

# Ontology Driven Graphical User Interface Development

## DISSERTATION

submitted to the  
Graz University of Technology,  
Faculty of Computer Science,  
for the attainment of the degree of  
Doctor of Engineering Sciences (Dr. techn.)

by

**Syed Khuram Shahzad**

Institute for Knowledge Management (IWM)  
Graz University of Technology



Graz University of Technology

First Assessor and Advisor: **Assoc.Prof. Dipl.-Ing. Dr.techn. Denis HELIC**

Second Assessor: **Univ. -Prof. Dr. Michael GRANITZER**

Graz, 7 February, 2012



# Ontologie basierte Benutzerschnittstellen Entwicklung

## DISSERTATION

vorgelegt an der  
Technische Universität Graz,  
Fakultät für Informatik,  
zur Erlangung des akademischen Grades  
Doktor der Technischen Wissenschaften (Dr.techn.)

by

**Syed Khuram Shahzad**

Institut für Wissensmanagement  
Technische Universität Graz



Betreuer & Begutachter 1: **Assoc.Prof. Dipl.-Ing. Dr.techn. Denis HELIC**  
Begutachter 2: **Univ. -Prof. Dr. Michael GRANITZER**

Graz, 7. February, 2012



## *Statutory Declaration*

I declare that I have authored this thesis independently, that I have not used other than the declared sources/resources, and that I have explicitly marked all material which has been quoted either literally or by content from the used sources.

Ich erkläre an Eides statt, dass ich die vorliegende Arbeit selbstständig verfasst, andere als die angegebenen Quellen/Hilfsmittel nicht benutzt, und die den benutzten Quellen wörtlich und inhaltlich entnommene Stellen als solche kenntlich gemacht habe.

-----  
Place, Date

-----  
Syed Khuram Shahzad



# Abstract

Any computer application is a computational model of real world concepts that can be conceived by the computer user through its representation at the user interface level. These real world concepts have been formalized through semantic and ontological frameworks in the shape of many knowledgebases. Such ontological frameworks have been used with increasing success in last decade in the field of knowledge management and especially for web applications. Such semantic formalizations ensures consistent and standardized concept delivery of intended domain concepts to the computer application users independent of its technical properties. Semantic formalizations are mainly used in business logic and on the database level. Later on semantics are added using translation of semantic data to imperative user interfaces. These translations are made through some added plug-ins and additional layers between the business logic and user interfaces. Semantic and ontological frameworks already provide complete knowledge on domain properties, their relationships and constraints. However, for the development of the user interfaces knowledge is not currently fully used by technologies. Hence, there is a need to carry semantic rules directly to the user interfaces during the design and development process. In this thesis, the area of semantic and ontological frameworks driven user interface development is explored to develop and provide transparent interfaces to the end user.

This PhD thesis highlights the significance of ontological modeling concepts for knowledge representation which are used for graphical user interface development. This research work is to investigate rules and user interface ontologies that preserve the domain semantics provided by Domain Ontologies (DO) and domain-specific constraints during user interface development. Moreover, our research provides the opportunity for maintaining consistency among user interfaces independent to the technology used for GUI development. We have adapted a well-known user interface layer models to build an ontological user interface base model. The base layer is fetched from the domain ontology structure that provides the semantic classification and architecture of information at the user interface level. Next layer states the user interaction techniques and flow. The third layer of the user interface model specifies the graphical properties for the look and feel of the user interface. These multiple layers also provide some context aware properties. These properties are dependent to the technology and user profile and user role. This User Interface Model (UIM) is quantified and instantiated to develop a Graphical User Interface. Our research has three major incremental steps:

- At the initial stage we adopted functional programming approach using relational algebra and Haskell implementation for DO and User interface property mapping. Higher order functions to map these concepts to application user interface. This experiment builds a direct mapping from DO to graphical user interface.

- In the second stage, a base model for user interfaces has been introduced by adding user interface properties to DO. In an imperative programming approach the base model has been used in the context of personal information management.
- At the final stage we came up with a User Interface Ontology (UIO) providing User Interface properties and their semantic relationships. Mapping of DO with UIO properties provide a User Interface Mapping (UIM) that is instantiated to develop a GUI.

Being a novel idea, semantics and ontological frameworks driven GUI development opens many new horizons to be explored in this regard. One major trend following this research is to explore usability testing for these user interfaces. Moreover, functional ontologies may be extended to join the user semantic user interaction methodology and semantic functions.





# Kurzfassung

Jede Anwendung repräsentiert Berechnungen über Konzepte der realen Welt, deren Ergebnisse über meist grafische Benutzerschnittstellen konsumiert werden. Semantische und ontologische Technologien bieten dabei einen Rahmen zur Formalisierung solcher Konzepte und wurden im letzten Jahrzehnt vermehrt im Bereich des Wissensmanagements und für Web-Anwendungen eingesetzt. Solche semantische Formalisierungen ermöglichen die konsistente und standardisierte Bereitstellung von Domänen spezifischen Konzepten mittels Anwendungen unabhängig von technischen Eigenschaften der verwendeten Systeme. Im Allgemeinen erfolgt der Einsatz semantischer Formalisierungen auf Ebene der Business-Logik und Datenbanken. Darauf aufbauend erfolgt die Übersetzung der Domänen Semantik in Benutzerschnittstellen unter Berücksichtigung Schnittstellen spezifischer Eigenschaften. Diese Übersetzungen werden durch proprietäre Plug-Ins und zusätzliche Schichten zwischen Business-Logik und User-Interface erstellt. Die ontologische Formalisierung bietet aber bereits vollständiges Wissen über Domänen-Eigenschaften. Wissen, welches derzeit von Technologien zur Erstellung von Benutzerschnittstellen nicht berücksichtigt wird. Daraus leitet sich der Bedarf ab, Domänen Semantik zur Entwicklung und Gestaltung von Bedienoberflächen zu verwenden.

Die vorliegende Dissertation erforscht daher die Bedeutung ontologischen Modellen zur Unterstützung des Entwicklungsprozesses von grafischen Benutzeroberflächen. Die Forschungsarbeit untersucht Regeln und Ontologien für Benutzerschnittstellen welche die zugrunde liegenden Domänen Semantik während der Entwicklung von Benutzerschnittstellen erhält. Darüber hinaus bietet meine Forschung die Möglichkeit, konsistente Benutzerschnittstellen unabhängig von der für die GUI-Entwicklung verwendeten Technologie zu erstellen. Durch Adaption bekannter Schichtenmodelle für Benutzerschnittstellen konnte ein ontologisches Modell für Benutzerschnittstellen entwickelt werden. Die Basisschicht besteht aus einer Ontologie welche die Klassifikation und Architektur der Informationen, die auf Benutzeroberfläche angezeigt werden, abbildet. Die darauf aufbauende Schicht behandelt Benutzer-Interaktionen und Workflows. Die dritte Schicht des Modells spezifiziert die grafischen Eigenschaften (e.g. Look-and-Feel). Dieses Schichtenmodell bietet auch einige kontextbezogene Eigenschaften welche unabhängig von Technologie, Benutzerprofil und Benutzer Rolle sind. Das User Interface Model (UIM) wird instanziiert, um eine Benutzerschnittstelle zu entwickeln. Die Forschung wurde dabei in drei Schritten durchgeführt:

- In der Anfangsphase erfolgte die Umsetzung der Domänen-Ontologie (DO) und Benutzerschnittstelleneigenschaften mittels funktionaler Programmierung und einer in Haskell abgebildeten relationalen Algebra. Funktionen höherer Ordnung definieren dabei die Zuordnung von Konzepten zur Benutzeroberfläche. Dieses Experiment erstellt eine direkte Zuordnung der DO zu bedienende grafische Benutzeroberfläche.

- In der zweiten Phase, erfolgte die Erstellung eines ontologischen Basis-Modells n zur Abbildung Benutzeroberflächen spezifischer Eigenschaften. In einem imperativen Ansatz erfolgte die Anwendung dieses Modells im Bereich Personal Information Management.
- In der letzten Phase erfolgte die Entwicklung der finalen User Interface Ontologie (UIO) zur Abbildung von User Interface Eigenschaften und deren semantischen Beziehung. Dies beinhaltet auch die Abbildung der DO mit der UIO. Das daraus entstehende UIM ermöglicht die automatische Instanziierung einer Benutzerschnittstelle.

