	✓	✓	✓
Graph visualization	✗	✗	✗	✓
Interactive Graph Visualization	✗	✗	✗	✓
Graph Visualization Tools	✗	✗	✗	✓
Annotation				
Automatic Annotation	✓	✓	✓	✗
All Types of Files Supported	✗	✗	✓	✓
All Types of Annotations	✗	✗	✓	✓
Retrieval				
Auto-completion	✓	✗	✓	✓
Full-text Search	?	✓	✓	✓
Advanced Search Options	✗	✗	✓	✓
Ontology Editing				
Ontology Editing	✗	✗	✓	✓

Table 3-2. Summary table concerning the differences among MIT SFS, GoNTogle, NEPOMUK and, OntoFM (✓=supported, ✗= not supported, ?=not mentioned)

Finally, as shown in the table above, only NEPOMUK and OntoFM offer the user the ability to edit the ontology by adding new instances or relations between two existing instances.

3.3.2 COMPARING RELATED VISUALIZATIONS TO ONTOFM VISUALIZATION

Before starting with the comparison of all four visualizations, it should be mentioned that the OntoFM visualization is an extended Ontograf visualization, edited to serve the OntoFM system purpose. This leads to the result that these two visualizations are much alike to each other.

As shown in Table 3-3, all four visualizations offer a zoom-able user interface, color characterization for recognizing visually in an easy and fast way classes and instances, choosing among various layouts and a graph overview either by zooming out

or by viewing a small ontology overview pane. While OWLViz cannot handle mouse events and does not support node searching, NavigOWL is not able to export a graph. Moreover, even though both OWLVIZ and NavigOWL do not offer node or edge filtering, Ontograf and OntoFM do not offer a node visibility state, but only for orphan nodes. Apart from that, both OWLVIZ and NavigOWL do not offer node pinning and, loading or saving a graph. Finally, except for the OntoFM visualization, none of the related visualizations supports different node coloring or icons for nodes that represent files or folders.

	NavigOwl	OWLViz	OntoGraf	OntoFM visualization
Zoom-able User Interface	✓	✓	✓	✓
Mouse Events (drag, mouse-over, etc.)	✓	✗	✓	✓
Node Searching	✓	✗	✓	✓
Node Visibility State	✓	✓	✗	✗
Color Characterization	✓	✓	✓	✓
Various Layouts	✓	✓	✓	✓
Graph Overview	✓	✓	✓	✓
Export graph	✗	✓	✓	✓
Node filtering	✗	✗	✓	✓
Edge filtering	✗	✗	✓	✓
Node pinning	✗	✗	✓	✓
Load/Save graph	✗	✗	✓	✓
Node Icons related to file type	✗	✗	✗	✓

Table 3-3. Summary table concerning the differences among NavigOWL, OWLViz, OntoGraf and, OntoFM ontology visualizations (✓=supported, ✗= not supported)

3.3.3 COMPARING RESULTS

As resulted by the above comparisons, OntoFM system does not differ much from the other related systems. The same comparing result was extracted for the OntoFM visualization too. However, OntoFM is a unified personal information management and retrieval system enriched with a mind map-oriented visualization. Due to the fact that

the idea of OntoFM is innovative, there is no related system to OntoFM. This leads to the need of “dividing” OntoFM into two main parts in order to make comparing feasible.

To this end, OntoFM is a novel combined system, which if compared to the other personal information management systems, it offers further functionality regarding managing, retrieving and visualizing information and more features than all related systems at once.

4

PREVIOUS EFFORTS

PREVIEW

- Earlier implementation of OntoFM file manager
- Problems occurred
- Solutions given

In this chapter, previous efforts concerning the OntoFM file manager are described. Also, problems faced and techniques followed to overcome these problems are mentioned.

In this chapter we describe an earlier implementation of OntoFM, namely what it consists of and how it works. We also list the main problems faced and techniques followed to overcome these problems.

4.1 EARLIER IMPLEMENTATION OF ONTOFM

OntoFM development started in 2010 for my Bachelor of Science thesis. As mentioned in the introduction subsection, OntoFM is a tool that allows semantic searching on a personal ontology which represents the user's personal information space. The tool is an ontology-based file manager that exploits the ontology relations to locate and display concepts associated to specific files, proposes new related concepts to users, and helps them explore the information space to locate the required file.

OntoFM was crafted as an application connected to a (Sesame) server, through which the ontology information could be retrieved using SPARQL queries. Depending on the user activities, OntoFM created suitable queries which were sent to the Sesame server; subsequently OntoFM received the results and processed them accordingly, to populate the user interface widgets with which the application user interacts.

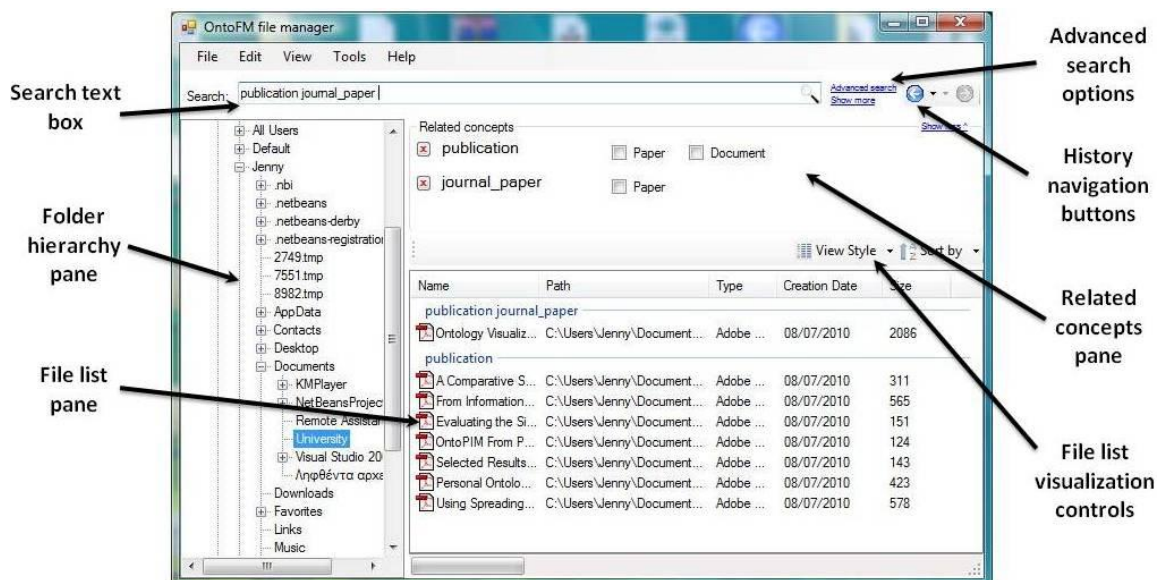


Figure 4-1. OntoFM screen with captions - Inputting multiple keywords. Entered terms appear in the related concepts pane along with directly connected concept instances.

Figure 4-1 depicts the OntoFM user interface with the file system view pane on the left and the results view pane on the right. On top of them a search bar allows the

user to search within the file system or personal ontology and suggests candidate search terms as soon as the user starts typing keywords. When a search is implemented, a hidden related concepts pane appears, showing all related to the keywords concepts. For each related concept, a list of first level related concepts is shown too.

To make that clear, Figure 4-1 is an example search image stump, showing the case when the user searches for files related to the keywords “publication journal_paper”. Apart from the related files, the OntoFM application retrieves the first level related concepts for both “publication” and “journal_paper”. For this example, for “publication” it retrieves “paper” and “document” first level related concepts.

The two main functions implemented by OntoFM were browsing through catalogues and files and searching through the file system using semantic metadata supplied by a personal ontology. Moreover, the user was able to select multiple folders and view their contents at once. Other features include customizing the presentation to view all file details in tubular format, file icons or simple file lists and multiple selection of folders to limit the search only to the selected ones. Finally, time related queries can be performed in two ways. The first one, is indirectly through the file list pane, when the details view is selected, by sorting files using one of the time related columns (created or modified) and the second one is directly by using the “Date” concept modeled in the personal ontology and entering a constrain on it.

4.2 PROBLEMS AND SOLUTIONS

Even though the OntoFM file manager was functional, the time needed for retrieving all the query results was not tolerated by the user. Every user wants immediate or almost immediate system response and the OntoFM could not satisfy that need. For this reason, we modified the tool architecture so that the personal ontology would be stored locally. However, while creating the module which would retrieve the necessary information from a locally stored personal ontology, a new problem came up to stop that effort too. The problem was related to the libraries needed for important functions, such as searching into the personal ontology or filtering results by type; these libraries were bundled within the Protégé executable and were therefore not available for use by standalone applications. Contacting with the (Protégé team) ended up to a

(Protégé bug report, 2011) asking to make Protégé 4 UI components available for use by standalone applications.

Meanwhile, the development of OntoFM continued, this time as a Protégé plug-in instead of a stand-alone application. This is a temporary solution to overcome the problems concerning time and libraries needed, until a solution is available.

5

DEVELOPMENT PROCESS

PREVIEW

- Assumptions
 - General assumptions
 - Visualization assumptions
- Specifications
- Architecture
- Implementation

This chapter describes the development process of OntoFM plug-in for Protégé. Necessary assumptions that were made are described and various requirements to implement prototype are listed. Moreover, the OntoFM's architecture and implementation information are commented.

This chapter describes the development process of OntoFM plug-in for Protégé. Necessary assumptions and specifications that were made are commented. Moreover, the OntoFM's architecture and implementation information are outlined.

5.1 ASSUMPTIONS

Before proceeding to the enumeration of requirements for the OntoFM plug-in, we need to set some assumptions.

5.1.1 GENERAL ASSUMPTIONS

First of all, we assume that the personal ontology which describes the personal information space of the user, as described in (Katifori, et al., 2008) and (Golemati, Katifori, Vassilakis, Lepouras, & Halatsis, 2007), already exists and is in use. The classes of personal ontology concepts are divided into two main groups, which constitute the top-level ontology concepts, namely «Thing» and «Value Class». «Thing» class contains all concepts within the field of discourse (i.e. the personal information space), which may be either abstract or concrete. «Value Class» class contains all possible values within the field of discourse. The purpose of adding this class was the need to separately represent and store elementary data elements such as personal names, phone numbers, etc. Under this approach, not only elementary data is distinguished from more complete information (such as instances of concepts) and can be subject to formatting rules, but additionally is allowed to have internal structure, which would not be possible if they belonged in «Thing» class.

Apart from the above, the personal ontology includes a «File» class referring to a file or folder from the file system hierarchy, or to instances of concepts such as «Person», «Project», «Location» and others, which may be related to instances of the concept «File». For each file and directory within the physical system hierarchy, there is a corresponding instance of the concept «File» that represents it. Therefore, the hierarchy of files and directories is still available to user, but in a different semantic way through the personal ontology. Approaches to management and maintenance of

personal ontology can be found in (Fernandez-Garcia, Sauermann, Sanchez, & Bernardi, 2006) and (Lepouras, et al., 2006).

Finally, as mentioned above, we assume that each file and directory has a corresponding "File" instance in the personal ontology. By this assumption we ensure that there is no need for the user to create a new "File" instance in the personal ontology. This leads to the decision of disallowing the user to create or delete instances or concepts. However, information concerning our everyday life change, so it is imperative to allow the user to change the ontology by letting her relate "File" instances to other instances or concepts. In this way the personal ontology's structure cannot be dramatically changed so that it ends up inappropriate to serve its purpose. Still, the user can modify the ontology in order to serve her daily needs. At this point it should also be mentioned that the reverse assumption is not necessarily true as it is permissible for instances or concepts in the ontology to be unrelated to any file.

5.1.2 VISUALIZATION ASSUMPTIONS

Based on the previous general assumptions, we proceed to visualization assumptions such as setting basic design issues and investigating various design strategies.

The first and probably most fundamental question for the OntoFM plug-in interface is whether the personal ontology should be visualized, allowing user to interact directly with ontology concepts and select files linked to the instances of concepts previously selected. The use of personal ontology serves a different file organization based on the relations between concepts, which might make finding a file of known content simpler, faster and easier compared to searching in a complicated and probably counter-intuitive file system with hierarchical structure and no cohesion. The notion of using ontology visualization methods as browsing aids has been explored in (Katifori, Torou, Halatsis, Lepouras, & Vassilakis, 2008) and (Katifori, Halatsis, Lepouras, Vassilakis, & Giannopoulou, 2007). (Alani, 2003) suggests TGViz as a visualization technique which may be chosen to employ for browsing the concepts in the personal information space while presenting the file system hierarchy. However, a more

enhanced and new ontology graph visualization plug-in as OntoGraf, which was discussed in 3.2.3 subsection, is recommended for the OntoFM plug-in.

The reason of recommending OntoGraf and not some other ontology visualization from the ones discussed before, is that we can observe several similarities with mind mapping. Starting from the fact that both are graphical methods, they both help distinguishing words or ideas, with colors and symbols. Mind maps are generally hierarchically organized with ideas branching into their subsections. On the other hand, OntoGraf displays not only the hierarchical structure of concept classes but also the instances of classes and the object properties. Finally, an extended OntoGraf visualization can be treated as a mind map if it focuses on multiple concepts. The similarities outlined above make OntoGraf ideal to use for the ontology visualization for the OntoFM plug-in.

However, the visualization of two hierarchies, one physical and one semantic, risks making navigation more difficult. On the side of the physical structure, each user has organized the files in a certain way, through which the user can easily and rapidly locate each file. In case of an often used file, the steps of finding and accessing the file in the personal computer may be performed even mechanically. Regarding the ontology, a file search could be an entirely different process and perhaps more complicated, causing confusion to the user, who would wander which hierarchy would be the best to use in each case.

For the above reasons, we decided to keep both hierarchies visible and add show/hide buttons for the beginner users. We also decided to have a single pane to display the files list (if navigation was performed) or search results (in case a search was performed). The file list widget was placed in the center of the application window, since it can constitute a link of the two hierarchies. A similar decision was made for the search bar too, as the user can search keywords consisting of file or folder names, concepts and instances of the personal ontology and retrieve the results in the result pane.

We hope that this design assumptions with the new design, will offer new possibilities in searching and organizing files and more familiarization with the personal ontology. The hide/show buttons help not only beginner users by offering a familiar environment, but also advanced users to perform more complicated processes. As noted

in (Henderson, 2009), an effective user interface management of personal files should provide a convenient, fast and powerful full text search, without involving the user's files and their organization. (Sauermann & Heim, 2008), found that in the personal environment, simple has-part and is-related relations are sufficient for users to file and re-find information.

To this end, the plug-in created in the context of this thesis is an extended common file manager user interface with the folder hierarchy on the left and the tabular file list on the center, with an ontology view pane on the right, complemented with an ontology-based searching mechanism.

5.2 SPECIFICATIONS

In this subsection we list the functional specifications of OntoFM. The specifications are listed in descending order of necessity, with the first ones being considered indispensable and the last ones less critical.

Firstly, we need to provide all the available features supplied by a common file browser. This means that functions such as browsing and searching in the file system hierarchy are necessary for the user. Since an ontology-based file manager is developed, the use of an existing ontology in the above mentioned functions is equally necessary, so that the results returned are not only structurally related (via the file system hierarchy) but also conceptually (via the ontology links).

Starting with the essential function of browsing, the user should be able to browse in the file system hierarchy and view each folder's contents. To achieve this, it is necessary to use a pane depicting the file system hierarchy and a pane which contains the contents of each selected folder, similarly to Windows Explorer. Moreover, features such as opening a file or accessing context menus should be available to the user, providing a smooth transition from the common file browser to the OntoFM plug-in.

Secondly, the returned search results should not only be structurally relevant but also conceptually. This can be implemented through the use of a personal ontology by linking files to concepts. As mentioned in the General Assumptions subsection, the personal ontology is in use and already populated with all necessary "File" instances

representing all files in the file system. The search results are displayed in the same result pane with the selected folder contents, as discussed in the Visualization Assumptions subsection.

Displaying the concepts related to the search keywords is also important, as the user can navigate easily through concepts to locate files of interest, or imposing restrictions to effectively limit the amount of results. As discussed in the Visualization Assumptions subsection, displaying the ontology or part of it is a complex issue, and can be classified as an “average necessity” feature. Moreover, updating the ontology by relating files to other instances or concepts, in order to keep the ontology updated to everyday needs, is an essential feature which can be also classified as an “average necessity” feature.

Search options such as selecting multiple folders or adding search criteria can be classified as “average necessity” features as well. When having selected multiple folders, the user must have access to a list containing all the selected folders’ paths (path labels) so that she has an overall view or be able to unselect a folder. A delete button in front of each path label of the list may be useful for quick and easy removal of the specific path from the selected locations. At this point, we can assume that each path label is a search criterion which imposes the result files to be contained in the selected folder. In this way, features such as inserting or deleting search terms could be implemented easily and quickly. Another search criterion could be the capability of choosing among searching in filenames, contents or concepts in order to perform an effective search.

5.3 ARCHITECTURE

As mentioned in subsection 2.3, Protégé was proposed as a means to interact with the personal ontology for supporting functionalities such as browsing, searching or managing the personal ontology. Apart from the fact that Protégé is the most widely-used ontology creation tool on the market, there are more reasons for making this choice. First of all, Protégé is not only an open source platform which is freely available but also easy extensible as it integrates the necessary and comprehensive tool suites for ontology development. Also, language independence makes the development process

easier. Finally, since the first Protégé tool was created in 1987, there is a significant amount of research and experiment that may be used as references.

The main idea was to create a standalone application, but due to the problems occurred in the previous efforts, as described in 4.2 subsection, OntoFM is implemented as a Protégé TabWidget plug-in, appearing as a tab in the Protégé user interface.