Als eine neue Idee öffnen semantische und ontologische Formalisierungen neue Horizonte in der Entwicklung graphischer Benutzerschnittstellen. Ein wichtiger Trend besteht in der Automatisierung von Usability-Tests für Benutzeroberflächen. Darüber hinaus weist die Arbeit in Richtung funktionaler Ontologie welche Interaktionen mit dem Benutzer in den bestehenden Ansatz integrieren.



*To my loving parents and family.  
Their support, insight, guidance, patience and kind wishes always,  
craft the success for me!*



# Acknowledgements

First of all, I would like to thank the Almighty Allah, for His divine guidance and providence. His support, blessings, goodness, and kindness were always with me. He blessed me with motivation, passion, and hard work. It was his blessings which made me able to plan, visualize, and execute my dreams into the reality. I would like to dedicate this achievement to Him and my inspirational father Syed Mashhood Ali and my loving mother Sabira Mashhood, whose prayers, motivations and belief in me got me that far. They are always a source of inspiration and driving force behind my studies. I would also like to share my achievements and joy with my brothers, my sisters and their families. I find no more words to thank them for their continuous support.

Every PhD student wishes to conduct research under a visionary and inspirational supervisor, as well as with a researcher who is intelligent and motivating. I am very lucky to find this ideal supervisor in form of Dr. Denis Helic. I am really thankful to him for giving me the exceptional opportunity of doing a PhD with him. His constant encouragement, help, and invaluable supervision helped me to groom my educational, research and writing capabilities which further excelled me to broaden my vision and brought best out of me for this research work. I owe my deepest gratitude to Prof. Dr. Michael Granitzer for being part of my dissertation evaluation committee and accepting the role of second reader despite his busy schedule. The discussions/comments of Prof. Dr. Michael Granitzer were very useful for my research and thesis. I am very obliged to Prof. Dr. Klaus Tochtermann who provided me the opportunity to start my PhD work at prestigious Knowledge Management Institute. I also want to thank Prof. Dr. Andrew U. Frank who has introduced research work and methodologies and many new research areas. His kind and expert reviews were always very helpful for at beginning of my research phases.

Furthermore, my dearest thanks go to my dear friends and colleague Dr. Tanvir Afzal and Dr. Atif Latif who indeed acted as a mentor to me. Their kind guidance and motivations always bring me out of the difficult patches throughout my research studies. I have learnt a lot from their company and discussions, which eventually led me define a path for my PhD writing and publication during my PhD work. The time we together passed in discussion, sports, travelling and leisurely, with no doubts is a precious and golden period of my life. Many thanks to my colleagues at the Knowledge Management Institute and Know Center, whose help, support and guidance supported me to achieve the success of my PhD studies. I want to thank for the development and testing teams specifically Mr. Nauman Jameel from Systematic Bytes Inc. and Mr. Jawad-ul-Husnain from Expertflow Pvt. Ltd for their kind efforts and cooperation for development and testing of research methodologies. I also want to thank here Mr. Farhan Hyder who helped me in my research work, writings, fruitful discussions, in introduction of new technologies and specially proof reading my thesis. I was lucky enough to find myself among very loving and caring family of Pakistani scholars in Graz. They

always fill in the gap of a family and made me happy while I was sad. They continuously motivated me to achieve my goals. Thanks to all who arranged and accompanied me in doing academics and non-academic activities in form of sports, parties and travelling. Especially, my gratitude goes out to Dr. Anwar us Saeed, Dr. Muhammad Umer Saleem, Dr. Javed Ferzund, Dr. Tanveer Iqbal and Mr. Mudassir Abbas. I am unable to mention names of all scholars yet all of you are special to me. At the end my heartiest tribute to my country Pakistan and Higher Education Commission (HEC), Government of Pakistan, for partially funding my research as without their assistance I would not have a chance to pursue my PhD and full fill my dreams. In the same spirit my tribute goes to the Austrian Agency for International Cooperation in Education and Research (OeAD-GmbH), Know Center, Knowledge Management Institute and Technology University of Graz for the moral and financial support to attend the conferences around the world.

Syed Khuram Shahzad

Graz, Austria, March 2012



# Contents

Introduction.....	1
1.1. Motivation.....	2
1.1.1. Why Ontology.....	3
1.2. Problem formalization .....	4
1.2.1. Research Questions .....	5
1.3. Scope and research focal point.....	5
1.4. Scientific experiments and contributions .....	6
1.5. Thesis Organization .....	8
Semantic and Ontological Framewor .....	10
2.1. Ontology .....	10
2.1.1. Definitions.....	10
2.1.2. History.....	11
2.1.3. Metaphysics.....	14
2.2. Semantics and Ontological framework .....	14
2.2.1. Semantics .....	15
2.2.2. Domain Ontologies and upper Ontologies .....	15
2.2.3. Ontology components .....	16
2.2.4. Vocabulary .....	18
2.2.5. Taxonomy and Meretopology .....	18
2.2.6. Universe of Discourse .....	19
2.2.7. Descriptive, Formal and Formalized Ontologies .....	19
2.3. Ontologies for Knowledge Representation and Sharing .....	20
2.3.1. Epistemology and Knowledge.....	20
2.3.2. Knowledge management .....	20
2.4. Computer Applications of Ontology.....	23

2.4.1.	Ontological Concept Modeling .....	24
2.4.2.	Ontology Engineering .....	24
2.4.3.	Cognitive Science and AI.....	24
2.4.4.	Ontology Languages .....	25
2.5.	RDF /OWL .....	26
2.5.1.	Serialization formats.....	26
2.5.2.	Resource identification.....	27
2.5.3.	Statement reification and context .....	27
2.5.4.	Query and inference languages .....	27
2.5.5.	RDFS.....	28
2.5.6.	OWL .....	29
2.6.	Semantic Web.....	29
2.6.1.	Standards.....	30
2.6.2.	Projects.....	31
2.6.3.	Semantic Databases.....	32
2.6.4.	Semantic web data spaces, linked data, and data portability .....	33
2.7.	Ontology Driven Information System.....	35
	Graphical User Interfaces.....	36
3.1.	Introduction.....	36
3.1.1.	Human-Computer Interaction.....	36
3.2.	History .....	36
3.2.1.	Precursors.....	36
3.2.2.	PARC user interface.....	39
3.3.	Structural elements.....	40
3.3.1.	Window.....	41
3.3.2.	Tabs.....	42
3.3.3.	Menus.....	42
3.3.4.	Icons.....	42

3.3.5.	Controls (or Widgets).....	43
3.4.	Interaction elements .....	46
3.4.1.	Cursor.....	46
3.4.2.	Pointer .....	46
3.4.3.	Selection.....	46
3.5.	Post-WIMP GUI .....	47
3.6.	User Interface Iceberg Analogy .....	48
3.7.	Element of User Experience .....	49
3.8.	User Interface Design .....	50
3.8.1.	Research – past and ongoing.....	50
3.8.2.	User Centered Interface design .....	52
3.8.3.	User Interface Designs for Web .....	54
3.8.4.	User Interface Design Processes .....	55
3.8.5.	User Interface Design Requirements.....	57
3.8.6.	Prototyping.....	58
3.9.	Usability.....	58
3.10.	User Interface Modeling.....	58
3.10.1.	Modeling Languages .....	59
3.11.	Semantics aware Interfaces .....	60
3.11.1.	Data Formats and Semantic Classification .....	60
3.11.2.	Input / Output Data Validation .....	60
3.12.	Intelligent User Interface.....	61
3.13.	User Interface Development.....	61
Functional Programming Approach .....		62
4.1.	Introduction.....	62
4.2.	Ontology formalization .....	64
4.2.1.	Formal Ontology .....	64
4.2.2.	Relational Algebra.....	65