OntoFM has been built according to a layered architecture, as illustrated in Figure 5-1. The bottom layer is the information store, comprising of the file system (as offered by the underlying operating system) and the personal ontology, which hosts the concepts that are important to the user and the associations between these concepts and the items within the file system (files and directories). The personal ontology is represented and stored using the Protégé open source ontology editor and knowledge-base framework.

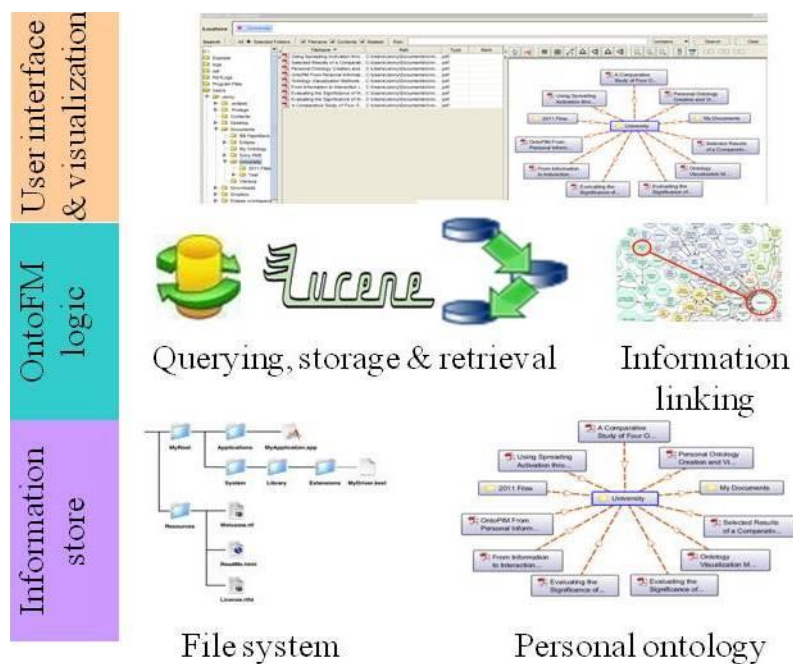


Figure 5-1. OntoFM architecture scheme

On top of the information store lays the OntoFM logic layer, which implements the functionalities of (i) *information querying*, i.e., finding file system items associated with given concepts, (ii) *linking of information*, i.e., allowing the user to associate selected items or item hierarchies with concepts, and (iii) *fetching and updating data* from the information store. The full-text search engine Lucene is used for the indexing

and fast retrieval of the ontology elements, the file/folder names, and the associated metadata.

Finally, the top layer (user interface and visualization) collects user input and displays results to the user. The ontology visualization is based on the Ontograf plug-in, which has been extended to support the additional functionality offered by the file manager and tailored to display the ontology according to the mind map paradigm. Mind mapping-inspired ontology visualization and interaction allows users to manipulate the ontology without requiring detailed technical knowledge in ontology building. In this way, we offer semantic expressiveness by exploiting the inherent mind mapping connections to enrich and present the user's information space.

5.4 IMPLEMENTATION

In order to develop the OntoFM plug-in, several programming tools and a development and execution platform were required. The following subsections, describe the platform and tools and the libraries used to implement OntoFM.

5.4.1 PLATFORM AND TOOLS

OntoFM plug-in is implemented in Java programming language and as Protégé does, it requires 1.6 version of JRE installed. The main development platform is (Eclipse IDE for Java Developers). This platform is preferred because it has an open-source license, a simple yet powerful user interface and many conveniently accessible features. Moreover, it is faster than other platforms and time saver due to the auto-compilation feature offered.

5.4.2 LIBRARIES

Efficient and fast searching through the entire ontology could not be implemented without Lucene, a powerful text search engine API. As mentioned in 2.3.2 subsection, Lucene is high-performance, scalable, full-featured, open-source, and written in Java. As far as the ontology tasks are concerned, (OWL API) and (Protégé OWL API)

libraries proved beneficial for the OntoFM implementation. Last but not least, the OntoGraf libraries helped visualizing the ontology in an extended mind map way.

6

ONTOFM PLUG-IN PRESENTATION

PREVIEW

- OntoFM plug-in overview
- OntoFM plug-in functionality
 - Browsing
 - Searching
 - Maintaining ontology

This chapter describes an overview outline and a functionality list using image stumps to illustrate the plug-in's functionality and to better explain the features provided.

6.1 OVERVIEW

Having all the visualization assumptions, as presented in 5.1.2 subsection, and specifications, as presented in 5.2 subsection, in mind, the OntoFM plug-in has an extended common file manager user interface. As shown in Figure 6-1, the OntoFM user interface consists of three main panes: the *folder hierarchy pane* on the left, the tabular *file list pane* on the center and the *ontology view pane* on the right. The folder hierarchy pane serves to access the physical file folder hierarchy and browse through the user's catalogues. The file list pane avails to access either the files contained in one or more folders or the files related to some terms in case a search is implemented. The ontology view pane serves to overview how a folder, file or search term is related to other items of the file system or the personal ontology.

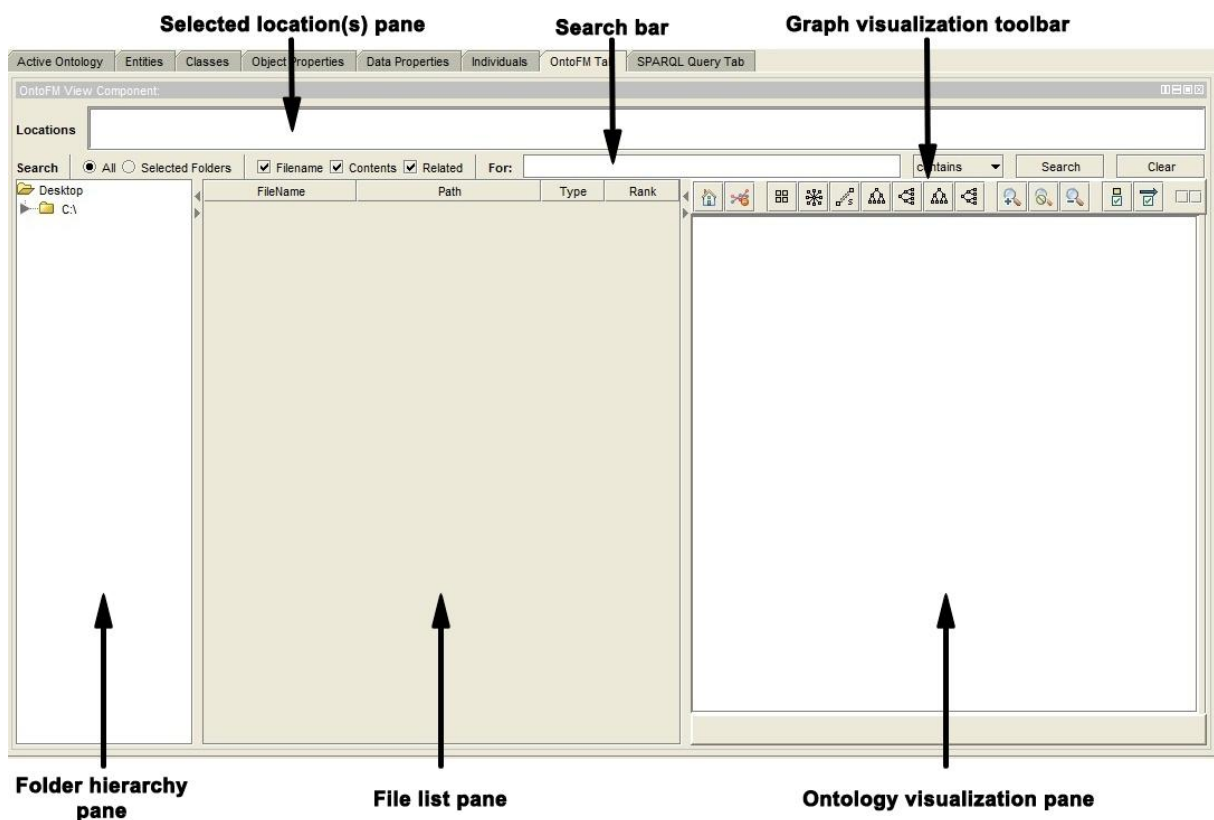


Figure 6-1. OntoFM initial screen with captions

Apart from these three panes, there are also some other objects that enrich the OntoFM's functionality. Starting with the *search bar*, it is located on top of these panes; decision which was documented in 5.1.2 subsection. The search bar consists not only

from the search box but also from smaller panes which provide additional tools with further functionality options, such as selecting the search region (all or selected folders options) or search scopes (search in filenames/contents/related options). Next is the *location pane* which is located above the search bar, allowing the user to overview the folders selected from the folder hierarchy pane. Finally, a *graph visualization toolbar*, located in the graph visualization pane, contains the graph visualization controls.

6.2 FUNCTIONALITY

OntoFM plug-in incorporates a number of functions and services, which could be categorized in three main groups: *browsing*, *searching* and *maintaining* functions. Each group, including its subcategories, will be explained below by using plenty image stumps to illustrate the plug-in's functionality and to better explain the features provided.

6.2.1 BROWSING THROUGH CATALOGUES AND ONTOLOGY ITEMS

Starting with the simplest task a user can perform, browsing may be implemented by using the file hierarchy on the left, navigating to location by selecting the desired folder and browse through files. When selecting a folder, the file system hierarchy gets expanded and all first level subfolders are displayed. Also, the file list pane contains all files located in the selected folder and the ontology view pane gets updated so that it shows all the current relations to other files, folders, concepts or instances of the selected folder.

Figure 6-2 shows an image stump of a browsing case. As soon as the user navigates through the folder hierarchy pane to the "University" folder and selects it, the file list and the ontology visualization panes get updated to show the corresponding information. In this example, the file list pane contains all the files located under the "University" folder and the ontology visualization pane visualizes part of the personal ontology, having the "University" folder instance node in the center and all directly related items around it represented by nodes and connected by arrows. Each node has a different icon in front of it, representing its type. When a node represents a concept, a yellow circle is placed in front of the node label, whereas when it represents an instance,

a purple rhombus is located in the same spot. In case the node represents a folder, a folder icon (📁) is placed in front of the node label whilst in case the node represents a file, a system shell icon depending on file's type is shown. In Figure 6-2 browsing case, all files retrieved have the same icon as they all have a Portable Document Format (*.pdf). As far as arrows are concerned, each arrow has a different color encoding, representing in this way the type of each relationship. In this image stump, all arrows have the same color as they represent the same type of relationship, "has_Location". However, the arrow directions indicate the existing hierarchical relation; i.e. "2011 Files" is contained in "University" folder which has location in "My Documents" folder.

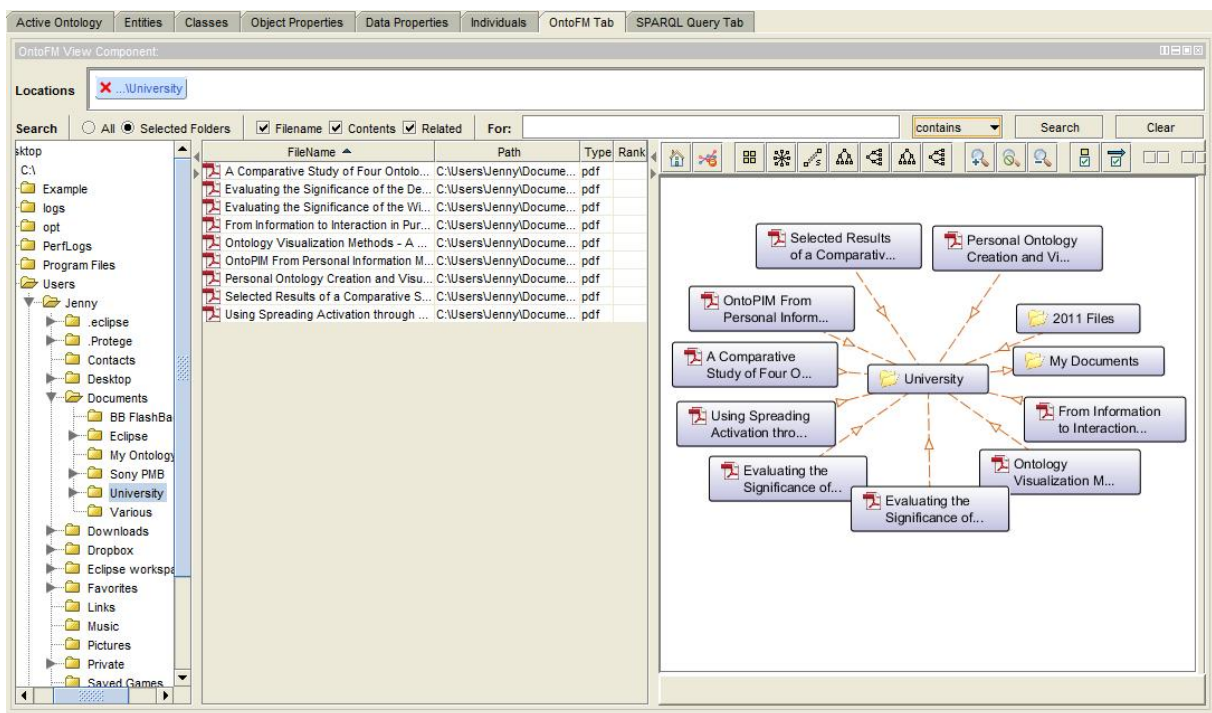


Figure 6-2. Browsing the file system hierarchy. Contents of the selected folder "University" at the center and corresponding ontology visualization on the right

In case the user wants to navigate through the concepts and instances of the personal ontology, she has to double click on a node, so that the ontology visualization gets expanded. In Figure 6-3, the user has double clicked the "Ontology Visualization Methods – A Survey" to view all the relations to other items such as files, folders, people, etc. In this image stump, there are arrows with different coloring representing relations such as "has_author", "has_individual", etc.

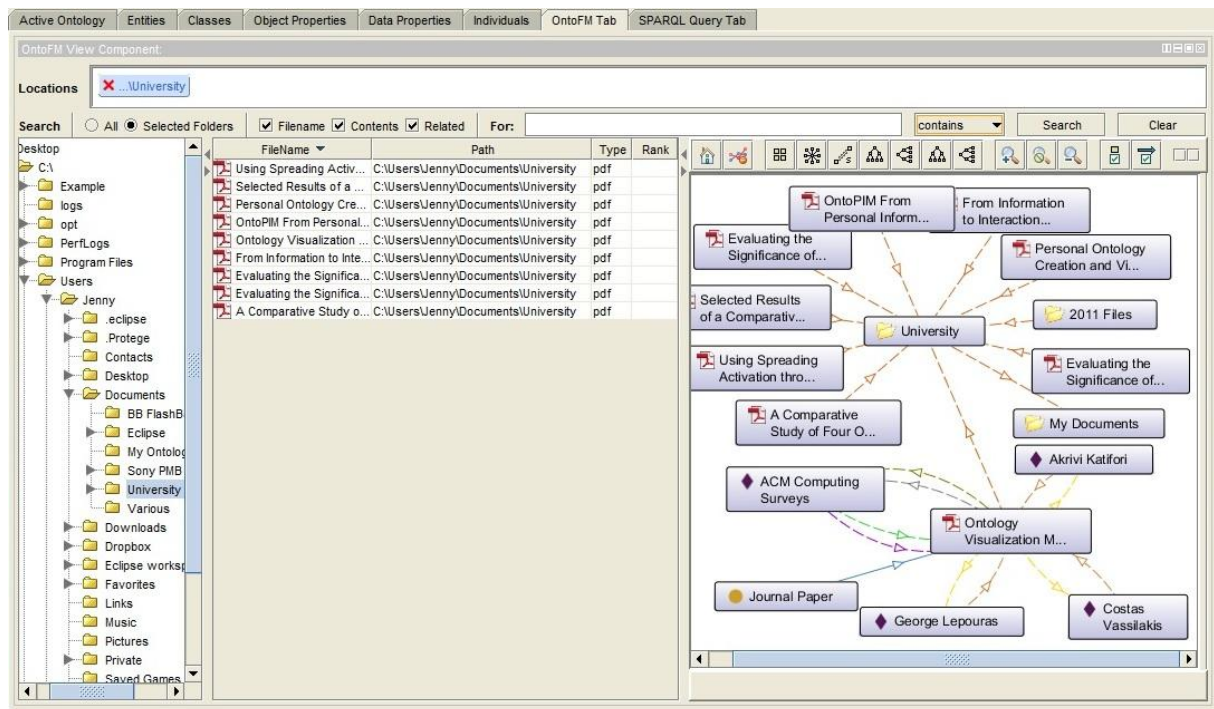


Figure 6-3. Browsing the personal ontology - Visualizing all relations to "Ontology Visualization Methods - A Survey" after browsing to the "University" folder

As long as the user continues navigating, at some point the ontology visualization will be cluttered. The OntoFM plug-in anticipates this possibility by offering some ontology visualization tools. Figures 6-4 and 6-5 show some ontology visualization options which allow the user to limit the types of nodes or arrows shown. By clicking on a checkbox, not only the checkbox gets unchecked but also the corresponding nodes or arrows get hidden. Apart from these two options, the user may hide all orphan nodes, change the visualization layout to one of the supplied ones (alphabetical grid, radial, spring layout, vertical or horizontal tree, vertical or horizontal directed tree), zoom-in or out by scrolling or pin nodes, open or save a graph, etc by pressing the available buttons.