4.2.1.	GUI States .....	66
4.3.	User Model by Haskell programming .....	67
4.3.1.	Why Haskell.....	67
4.3.2.	Representing Algebra in Haskell.....	68
4.3.3.	Manipulating States (State Monads).....	69
4.3.4.	Data types and Typed Classes .....	70
4.3.5.	User Model.....	70
4.4.	User Model in Haskell .....	72
4.4.1.	Graph.....	72
4.4.2.	Node .....	72
4.4.3.	Edge .....	72
4.5.	Mapping User Model to GUI.....	73
4.5.1.	GUI construction tools .....	73
4.5.2.	Drawing at Canvas .....	74
4.5.3.	GUI .....	74
4.5.4.	Limitations and constraints.....	76
4.6.	Results and Conclusion.....	77
4.7.	Future work.....	77
	Ontology based User Interface Development: User Experience Elements Pattern.....	79
5.1.	Introduction.....	79
5.2.	User Experience Elements .....	81
5.2.1.	Conceptualization through Ontological Modelling .....	83
5.2.2.	Personal Information Management (vCard/hCard) .....	83
5.3.	Ontological Framework for User Interface Development.....	85
5.3.1.	Ontology Parser.....	85
5.3.2.	User Interface Properties Mapping:.....	87
5.3.3.	GUI Development .....	89
5.4.	Customized User Interface Control.....	90

5.5.	User Interface Generation .....	90
5.6.	Ontology Modelling based on Context .....	91
5.7.	Results.....	91
5.8.	Conclusion .....	92
Ontological Model Driven GUI Development: User Interface Ontology Approach .....		93
6.1.	Introduction.....	93
6.1.1.	Motivation .....	95
6.2.	Ontology Engineering for User Interface Ontology (UIO) .....	95
6.2.1.	Modeling User Interface Aspects .....	95
6.2.2.	Ontology Engineering for User Interfaces.....	96
6.3.	Mapping Domain Ontology with UIO .....	98
6.3.1.	Mapping Visualization Classes .....	99
6.3.2.	Mapping User Interface Properties.....	99
6.3.3.	Mapping Graphical Properties.....	99
6.3.4.	Mapping Context aware properties.....	99
6.3.5.	User InterfaceModel (UIM) .....	99
6.3.6.	vCard Ontology for Personal Information Management .....	100
6.4.	User Interface generation .....	100
6.4.1.	Quantifying Context aware properties.....	100
6.4.2.	Instantiating UIM .....	101
6.5.	Software Engineering Aspects fo UIO.....	101
6.6.	Results.....	102
6.7.	Conclusion .....	102
6.8.	Future Work.....	103
Results and Conclusion .....		104
7.1.	Introduction.....	104
7.2.	Functional programming approach .....	105
7.3.	Impreative Programming Approach.....	105

7.3.1. Experiment Structure.....	105
7.4. User Interface Ontology.....	107
7.4.1. Ontology Engineering .....	107
7.4.2. GUI Generation .....	108
7.5. Conclusion and Future Work .....	109
7.5.1. Research Targets .....	109
7.5.2. Experiment Results outcomes .....	109
7.5.3. Future Work .....	110
Bibliography.....	111

# List of Figures

Figure 1.1: Thesis Architecture .....	8
Figure 2.1: Table of Contemporary Ontologists.....	12
Figure 2.2: Example Taxonomy of Vehicles.....	18
Figure 2.3 The Semantic Web Stack .....	30
Figure 2.4: Class linkages within the Linking Open Data datasets October 2008.....	34
Figure 2.5: Class linkages within the Linking Open Data datasets September 2011.....	34
Figure 3.1: IBM 029 card punch .....	38
Figure 3.2: Screenshot MS Dos.....	38
Figure 3.3: Xerox Star Workstation .....	39
Figure 3.4: An Example of GUI structure for Printer Properties for MS Office .....	44
Figure 3.5: The iceberg analogy of usability by Dick Berry .....	48
Figure 3.6: Element of user Experience by David Garret (Garrett, 2002).....	49
Figure 3.7: A Browser User Interface United States Patent .....	54
Figure 4.1: Chapter 4 organization .....	63
Figure 4.2: An Example representation of Graph.....	65
Figure 4.3: Node Number as a Functional Types to represent ontology .....	74
Figure 4.4: Panel and canvas of the graph properties through typed controls .....	75
Figure 5.1: Chapter 5 organization.....	80
Figure 5.2: Adapted User Experience Elements by Garrett.....	81
Figure 5.3: Structure of vCard classes.....	84
Figure 5.4: vCard Object Properties .....	84
Figure 5.5: vCard Data Properties .....	84
Figure 5.6: User Interface Ontology for vCard (Skeleton) .....	88
Figure 5.8: Example UI for a vCard.....	89
Figure 5.7: Ontology representation at GUI.....	90

Figure 6.1: Chapter 6 Organization ..... 94

Figure 6.2: Association: Relating Domain ontology with User Interface Ontology ..... 96

Figure 6.3: Portion of mapping application screenshot ..... 98

Figure 6.3: Knowledge Experts contribution breakdown and collaboration stages ..... 101

Figure 7.1: Major steps for User Interface Design and Development ..... 106

Figure 7.2: User Interface Model Layers..... 106



# List of Tables

Table 4.1: Ontology Description of a Graph .....	65
Table 4.2: The mapping of User Model to GUI through Data Flow (Haskell programming) .....	71
Table 5.1: Adapted User Experience Elements by Garrett (Garrett, 2002) .....	82
Table 5.2: Structure of vCard properties .....	87



# Introduction

In the decade of the 80's, many of the researchers lead by Smith (Smith, 1978) (Smith, 2004), Gruber (Gruber, 1995) and Guarino (Guarino, 1995) carried out the idea of semantic and ontologies from philosophy and cognitive science to formal method of conceptualization in computational models. They and many others used the idea in a way to the field of computer science to make up an ontologically formalized computational model of any real world's domain concept in interdisciplinary fields (e.g. bioinformatics (Smith, 2008), geoinformation, eBusiness and social cognition (Osterwalder, et al., 2002)). Semantically formalized knowledge helped in machine learning and understanding of concepts which brought up new fields (Wand, et al., 1990) of Artificial Intelligence, Knowledge Management and representation, Semantic Web and semantic databases. Moreover the huge list of ontologies<sup>1</sup> and many semantic schemas bring into sights new fields like explore standardization and alignments ontologies.

While research in other interdisciplinary fields were focused on better and standardized ontological representation of concepts (semantic cloud), Information science (computer science) research were targeting better techniques and methodologies for information modeling, storage, retrieval and visualization by exploiting the semantics and came up with Semantic Web and semantic databases.

By witnessing of ever-increasing research on semantic and ontological framework and its importance in Artificial Intelligence, Computational Linguistics, and Database Theory, Nicola Guarino tossed the idea of Ontology Driven Information System (ODIS) in FOIS'98 (Guarino, 1998). He detailed ODIS in a discussion of using ontologies at run time or development time for different computer application components. There were attempts to use ontological conceptualization of objects to improve Object Oriented Models for any Information System (IS). Rather than using ontologies and semantics additional layers to enhance conceptualization of any computational model, Guarino pointed out the need of information system modeling and development based on ontologies and semantic frameworks. This concept was adopted in different field of computer application development especially in semantic web. Uschold also talked about the same idea to improve conceptual modeling in software engineering field by joining model driven software development and ODIS. He also pointed out the need for the use of ontological models for development of any software component. This idea has been implemented in semantic web and semantic databases for semantic structuring of information.

There are now many semantic and ontological standards for different domain concept representation only Linked-data provides more than 30 trillion triples. That much variety of domains, classes and

---

<sup>1</sup> [http://protegewiki.stanford.edu/wiki/Protege\\_Ontology\\_Library](http://protegewiki.stanford.edu/wiki/Protege_Ontology_Library)

structures in provided shared knowledge bases have semantics and ontological knowledge representation in common. This framework defines standards for domain concept representation through the computational models. Currently, numerous Knowledge Bases (also known as semantic data storage) are available and can be retrieved through semantic query (SPAQL) mechanism. However, still there is a need to provide semantic user interface which can be validated through the reasoning software. Current knowledge visualization methodologies are based on conventional user interface development mechanism with additional layers and translation software (plug-ins or hookups) (Paulheim, 2009). There is a requirement to provide a semantic and ontological framework driven user interface development methodology that can ensure semantic consistency for information architecture and classification at developed user interface.