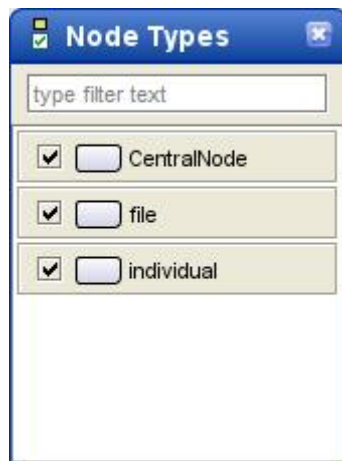


Figure 6-4. Ontology visualization options - Show specific node types



Figure 6-5. Ontology visualization options - Show specific arc types

6.2.2 SEARCH AND LOCATE FILES

One of the most common functions a file browser offers is to help users locate files in their personal file system. Depending on the information already known by the user we can categorize the file hierarchy exploration task in two basic categories.

6.2.2.1 Perfect Exploration

The first category is *perfect exploration*, in which the user knows both the file name and location. When this information is available, the task of locating a file is ideal and straightforward. As described in the browsing case above, the user can use the file hierarchy on the left, navigate to location by selecting the desired folder and browse through files to find the one required.

6.2.2.2 Limited/Incomplete/Partial exploration

When the user does not have all the necessary information for locating the needed file, she has to go with a *limited / incomplete / partial exploration*. In this case, finding a file is more difficult than in the perfect exploration case. Depending on the available information, there are three possible cases:

- Location is known.
- All or part of the filename is known.

- Metadata such as author, keywords from the content, date and other, are known.

Depending on the above circumstances, there are some techniques which may be used in order to locate a file. These techniques are going to be described below.

The case of having available only the file location can be likened to the perfect exploration case as even if only the location is known, the file hierarchy may be used to locate the folder, and subsequently files within the folder may be browsed to locate the requested file. Likely before and starting from the top, when the file location is known, the user can navigate to the specific folder by leveraging the folder hierarchy pane on the left. As soon as the folder gets selected, the file list pane located in the center of the plug-in set-up, shows all files located under it, allowing the user to browse through them. When the desired file is found, the user can double click it, as in a common file browser, to open or execute it. At this point, it should be mentioned that when browsing in a folder's contents, a file preview may be useful for an easier, quicker and more visualized search.

Continuing with the case when location is not known, the search facility may be employed. As mentioned in 5.1.1 subsection, since there is a «File» concept in the personal ontology, all file names are included to every implemented search. Taking advantage of the search engine provided, the user may type all or part of the file name in the search box. As soon as she starts typing, a suggestion list appears automatically, containing all possible keyword terms and making search easier through the entire system. Figure 6-6 image stump shows the autosuggestion options which match to the typed by the user letters "ont".

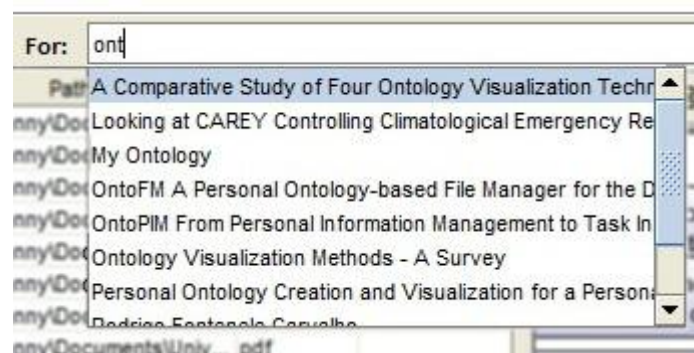


Figure 6-6. Autosuggestion feature

After selecting a keyword or phrase from the suggestion list or typing one or more terms, the user may press the “Enter” button of her keyboard or click the “Search” button supplied by OntoFM’s user interface. The file system will be scanned and all related to keywords files will be retrieved. Figure 6-7 is a limited / incomplete / partial search case example in multiple selected folders where the user knows that the file name contains the keyword “ontology”. The file list pane contains all files related to the keyword “ontology”, showing information not only about their location and type but also their ranking concerning the keywords entered. The file list is sorted by ranking in descending order, having in this way the most related file to the search keywords first. The ontology visualization pane shows a corresponding part of the personal ontology, having a pseudo node in the center of the visualization representing the keywords entered by user and all directly related items around it. Each of the related items is connected to the center node with a pseudo relationship “has result”.

The screenshot displays the OntoFM View Component interface. At the top, there are tabs for 'Active Ontology', 'Entities', 'Classes', 'Object Properties', 'Data Properties', 'Individuals', 'OntoFM Tab', and 'SPARQL Query Tab'. Below the tabs, the 'Locations' section shows two selected folders: '..\University' and '..\2011 Files'. The 'Search' section is set to 'All Selected Folders' with search criteria for 'FileName', 'Contents', and 'Related', and the search term is 'ontology'. The search results are displayed in a table with columns for 'FileName', 'Path', 'Type', and 'Rank'. The results list several PDF files related to ontology, such as 'Ontology Visualization Metho...', 'Latest Developments in KC...', 'From Information to Interactio...', 'OntoFM A Personal Ontology...', 'Personal Ontology Creation ...', 'Selected Results of a Compa...', and 'A Comparative Study of Fou...'. The ontology visualization pane on the right shows a central node labeled 'TERMS: ontology' connected to various related items, including 'Ontology Visualization M...', 'From Information to Interaction...', 'A Comparative Study of Four O...', 'Personal Ontology Creation and Vi...', 'Selected Results of a Comparativ...', 'Akrivi Katifori', 'Elena Torou', 'Constantin Halatsis', 'ISWC 2011', 'Jenny Rompa', 'Latest Developments in...', 'OntoFM A Personal Ontology-based...', 'ACM Computing Surveys', 'George Lepouras', 'Christos Vassilakis', 'My Ontology', 'Christos Tryfonopoulos', and 'IV 2006'.

Figure 6-7. Search for “ontology” relevant items in multiple selected folders

The last case of the limited / incomplete / partial exploration is when file’s metadata such as the author, keywords from the content, date, etc, are known by the user. Current file managers provide searching mechanisms for files with specific metadata, or even specific content (full text search). The user can combine search keywords and other metadata such as file type or date created. In some file managers,

search criteria may be progressively refined to limit the number of results. However, search is not always easy as many files could be retrieved while other metadata may not filter results enough, or even the user may not remember exact keywords. To this end, ontology based search was suggested to alleviate such search problems.

Figure 6-8 is a continuation of the previous use case and an example of this limited / incomplete / partial exploration case. Having available all files and items related to the “ontology” term from a previous accomplished search, the user wants to limit her search by adding the term “iswc” as she knows that the requested file was accepted in the ISWC conference. The file list and the ontology visualization panes get updated and show the narrowed down search results.

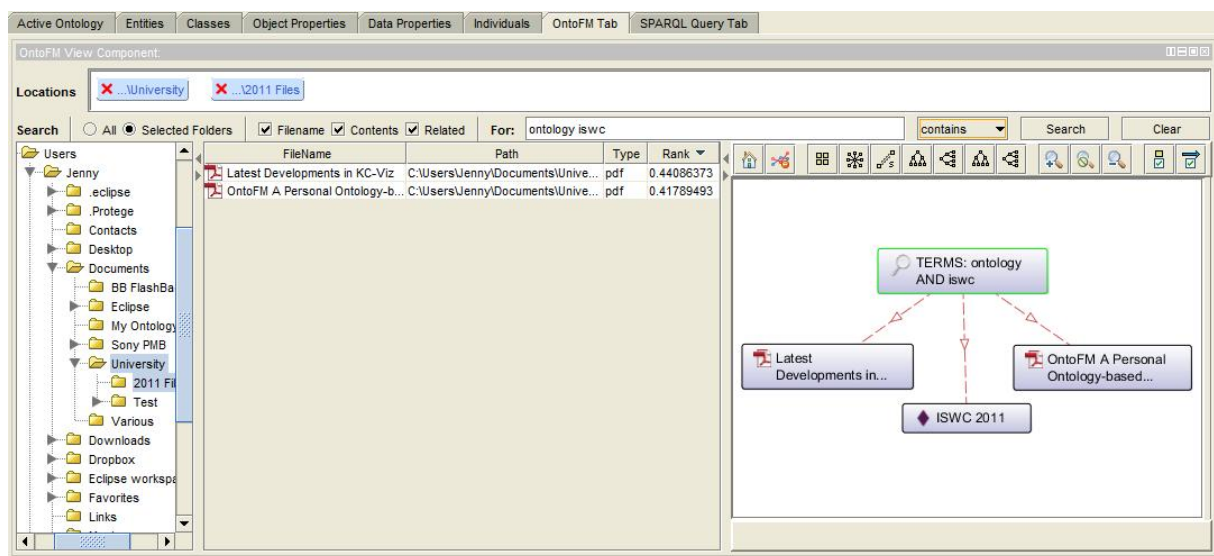


Figure 6-8. Search for "ontology iswc" relevant items in multiple selected folders

6.2.3 ADVANCED SEARCH

Searching a file is not always easy as in many cases a large amount of results is retrieved. The OntoFM plug-in provides the user with several advanced search options, such as searching in specific folders or searching not only file names but also contents or relations.

6.2.3.1 Select Multiple Search Folders

As mentioned in a previous example user case, the OntoFM plug-in offers a new feature allowing the users to select multiple folders while browsing or searching. Figure

6-9 shows an example case where the user has selected multiple folders. In order to achieve multiple selection, the user must hold down the Ctrl key from her keyboard and then select the desired folder. This procedure could be repeated in case the user wants to select more folders from the file hierarchy list. With multiple folder selection, more than one folders can be selected giving the user a chance to have all files contained in the selected folders available at once. When a folder gets selected, the OntoFM plug-in displays all files contained to the folder and inserts a path label to the locations pane with the new selected folder. Before each path label, a removal icon is shown, providing a fast removal method in case the user wants to unselect one folder but keep the rest of the selected ones.

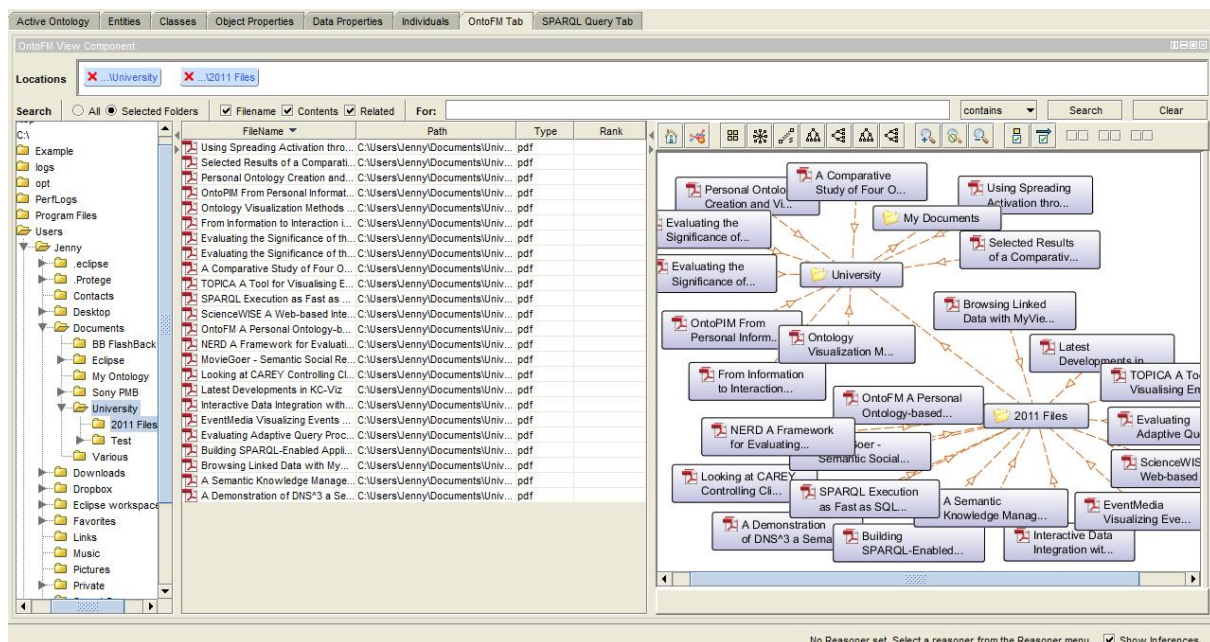


Figure 6-9. Multiple folders selection - "University" and "2011 Files" have been both selected to view their contents

This feature is useful as in this OntoFM plug-in version selecting a folder and showing all files located in its subfolders recursively is not feasible. Nevertheless, this task can be simulated by exploiting the multiple folder selection capability through selecting all the desired subfolders. Moreover, in case the user needs to limit her search to specific folders only, she can select the desired ones from the file system hierarchy on the left and filter her results straight ahead. Initially, all folders of the user's personal space are selected for searching. However, when the user selects one or multiple folders, the «Search» option automatically changes from «All» to «Selected Folders».

At this point, the following case should be discussed: Each personal computer has many files with -usually- different names. The single restriction imposed is the non-existence files with same name (including the extension) in the same folder. However, their existence in a different folder is allowed. When the user selects multiple folders, it is possible that more than one of these folders have the same name (but different file system path). In this case, the path labels in the locations pane would be indistinguishable, and therefore the user could not easily identify and remove the desired folder. To alleviate this problem, the system displays the full folder path when the mouse hovers over the path labels in the locations pane.

6.2.3.2 Search criteria

Searches may vary regarding the terms entered in order to locate the requested file. The keywords may be an entire file name or part of it, part of the file's content (for example in text files) or even concepts that are linked with the desired file.

To encounter all these cases, the OntoFM tool provides options to control whether the search includes file names, file contents or concepts associated to files. These options are available as checkboxes to the user so that more than one search scopes may be selected.

6.2.4 RELATE FILE TO ITEMS

As mentioned in 5.1.1 subsection, we assume that each file and directory has a corresponding "File" instance in the personal ontology, so that there is no need for the user to create a new "File" instance in the personal ontology. However, information concerning our everyday life change, so it is imperative to allow the user change the personal ontology by letting her relate "File" instances to other instances or concepts.

File relating could be implemented in two ways; either by right clicking the file and selecting the type and the target of the new relationship or by dragging and dropping a file on to a node shown in the graph visualization pane.

6.2.4.1 Relate File by Right Clicking

Starting with the first case, the user may right click on a file from the file list pane, select the “Add relation...” option and a new frame will appear. As shown in Figure 6-10, a label containing the name of the file selected to be related, is located on the top of the “Relate file to...” dialog. On its right, a combobox containing all possible types of relationships allows the user to select one of them to be added. Figure 6-11 shows all candidate relations for “OntoFM A Personal Ontology-based File Manager for the Desktop” file contained in the candidate relations combobox. First of all, the “is a” relation exists in case the user wants to relate the selected file to a concept. Secondly, “submittedIn”, “acceptedIn”, “authors” and “owner” relations exist due to the fact that the file is a “Demo_Paper” and inherits recursively all parental object properties. Finally, “related_thing” relation exists due to the fact that the selected item is a file.

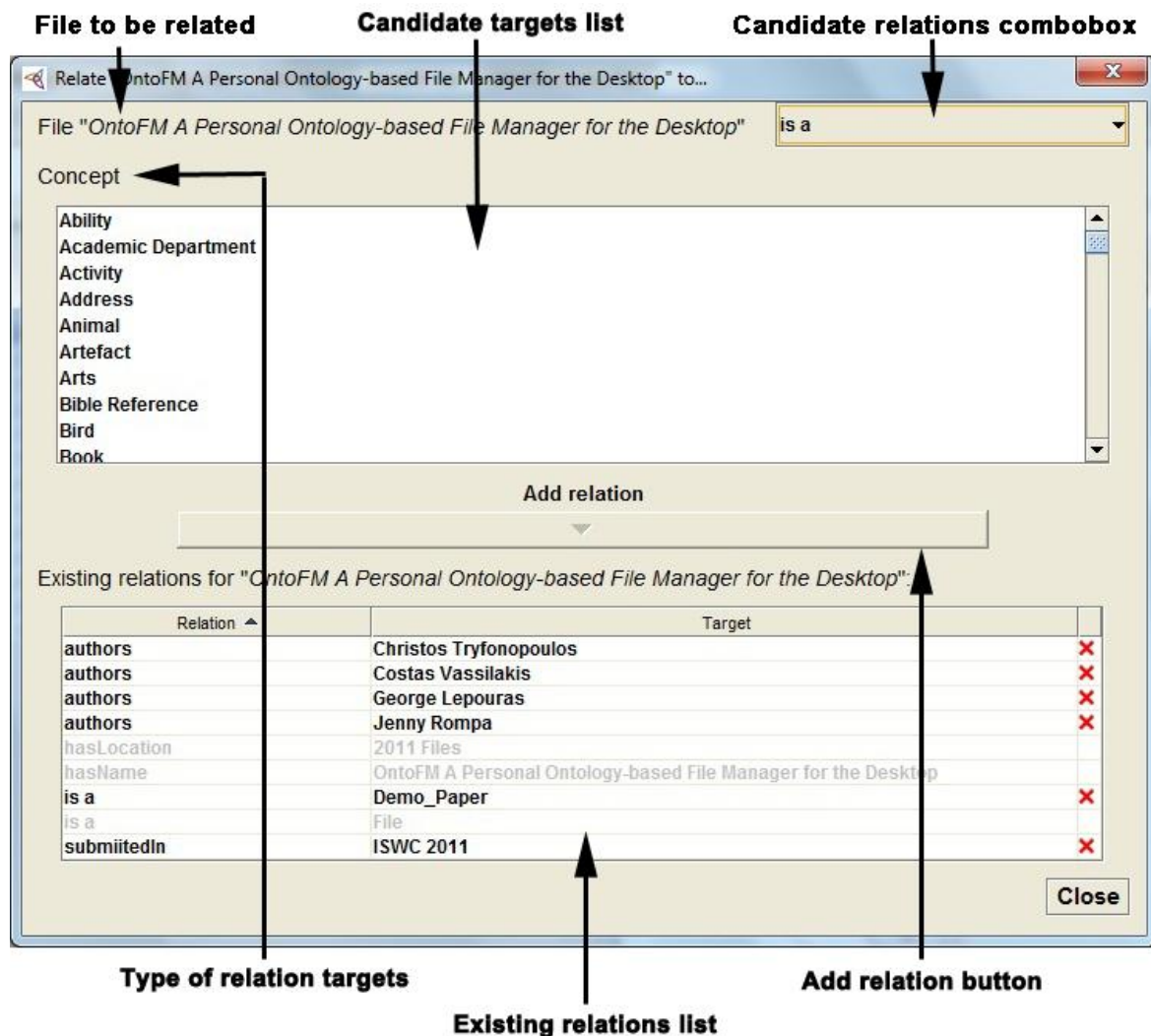


Figure 6-10. Relate by right clicking file to concept/instance dialog with captions