The conceptual modeling gap between user interfaces and other semantically driven software components has been explored during this research. The need to fill up the gap between user interface component and semantic and ontological framework based was considered much earlier in research by researchers (Cocchiarella, 1996) (Gruber, 1993) (Guarino, 1995) (Smith, 2004) (Wand, et al., 1990) (Uschold, 2008) (Paulheim, 2009). There are also guidelines and standards provided in the form of the technical reports, documentation and RFCs from open standards and ontologies provided at Linked Data or other knowledge bases e.g. vCard and vCalendar documentation provides “*User Interface (UI) Support Recommendations*”<sup>2</sup>. Next step explains the motives behind this PhD research thesis and next section provides problem formalization and boundary of target areas.

## 1.1. Motivation

The growing use of semantic and ontological framework in information sciences has upgraded modeling of any domain concept much abstract. This framework has become the base model to develop all components of computer applications, except user interfaces. User interfaces had much rapid development in devices and tools in last two decades. At lower level, many new user interaction devices are introduced with new interaction mechanism. At implementation level many data retrieval and visualization algorithms are implemented for fast and active user interface. All of these developments in user interfaces were achieved in technical fields. Some part of research in last thirty years reflected user concerns and profiling through mathematical modeling. With so much variety of devices and technologies, User Interface development has become too much technology dependent. The last change made at the conceptual level in user interface field was in 1981 by Xerox. Xerox introduced graphical user interface by introducing office environment concept for visual modeling (e.g. files, folder and documents) resulting in windows environment.

---

<sup>2</sup> <http://www.imc.org/pdi/pdiproddev.html>

There is big technology shift in other components of computer application at conceptual level from object oriented view to semantic and ontological representation of reality world concept in information system. This change has been introduced as Semantic Web (Web 3.0) information available here is represented, classified and stored according to semantic schema. The reason to use semantic and ontological representation of information is to avoid technology bias and limitations. The reason for making more generic patterns and classification is providing realistic knowledge representation. This aim can be achieved only if the formalized concept is delivered to user with semantic and ontological properties and constraints.

### 1.1.1. Why Ontology

Use of semantics and ontological framework as the base for GUI development is the main theme for this research. Arguments supporting Ontology Driven Information System (ODIS) are the base motivation. These arguments provide the benefits of adopting ODIS in comparison to other software engineering models. Uschold (Uschold, 2008) has provided these benefits as follows in his talk about past, present and future of the ODIS:

- Standardization and Consistency: All components and agents are uniformly designed and verified
- No Technology Bias: Information system modeling do not required any technology details or skills for specific programming language.
- Heterogeneous Knowledge Representation through Knowledge-Cloud: Large connect knowledgebase like Linked Open Data (LOD) provides different heterogeneous knowledgebase and respective semantic agents for data storage, representation and visualization.
- Reusable Semantic Agents: Knowledgebase provides their respective semantic agents for data representations, storage and visualization.
- Transactional Database: Semantic agents are stored in transactional databases with their version control systems
- No Code Generation and Model Translation to programming languages: Graphical environment for designing and development of application through linking and reusing pre-coded various kinds of semantic agents.
- Runtime Expandability: Any web interface can be extendable to a new semantic data set just my linkage to its respective semantic agent.

Two major motives for this research are outlined as:

- 1 First motive is the UI development process modeling providing user interface design rules which bind the UI designer and developer more than guidelines do. These proposed UI deigning guidelines are flexible and vary according to the domain represented at UI. With other design guidelines that domain ontology specifies many constraints and rules for information

classification and architecture that has to be presented at UI, e.g. a data classification and types at any concept visualization should be based on the semantic classification rather than any graphical library provided data types.

- 2 The second motive is to fill the gap between the conceptual designing of knowledge representation and its presentation at user interface. Knowledge representation schema is designed at very abstract level with formalized ontology, while its visualization designing is still technology dependent. There are some attempts to fill the gap by enabling these visualizations and user interfaces for handling and carrying these semantic and ontological rules defined at knowledge representation. These user interfaces contain additional qualities for translation of this ontological schema to conventional graphical library and device dependent user interfaces (Paulheim, 2009). This translation process bridges the semantic knowledge representation to the user interface for data visualization but cannot ensure the semantic properties and constraints delivery at user interfaces.

Smith (Smith, 2004) and Gruber (Gruber, 1993) states the requirement of domain concept delivery to the end user at user interface. Moreover it is also required to define design and development methodology for the semantically consistent user interface (Smith, 1989). Thus it provides the idea for improving software engineering is to increase the level of formal methods and structure in various phases of the software lifecycle. Using ontologies as the base models in model-driven software development represents the joining together of these two key ideas and the subject of the talk about ODIS (Uschold, 2008). A user interface designing and development methodology was, therefore, required that ensures consistent concept delivery from domain ontology. It should keep all the ontological properties and constraints defined in domain ontology and should not allow any technological (programming language or Graphical library or I/O device) bias to void any semantic or ontological rule stated by domain ontology.

## **1.2. Problem formalization**

Semantic and ontological framework has provided a base information architecture and classification for designing and development process for Semantic Web, Semantic data-storage and retrieval strategies except user interfaces. To achieve a semantically consistent concept delivery from these components to user interfaces, these user interfaces also need to be designed and developed on the basis of the domain concept. The exact issue discussed in this research is the design and development methodology for ontology driven GUI development for any domain ontology. To achieve consistency of information architecture and semantic at resulting user interface, a logically consistent methodology is required that doesn't allow the developers to void any semantic rules defined in domain ontology or semantic user interface model for the domain ontology.

The points to explore in research are the methodologies which can be developed for a user interface based on semantic and ontological framework. The two major questions about semantically consistency are 1. validation of the product (UI) and 2. process of UI design and development methodology through all components and product (e.g. UI components, base model for User Interface and completed GUI) at each phase (Sommerville, et al., 1994). The model based software engineering methodology has been adopted to ensure that UI and its design are completely based upon through a semantic and ontological UI model.

### 1.2.1. Research Questions

In this thesis four main research questions are addressed. The formulated main and sub research questions are given below.

- 1 How can semantically defined concepts be delivered at GUI?
- 2 Can Semantics and Ontological frameworks provide a based model for GUI?
  - a. How can a base model for UI contain and Domain Ontology relations and constraints?
- 3 How a base model for UI can be semantically consistent?
- 4 How a base model for UI can be instantiated to GUI ensuring that all IO Operation preserves semantic properties and allow data flow according to the semantic data classification using:
  - a. Information Structure (Mereotopology)
  - b. Semantic Classification (Taxonomy) including
    - i. Semantic data types
    - ii. Semantic data ranges and validation rules

## 1.3. Scope and research focal point

This research work mainly focuses on the use of semantics for graphical user interface development. The strategies to exploit the properties of semantic and ontological framework to present the domain concepts at user interface are discussed in this part of writing. Thus the focal point is the mapping of domain ontologies to the user interface either by bijective morphism or through a semantically defined user interface model carrying the domain ontology with user interface concerns.

We explore the methodology for designing and development of user interface as part of ontology driven information system development (Ontology Driven software engineering methodology).

The aim here is to preserve domain concept properties, laid down the semantic and ontological framework. Thus we focus not only logical consistency of the information but also provide a process that can be validated in contrast to raw, flexible and optional user interface design guidelines. We target the area of software engineering thus the user interface designers, HCI expert and user interface

developers are directly concerned to the research, while the end user is indirectly connected to the research.

The study strengthens the argument of using semantic and ontological framework with a history from ancient Greeks till today's revolutionary semantic web. Different subfields of study like Taxonomy, epistemology and mereotopology and their properties for defining a knowledge domain. These properties are the criteria to testify the consistency of information architecture and ontology concerns provided at user interfaces. As a novel approach to user interface development we have testify only the information and metadata consistency and completeness at user interface regarding domain concept delivery to end user. Detailed usability testing and consequent improvements to the methodologies are required to perform in next step and specified as future work.