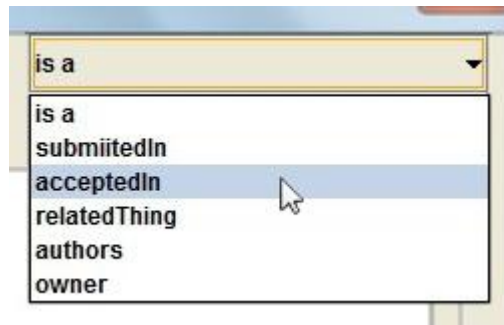


Figure 6-11. Candidate relations for "OntoFM A Personal Ontology-based File Manager for the Desktop" file

After selecting the type of relation to be added, the label naming the types of relation targets and the candidate targets list below it, get both updated. In Figure 6-10 the "is a" relation is selected and all possible "Concepts" are contained in the candidate targets list whilst in Figure 6-12 "acceptedIn" relation is selected and all possible "Conferences" and "Journals" are contained in the candidate targets list. As soon as the user selects a target from the corresponding list, the "Add relation" button gets enabled.

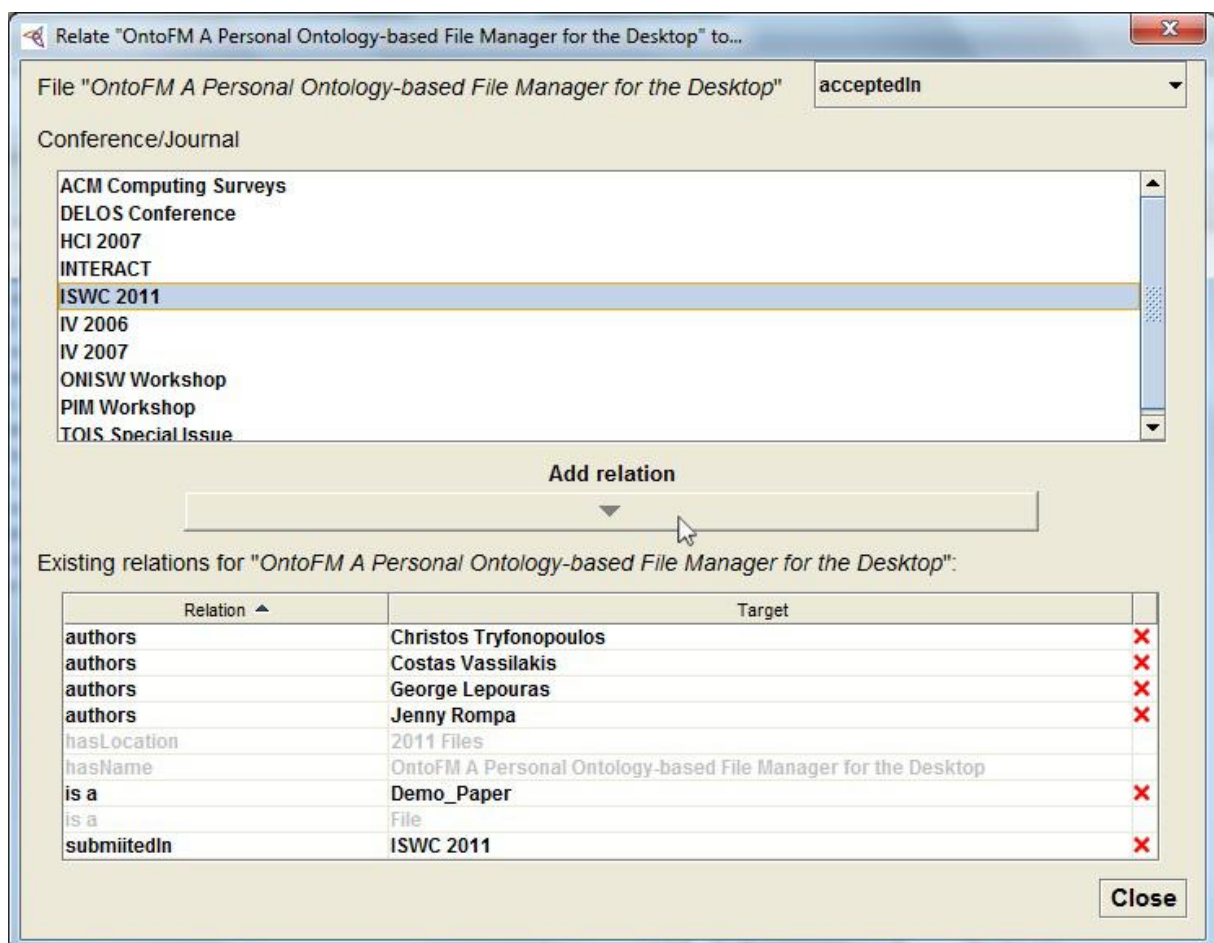


Figure 6-12. Creating new relation - "OntoFM A Personal Ontology-based File Manager for the Desktop" file "acceptedIn" "ISWC 2011" "Conference/Journal"

Below the “Add relation” button the existing relations list is located, containing all existing relations followed by their targets regarding the file selected to be related. When the user presses the “Add relation” button, the new selected relation between the file and the target is created in the personal ontology. Candidate targets list gets updated in order to remove the candidate target added. Likewise, the existing relations list creates a new row consisting of the type and the target of relation added and followed by a removal icon. As shown in Figure 6-12, not all of the existing relationships are followed by a removal icon, having the ones not followed grayed out. Such provision was made in order the user not to be able to delete crucial relationships such as “hasName”, “hasLocation”, or “is a” followed by “File”. As mentioned in 5.1.1 subsection, the personal ontology’s structure should not be dramatically changed preventing in this way to end up inappropriate to serve its purpose. This led to the characterization of these types of relationships as *System Relationships* and they should get updated automatically by the system.

In case the user wants to remove an existing relationship, she may click on the removal icon of the desired relationship. The selected relationship will be removed from the personal ontology and the existing relations list will be updated so that it does not contain the relationship anymore. If the relation removed matches the selected relation from the candidate relations combobox, candidate targets list will also get updated to contain the target removed.

When the user does not want any more changes, she may click on “Close” button, located at the bottom right corner.

6.2.4.2 Relate File by Dragging and Dropping

The other way of relating a file to either a concept or an instance is by dragging and dropping the file to the corresponding node. Relating a file to a concept differs from relating a file to an instance, as the only type of relationship that may exist between a “File” instance, or even better any instance, and a concept is the “is a” relationship. In case the user drags and drops a file on to a concept node, an information dialog appears informing the user whether the linking procedure was successful. However, in case the user drags and drops a file on to an instance node, as soon as she releases the mouse button, a new frame containing all possible relationships will appear. As shown in Figure

6-13, the dialog consists of an array containing all possible relations that could be assigned to the selected items. Each row consists of three fields; the first one is the source file, the second one is the candidate relation and the last one the target on which the file was dropped on to. The user may either select the relation she wants to be added and then click the "Add" button or click the "Cancel" button and close the dialog.



Figure 6-13. Relate by drag and drop file to concept/instance dialog

This way of adding a relationship is easier, faster and more direct. However, it is useful only for adding relationships and not for removing the existing ones. To this end, an evaluation is proposed to determine which method is mostly preferred by the users.

7

FUTURE WORK

PREVIEW

- Suggestions for future work

This final chapter gives some suggestions for the future work.

The goal of this thesis was to implement a personal ontology-based file manager, offering a mind map-oriented visualization to support user navigation within the personal information base. The OntoFM plug-in implemented, while functional, is still being developed.

We are planning to extend the functionality as to populate automatically the personal ontology by extracting information from documents, images, e-mails, contacts, multimedia and so forth. In this way not only we save the user a large workload on matching information items with the personal ontology but also we ensure that mapping is implemented in a proper manner keeping the personal ontology appropriate to serve its purpose and retrieve the expected results when a search is performed.

The way search results are displayed is still being surveyed and various options will be tried to prevent cluttering. Additionally, in the current development stage all concepts are treated with equal weight. However, if user activity is monitored, some concept instances may be assigned higher weight values than others; a prominent way of setting concept instance weights is described in (Katifori, Vassilakis, & Dix, 2008). For a densely populated personal ontology this may help users select concepts of importance.