## **1.4. Scientific experiments and contributions**

During the study, benchmarks of the study have been published as follows:

- 1 The first publication was done on the results from the experiments made on the developed research application for functional programming methodology discussed in 1.4. In these experiments an application to model a vector graph was constructed as an example of spatial ontology. The bibliographic details for the publication are:  
Shahzad, S., Granitzer, M., Tochtermann K.: Designing User Interfaces through Ontological User Model, Proceedings of the Fourth International Conference on Computer Sciences and Convergence Information Technology ICCIT 2009, Seoul, Korea, 24-26 November 2009. p 99-104.
- 2 Second publication was made using the imperative programming application results. These experiments are described in details in 1.5. Garrets model of user experience is used for structuring the user interface development process in four phases. In these experiments vCard ontology have been adopted for personal information management system. Bibliographic details for this publication are as follows:  
Shahzad S.K., Ontology-based User Interface Development: User Experience Elements Pattern, Journal of Universal Computer Science, vol. 17, no. 7 (2011), 1078-1088, submitted: 30/10/10, accepted: 15/3/11, appeared: 1/4/11 © J.UCS
- 3 Second publication is extended with introduction of User Interface Model (UIM) as a base model for the user interface. This extension resulted in third publication that is also discussed in 1.5. Bibliographic details for the publication are:  
Shahzad, S., Granitzer, M.: Ontological framework Driven GUI Development, Proceedings of 10th International Conference on Knowledge Management and Knowledge Technologies, 1-3 September 2010, Graz, Austria. P 198-206.



- 4 Last publication from the major stream of research is about user interface ontology. Same example of vCard has been used for these experiments to gain comparative results with last experiments. The bibliographic details for the publication is as:  
Shahzad, S., Granitzer, M., Helic D.: Ontological Model Driven GUI Development: User Interface Ontology Approach, 2011 6th International Conference on Computer Sciences and Convergence Information Technology (ICCIT), Seogwipo, Korea, Nov-Dec 2011.
- 5 Fifth publication is not from the core of the research method but it is done over the risks personal information management. It focuses at personal data sharing and publishing over web on different official and social portals. The bibliographic details for the publication is as follows:  
Sahito F., Slany W., Shahzad S., Search Engines: The Invader To Our Privacy ? A Survey, 2011 6th International Conference on Computer Sciences and Convergence Information Technology (ICCIT), Seogwipo, Korea, Nov-Dec 2011.

## 1.5. Thesis Organization

This thesis is organized in three major parts other than introduction to the research compilation.

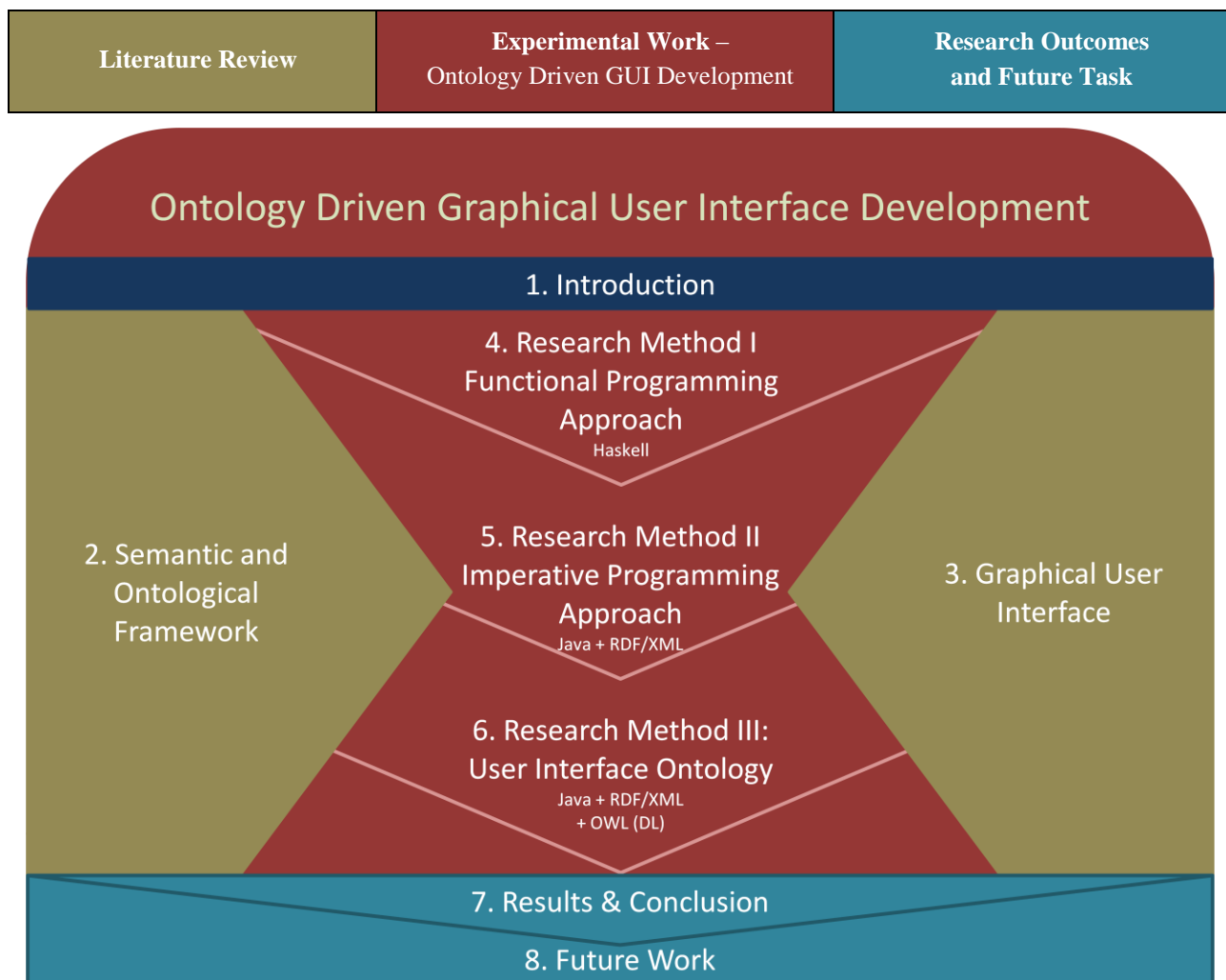


Figure 1.1: Thesis Architecture

Literature review provides definitions, the history and major research work of the base concepts used for our research. These base concepts are semantic and ontological framework that has roots from psychology and graphical user interface that is a technology based study area.

In experimental work we have explored the methodologies and strategies to structure the overlapping area of semantic and ontological framework through semantic mapping between domain ontology and its user interface concerns. At start we have adopted functional programming approach so that domain ontology (expressed in a mathematical function) can be directly mapped through bijective morphism to the user interface (as an IO function) to demonstrate the semantic and ontological set of rules

carriage to GUI. Second experiment methodology we have adopted imperative programming to provide a user interface development and designing procedure for the available standards (vCard) with imperative programming approach (Jena + RDF/XML) to generate user interface model and its instantiation by a graphical library (SWT). The last but not least experiment provides a methodology or improvement to the last experiment by introducing User Interface Ontology (UIO) to ensure semantic and ontological framework driven User interface properties and constrains and merging two the ontology (Domain Ontology and UIO) resulting a semantically consistent User Interface model which can be tested by any reasoner application.

Results and conclusion, the next section of the research compilation, discusses the results from experiments and testing of three adopted methodologies for GUI development based on semantic and ontological framework. Here we test and verify the semantically consistent knowledge delivery at GUI through a restricted design and development procedures. These procedures are restricted only to provide direct mapping and using semantic and ontological framework for providing a base model for user interface.

Future research work and research applications will be discussed in the last section of the writing.