Tasks like creating a folder or cut or copy a file and then paste it to another catalogue are not supported by this version of the OntoFM plug-in. However, it would be useful if the user could execute such tasks directly from the OntoFM interface. Apart from file system management, tools concerning ontology management should be offered too, allowing the user to create fast and easy new concepts by using graphic tools.

Finally, we intend to start a long term evaluation study of the OntoFM file manager to reveal strong and weak points, to find new ways to improve its functionality and better understand user filing and recovering patterns in a semantically enhanced environment. In the near future we plan to start an evaluation study on how the users prefer to relate files to other ontology items.

REFERENCES

- Alani, H. (2003). TGVizTab: An Ontology Visualisation Extension for Protégé. *Proceedings of Knowledge Capture (K-Cap'03), Workshop on Visualization Information in Knowledge Engineering*. Sanibel Island, Florida.
- Barreau, D. (1995). Context as a factor in personal information management systems. *Journal of the American Society for Information Science, Volume 46, Issue 5*.
- Boardman, R. (2004). *Improving Tool Support for Personal Information Management*. London: Doctoral dissertation, Imperial College.
- Buzan, T. (1991). *The Mind Map Book*. New York: Penguin.
- Cai, Y., Dong, X., Halevy, A., Liu, J., & Madhavan, J. (2005). Personal Information Management with Semex. In *SIGMOD '05: Proceedings of the 2005 ACM SIGMOD international conference on Management of data* (pp. 921 - 923). New York, NY, USA: ACM Press.
- Davies, M. (2011). Concept mapping, mind mapping and argument mapping: what are the differences and do they matter? In *Higher Education, Volume 62, Issue 3* (pp. 279-301).
- Dong, X., Halevy, A., Nemes, E., Sigurdsson, S., & Domingos, P. (2004). Semex: Toward on-the-fly personal information integration. In *Workshop on Information Integration on the Web (IIWEB)*.
- Eclipse IDE for Java Developers*. (2001, November). Retrieved 02 14, 2012, from <http://www.eclipse.org/>
- Eppler, M. (2006). A comparison between concept maps, mind maps, conceptual diagrams, and visual metaphors as complementary tools for knowledge construction and sharing. In *Information Visualization, Volume 5, No. 3* (pp. 202-210).
- Fernandez-Garcia, N., Sauermann, L., Sanchez, L., & Bernardi, A. (2006). PIMO Population and Semantic Annotation for the Gnowsis Semantic Desktop. *Proceedings of the Semantic Desktop and Social Semantic Collaboration, Volume 202 of CEUR-WS*.
- Gennari, J., Musen, M., Ferguson, R., Grosso, W., Crubézy, M., Eriksson, H., et al. (2003). The Evolution of Protégé: An Environment for Knowledge-Based Systems Development. In *International Journal of Human-Computer Studies, Volume 58, Issue 1* (pp. 89-123). Duluth, MN, USA: Academic Press, Inc.
- Giannopoulos, G., Bikakis, N., Dalamagas, T., & Sellis, T. (2010). GoNTogle: A Tool for Semantic Annotation and Search. *ESWC*, (pp. 376-380).
- Gifford, D., Jouvelot, P., Sheldon, M., & O'Toole, J. (1991). Semantic File Systems. *Proceedings of the 13th ACM Symposium on Operating Systems Principles, Volume 5*, 16-25.

- Golemati, M., Katifori, A., Vassilakis, C., Lepouras, G., & Halatsis, C. (2007). Creating an Ontology for the User Profile: Method and Applications. *Proceedings of the First IEEE International Conference on Research Challenges in Information Science (RCIS)*. Morocco.
- Graphviz*. (2004). Retrieved February 4, 2012, from <http://www.graphviz.org/>
- Haystack*. (2004). Retrieved January 31, 2012, from <http://simile.mit.edu/hayloft/>
- Henderson, S. (2009). Guidelines for the Design of Personal Document Management User Interfaces. *Personal Information Management 2009 Workshop*. Vancouver.
- Jambalaya*. (n.d.). Retrieved July 6, 2011, from <http://protegewiki.stanford.edu/wiki/Jambalaya>
- Jones, W. (2008). *Keeping found things found: The Study and Practice of Personal Information Management*. Burlington, MA: Morgan Kaufmann Publishers.
- Jones, W., & Teevan, J. (2007). *Personal Information Management*. University of Washington Press.
- Jones, W., Bruce, H., & Dumais, S. (2001). Keeping found things found on the web. *Proceedings of the tenth international conference on Information and knowledge management*, pp. 119–126.
- Katifori, A., Halatsis, C., Lepouras, G., Vassilakis, C., & Giannopoulou, E. (2007, October). Ontology Visualization Methods - A Survey. *ACM Computing Surveys, Volume 39, Issue 4, Article No.: 10*, p. 43.
- Katifori, A., Poggi, A., Scannapieco, M., Catarci, T., & Ioannidis, Y. (2005, June). OntoPIM: How to Rely on a Personal Ontology for Personal Information Management. *ISWC Workshop on Semantic Desktop*, pp. 78-81.
- Katifori, A., Torou, E., Halatsis, C., Lepouras, G., & Vassilakis, C. (2008). Selected results of a comparative study of four ontology visualization methods for information retrieval tasks. *Proceedings of the 2nd International Conference on Research Challenges in Information Science, RCIS 2008*, (pp. 133-140).
- Katifori, A., Vassilakis, C., & Dix, A. (2008). Using Spreading Activation through Ontologies to Support Personal Information Management. *International Conference on Intelligent User Interfaces IUI*.
- Katifori, A., Vassilakis, C., Daradimos, I., Lepouras, G., Ioannidis, Y., Dix, A., et al. (2008). Personal Ontology Creation and Visualization for a Personal Interaction Management System. *CHI*.
- Kljun, M., Dix, A., & Solina, F. (November 2009). *A Study of a Crosstool Information Usage on Personal Computers: how users mentally link information relating to a task but residing in different applications and how importance and type of acquisition affect this*.
- Lepouras, G., Dix, A., Katifori, A., Catarci, T., Habegger, B., Poggi, A., et al. (2006). OntoPIM: From Personal Information Management to Task Information Management. *Personal Information Management ACM SIGIR 2006 Workshop*. Seattle, Washington.
- Lucene*. (1997). Retrieved 2 14, 2012, from <http://lucene.apache.org/core/>

Mahalingam, M., Tang, C., & Xu, Z. (2003). Towards a Semantic, Deep Archival File System. *Proceedings of the The Ninth IEEE Workshop on Future Trends of Distributed Computing Systems*, pp. 115-121.

NavigOwl. (n.d.). Retrieved February 2, 2012, from <http://klatif.seecs.nust.edu.pk/navigowl/>

NEPOMUK. (n.d.). Retrieved February 2, 2012, from <http://nepomuk.semanticdesktop.org>

Noy, N. F., Fergerson, R., & Musen, M. (2000). The Knowledge Model of Protégé-2000: Combining Interoperability and Flexibility. In *Proceedings of the 12th European Workshop on Knowledge Acquisition, Modeling and Management* (pp. 17-32). London, UK: Springer-Verlag.

OntoGraf. (n.d.). Retrieved July 6, 2011, from <http://protegewiki.stanford.edu/wiki/OntoGraf>

OWL. (2005). Retrieved March 5, 2012, from <http://www.w3.org/TR/owl-features/>

OWL API. (2004). Retrieved 2 14, 2012, from <http://owlapi.sourceforge.net/>

OWL Viz. (n.d.). Retrieved February 4, 2012, from <http://protegewiki.stanford.edu/wiki/OWL Viz>

Pizza ontology. (n.d.). Retrieved February 4, 2012, from <http://www.code.org/ontologies/pizza/2007/02/12/>

Protégé. (n.d.). Retrieved July 6, 2011, from <http://protege.stanford.edu/>

Protégé bug report. (2011, June 13). Retrieved June 28, 2011, from https://bmir-gforge.stanford.edu/gf/project/owleditor/tracker/?action=TrackerItemEdit&tracker_item_id=3324&start=0

Protégé OWL API. (n.d.). Retrieved 2 14, 2012, from <http://protege.stanford.edu/plugins/owl/api/>

Protégé team. (n.d.). Retrieved July 6, 2011, from <http://protege-ontology-editor-knowledge-acquisition-system.136.n4.nabble.com/>

Sauermann, L., & Heim, D. (2008). Evaluating Long-Term Use of the Gnowsis Semantic Desktop for PIM. *Proceedings of the 7th International Conference on The Semantic Web* (pp. 467 - 482). Karlsruhe, Germany: Springer-Verlag.

Semex. (n.d.). Retrieved January 31, 2012, from <http://db.cs.washington.edu/semex/>

Sesame. (1999). Retrieved June 28, 2011, from <http://www.openrdf.org/>

SPONGE. (2008). Retrieved February 4, 2012, from <http://imu.ntua.gr/?q=node/201>