# Semantic and Ontological Framework

Ontologies and semantic frameworks are now being used for specification of any real world concept. This part of literature survey discusses the history and the state of the art of semantics and ontological frameworks. Moreover the properties of this framework that has to be provided at GUI that to ensure a consistent knowledge delivery to user.

## 2.1. Ontology

Computer science always based on a documented formal specification of the concepts and their combination (their relationship and possible behavior), which have to be represented in a computer model. Ontology is base concept for this research that provides a base specification of concepts to present at any Graphical user interface (GUI). There is no such specified definition of ontology.

### 2.1.1. Definitions

#### 2.1.1.1. *Philosophical definition*

Ontology is a study and analysis of a logical description about “What exists” and its details stating the relationship of extents. Whatever is around us or proves its existence in many ways, either by its properties, actions or effects that can be conceived through the environment by human beings. These observations create a concept about any existent in human mind that can be written down in a logical fashion through ontology. Ontology is also defined as "formal, explicit specification of a shared conceptualization" by Tom Gruber (Gruber, 1995). Barry Smith defines ontology as the science which deals with the nature and the organization of reality (Smith, 1996).

#### 2.1.1.2. *Historical Definition*

Ontology, by history<sup>3</sup>, is a new name of metaphysics and/or a sub domain of philosophy, that talks about the reality and what exists in it. The base of ontology is concerned about the study and analysis of a similar and different existence modes and types with certain properties and relationships between one with whole and one with one. Here, whole means complete natural environment that identifies the relationships and associations in the reality to discover object and their categories.

---

<sup>3</sup> <http://www.ontology.co/idx03.htm>

### 2.1.1.3. *Modern / Working Definition*

Modern ontology is not exactly about finding the exact definition of what exist but about description of what is commonly conceived by human from nature. It is a way or study of expressions and defining the concepts as an image of reality in human brain. These concept are perceived from nature but contrary to the Greek philosophers (Parmenides, Plato, Aristotle etc.) rather than calling it an absolute truth but modern era name of things existent as concept (conceived from reality) or knowledge (proven facts). It defines ontology, a way of expression about concepts and knowledge. These concepts are directly dependent to the domain of knowledge. Same concepts can have different properties and behavior in different domains. These behaviors and relationships are also used for reasoning about the entities within that domain and may be used to describe the knowledge of the domain.

Thus it also introduces many different sub-domains of knowledge talking about the properties and their relationships and their representation like shared vocabulary, Formal Ontology, Mereology, taxonomy, Mereotopology, meta-ontology, physicalontology and some new fields like quantum ontology (Brey, 2005 ). These representations of relationships and study areas for concept description and analysis help us in our research to logically validate the consistency of information structure and limitations in a computational model. Here epistemology validate the process of information collection and verification, taxonomy works for classification, meretopology provides the hierarchical relationship of parts and complete set with the boundaries of set.

## 2.1.2. History

The question of being or “what exists” presents in human mind since the existence of human beings. While digging the history of ontology in literature we managed to attain the science of philosophy that answer the questions of what things exist in the world. The philosophy of anything starts with the question of what exists, what is reality, truth, properties, relationship and it counter effects that posits which objects exist in the world. Later this thought helps to validate the quality of information to the fields of knowledge discovery, knowledge sharing and representation in computational models for data.

### 2.1.2.1. *Greek Thought*

The discussion about existence of human and nature has been started much earlier. In history and literature the earliest evidence of these discussions have found in Greek times. Parmenides<sup>4</sup> (founder of the Eleatic school of philosophy), pre-Socratic philosophers at Elea and Plato (427-347 BC) (Platonic realism) started working at philosophy and metaphysic in this era. They also analyzed the

---

<sup>4</sup> <http://www.uni-leipzig.de/~philos/stekeler/aufsaeetze/twot-p.pdf>

truth, nature and argumentation about what exists and what not exists. This period also introduced the work on semantics and predication by Eleatic school in 500BC. This period is also famous for Aristotle work which is very modest attempt in the field of logic by categories, interpretation, prior Analytics, posterior Analytics and sophistical Refutations and also the relationship of universal and individual.

### 2.1.2.2. Modern Era

In 17th and 18th century the major work was done in central Europe on philosophy. Europeans described ontology as a field of study understanding and intelligible ideas and concept of the real world as individuals and complete (Christian Wolff, Gottfried Leibniz, Alexander Baumgarten, Immanuel Kant, Salomon Maimon, Bernard Bolzano, Franz Brentano, Alexius Meinong, Kazimierz Twardowski and Edmund Husserl). Since mid 1970s, researchers in the field of Artificial Intelligence (AI) have recognized that capturing knowledge is the key to building large and powerful AI systems.

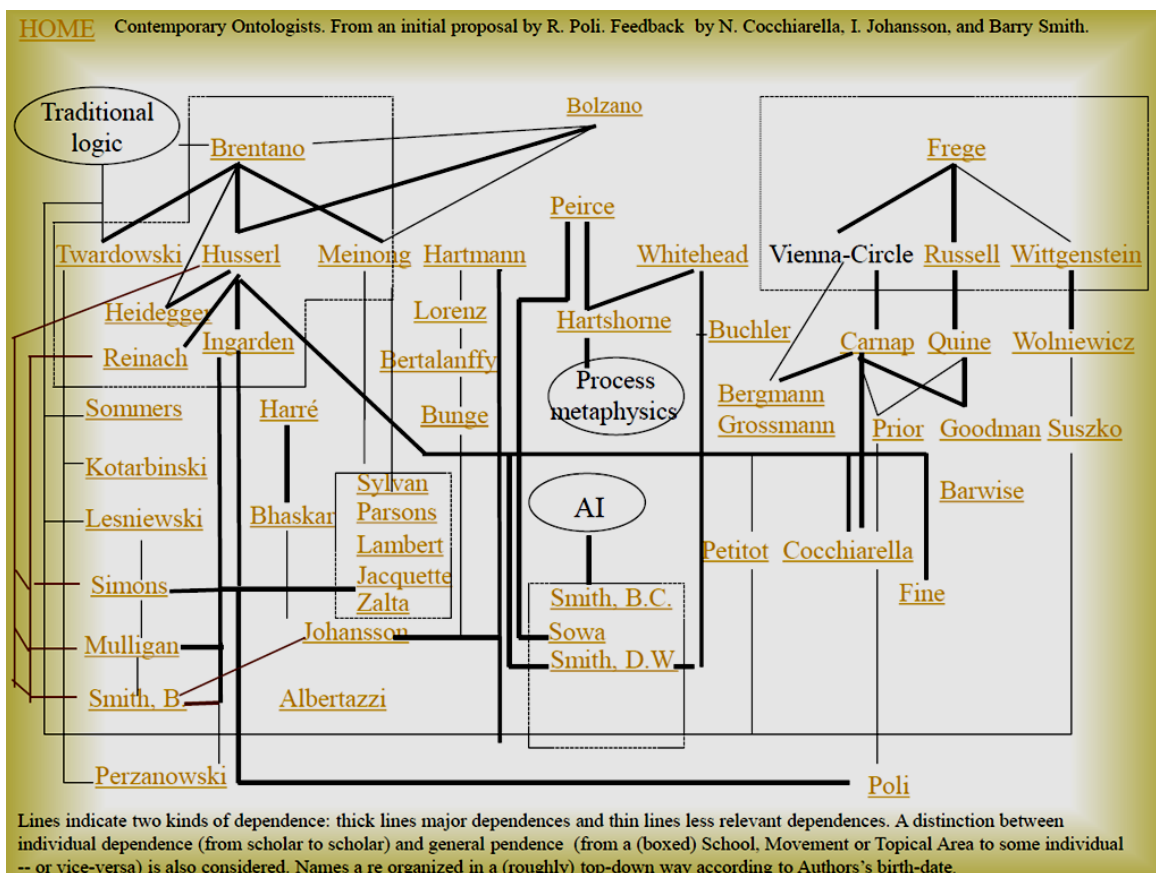


Figure 2.1: Table of Contemporary Ontologists<sup>5</sup>

<sup>5</sup> <http://www.ontology-2.com/essays/table-ontologists.pdf>

AI researchers argued that they could create new Ontologies as computational models that enable certain kinds of automated reasoning. In the 1980s, the AI community began to use the term Ontology referred to theory of a modeled world and a component of knowledge systems. Some researchers draw inspiration from philosophical Ontologies and viewed computational ontology as a kind of applied philosophy.

### 2.1.2.3. *Ontologies for computations*

The second half of the twentieth century was centered on the philosopher's debates on Ontologies building through possible and achievable methods without constructing any elaborated Ontologies themselves. Computer scientists on the other hand were aiming for robust Ontology, for instance, Cyc and WordNet. In the beginning of 90s, Tom Gruber's paper "Toward Principles for the Design of Ontologies Used for Knowledge Sharing" (Gruber, 1995) raised the eyebrows of the research agencies when it was recognized with the authentic definition of Ontology in computer science circle. Later on, Gruber introduced specification of a conceptualization that enumerate Ontology as a narration or description of the concepts and relationships exist for an agent, as an individual or a community of agents. By contrast, this term has a different meaning of the word as it is described by the philosophers.

Gruber has stated ontologies class definitions, their hierarchies and the subsumption relation, but studies argue that ontologies need not be limited to these forms. Ontologies are also not limited to conservative definitions in the traditional logic sense that only introduce terminology without providing any knowledge about the world. To specify a conceptualization, one needs to state axioms that do constrain the possible interpretations for the defined terms. (Gruber, 1993)

In current times, Barry Smith<sup>6</sup> has a big role in introduction and use of ontologies in information science and many interdisciplinary fields like Bioinformatics and Geographic Information System (GIS). He has done much work on ontology formalism, framework and for knowledge representation arguing logic, common sense and cognitive sciences. He has also discussed the problems of mereotopolgy and relationship between parts, as whole and among boundaries. (Smith, 1978) (Smith, 2004) (Smith, 1995) (Smith, 1989) (Smith, 1996) (Smith, 2008) (Smith, et al., 2001)

Ontologies are essential part of the W3C standards family of the Semantic Web that identifies standard conceptual vocabularies. To attain enhanced searching of the desired information from the bulk data, semantic web has drawn these Ontologies and semantic framework to classify and organize data as technical or implementation criterion. This shift has brought information science to the implementation level for computer user or a lay man (Najar, et al., 2009). This direct connection is also used to classify information and make machine understand to exchange information among

---

<sup>6</sup> <http://ontology.buffalo.edu/smith/>

systems. The aim is also to recommend services to assist interoperability across multiple, publish reusable knowledge bases to provide services to heterogeneous systems and databases as well as for respond queries. Furthermore, identification of data modeling representation at abstraction level above database design (logical or physical) is also the major role of Ontologies to specify that information or data can be queried, translated, unified and explored independent based systems and services, that leads to integration of web services and database interoperability.

### 2.1.3. Metaphysics

Metaphysics is a very broad field and according to metaphysicians it is a science that studies qua being. In other words, it determines the real nature of things or it considers reality that determines the meaning, structure and existing of actual objects. Meinong argue<sup>7</sup> that the theory of objects is a priori science. According to him, this principle concerns with the existent or nonexistent and whole of what is given. In the science of reality, existent entities must be distinguished from ideal objects or subsistent, for instance, diversity, identity or number. He further explains that the 'pure object' considered in the theory of objects is beyond being and nonbeing, whereas, existence and subsistence objects are the two forms of being by Alexius Meinong and Ernst Mally from the Graz School of experimental psychology and object theory in 1904.<sup>7</sup>

Mostly ontology is considered as a sub domain of metaphysics. Although the term "ontology" and "metaphysics" are far from being univocal and determinate in philosophical jargon, an important distinction seems often enough to be marked by them. What we may call ontology is the attempt to say what entities exist proven by logic. Metaphysics, by contrast, is the attempt to say, of those entities and what they are. In contrast, one's ontology is one's list of entities, while one's metaphysics is an explanatory theory about the nature of those entities.

## 2.2. Semantics and Ontological framework

A conceptual model is a representation (typically graphical) constructed by Information Systems professionals of a specific sub domain (any individual or a class) perception of a real-world domain. It might be used to facilitate the design and implementation of an information system. It might be used to evaluate the balance between an organization's needs and the business models embedded within an enterprise application software package (Shanks, et al., 2003).

---

<sup>7</sup> <http://www.ontology.co/meinonga.htm>



### 2.2.1. Semantics

Semantics is the term, adopted from linguistics, which talks about the words and their formation in a sentence to express some ideas. It is a study of syntax and vocabulary of the language. Thus a concept can be represented in a combination of semantics and ontologies. Ontology defines what it is, and semantics provide a way of expression for share the ontologically defined concepts.

Though there is a huge history of work on semantics and natural language evolutions, still it is not completely formalized due to complex human nature. With all of difficulties in natural languages, there are also many achievements in Natural language processing and now machines can perform natural conversation with human (Zaihrayeu, et al., 2007). In the field of computer science, machine to machine communication is comparatively easier than human-computer communication due to translation process, thus Semantics and Ontology were formalized much earlier by simple logical grammar and constructing precise mathematical models.

Semantical properties of the web processes are expressed in form of ontologies. Ontology is a document or a file that formally defines relations among terms. In the web service model, ontologies consist of hierarchical definitions of important concepts and description of the properties of each concept. The ontologies can be defined in DAML-OIL or OWL (Georgiev, 2005).

### 2.2.2. Domain Ontologies and upper Ontologies

A domain ontology (or domain-specific ontology) models a specific domain which represents part of the world. Particular meanings of terms applied to that domain are provided by domain ontology. For example, the word card has many different meanings. An ontology about the domain of poker would model the "playing card" meaning of the word, while an ontology about the domain of computer hardware would model the "punched card" and "video card" meanings.

An upper ontology (or foundation ontology) is a model of the common objects that are generally applicable across a wide range of domain ontologies. It employs a core glossary that contains the terms and associated object descriptions as they are used in various relevant domain sets (Navigli, et al., 2004). There are several standardized upper ontologies available for use including Dublin Core<sup>8</sup>, GFO<sup>9</sup>, OpenCyc/ResearchCyc<sup>10</sup>, SUMO<sup>11</sup> and DOLCE<sup>12</sup> (Gangemi, et al., 2002). The WordNet,

---

<sup>8</sup> <http://dublincore.org/>

<sup>9</sup> <http://www.onto-med.de/ontologies/gfo/>

<sup>10</sup> <http://research.cyc.com/>

<sup>11</sup> <http://www.ontologyportal.org/>

<sup>12</sup> <http://www.loa.istc.cnr.it/DOLCE.html>

while considered an upper ontology by some, is not strictly an ontology. However, it has been employed as a linguistic tool for learning domain ontologies. In philosophy, the term formal ontology is used to refer to an ontology defined by axioms in a formal language with the goal to provide an unbiased (domain- and application-independent) view on reality, which can help the modeler of domain- or application-specific ontologies (information science) to avoid possibly erroneous ontological assumptions encountered in modeling large-scale ontologies. By maintaining an independent view on reality formal (upper level) ontology gains the following properties: indefinite expandability: the ontology remains consistent with increasing content. The Gellish<sup>13</sup> ontology is an example of a combination of an upper and domain ontology.

Since domain ontologies represent concepts in very specific and often eclectic ways, they are often incompatible. As systems that rely on domain ontologies expand, they often need to merge domain ontologies into a more general representation. This presents a challenge to the ontology designer. Different ontologies in the same domain can also arise due to different perceptions of the domain based on cultural background, education, ideology and due to different languages.

At present, merging ontologies that are not developed from common foundation ontology is a largely manual process and therefore is time-consuming and expensive. Domain ontologies that use the same foundation ontology to provide a set of basic elements with which to specify the meanings of the domain ontology elements can be merged automatically. There are studies on generalized techniques for merging ontologies, but this area of research is still largely theoretical (Guarino, et al., 1995).

### 2.2.3. Ontology components

Ontologies are commonly encoded using ontology languages. An ontological definition of a domain can include following common components provided by majority of ontology engineering languages and packages like Protégé (OWL)<sup>14</sup>.

#### 2.2.3.1. *Individuals*

Individuals or instances are the ground level elements of an ontology that may embrace objects like people, animals, planets, chairs, vehicles and particles. Abstract individuals, on the other side, include numbers and words. Ontology is not limited to any individuals but the aim is to provide means of classify different individuals. However, only the utterances of numbers and words are believed as individuals in formal extensional Ontologies where names and numbers are themselves considered as classes such as ISO 15926 and the IDEAS Group model.

---

<sup>13</sup> <http://www.gellish.net/>

<sup>14</sup> <http://owl.cs.manchester.ac.uk/tutorials/protegeowltutorial/>

### 2.2.3.2. *Classes*

Classes can be defined as a subclass of collection that may classify individuals, classes, or a combination of both, for instance: Person, animal, automobile, car, class or Thing. The classes of ontology may be extensional or intensional in nature and always vary on different classes, such as, whether a class can belong to itself or as a universal class. A class is extensional if and only if it is exemplify by its membership. However, the class is intensional if a class does not satisfy this condition. Intensionally classes often have necessary conditions associated with membership in each class where classes with sufficient conditions considered as a fully *defined* class (Guarino, et al., 1995). A class can subsume or be subsumed by other classes as well. In this regard, if a class subsumed by a different class is called a *subtype* or subclass of the *supertype* class or subsuming class. In most ontologies, classes are permitted to have any number of parents (*multiple inheritance*), but in some ontologies examples, a class is only permitted to have one parent (*single inheritance*). Though, all essential properties of each parent are inherited by the subsumed child class.

Importantly, a partition is a set of related classes and associated rules that authorize objects to be classified by the proper subclass that distinguish the subclasses from the superclasses. If the one partition rule(s) guarantee that a single object cannot be in both classes, then the partition is considered to be a disjoint partition. However, if these rules claim that every concrete object in the super-class is an example of any one of the partition classes, then it will be considered as an exhaustive partition.

### 2.2.3.3. *Attributes*

Attributes are basically concepts in ontology that can be explained by relating them to other concepts things or parts and this relation can be described as *attributes*. However attributes may be independent concepts or things or attributes can themselves considered as an individual or a class. The type of object or the types of attribute establish the kind of relation between them to state a fact that is specific to that particular object which it is related.

### 2.2.3.4. *Relationships*

Relationships specify how and what sense objects are related to other objects in Ontology. For instance, the concept Ford Bronco and concept Ford Explorer might be related by type <is defined as a successor of> and can be expressed as Ford Explorer *is defined as a successor of*: Ford Bronco. This example elaborate that how the Explorer has replaced the Bronco and also illustrates that relation has a direction of expression. On the other side, the inverse expression will state the same fact with a reverse phrase.

The list of relations explains the semantics of the domain that also demonstrates the power of Ontologies, such as, the subsumption relation defines which objects are classified by different classes. The examples are: *is-a-subclass-of*, *is-a-superclass-of* and the converse of *is-a*, *is-a-subtype-of*. These

relationships create taxonomy such as tree-like structure to define how objects are relates to other objects. The mereology relation is also another example that represent composite objects such as Steering Wheel ("Steering Wheel is-by-definition-a-part-of-a Ford Explorer").

### 2.2.4. Vocabulary

Ontology renders shared vocabulary and taxonomy which models a domain with the definition of objects and/or concepts and their properties and relations.<sup>15</sup> This vocabulary consists of all the entities taking part in the formation of an ontology. It is a set of individuals independent of its classification and relationship. In formal ontology, a domain ontology's vocabulary is the set of entities (concepts) participating in ontology without its relationship details. Vocabulary specifies participating concept and its type like class, concept, literal, attribute etc.

### 2.2.5. Taxonomy and Meretopology

Taxonomy (from Greek *taxis* meaning arrangement or division and *nomos* meaning law) is the science of classification according to a pre-determined logical criterion. In formal science it is mostly define as a classification tree of specific domain ontology that results as a catalog used to provide a conceptual framework for discussion, analysis, or information retrieval.

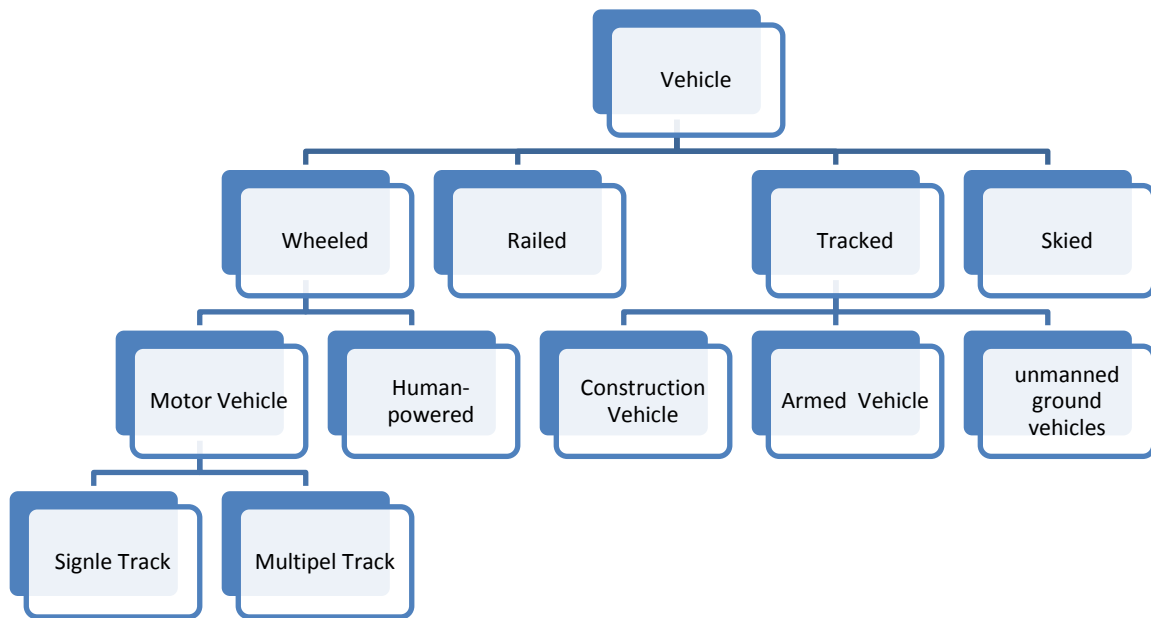


Figure 2.2: Example Taxonomy of Vehicles

---

<sup>15</sup> <http://www.ida.liu.se/~janma/SemWeb/Slides/ontologies1.pdf>

Development of a taxonomy is a part of ontology engineering process that takes into account the criterion of the importance of separating elements of a group (taxon) into subgroups (taxa) that are mutually exclusive, unambiguous, and taken together, include all possibilities. In practice, a good taxonomy should be simple, easy to remember, and easy to use.

Barry Smith defines mereotopology as a theory of parts and boundaries. It is a combination of mereology and topology that provides the topological notions of boundaries and connections married to mereology. (Smith, 1996)

These taxonomy and mereotopology correspond to the concepts of inheritance and aggregation respectively from computer science field of object oriented modeling of reality. The rigidity and technology bias have motivated concepts based entities and structures to provide more realistic modeling of the reality.

### 2.2.6. Universe of Discourse

The term universe of discourse generally refers to the collection of objects being discussed in a specific discourse. In model-theoretical semantics, a universe of discourse is the set of entities that a model is based on. In formal logic, ontologies are mostly defined for specific domain rather than complete universe (Gruber, 1995). These domain ontologies anyhow belongs to a specific range of values called Universe of discourse (also called the domain of discourse). It is the set of complete range of objects, events, attributes, relations, ideas, etc., that are expressed, assumed, or implied in a discussion. Ontological specification and ontology engineering also allows people to specify the bounds of the universe of discourse (restricting it arbitrarily) allows them to pass subtle fallacies unnoticed. The domain of discourse is usually identified in the preliminaries, so that there is no need in the further treatment to specify each time the range of the relevant variables. A database is a model of some aspect of the reality of an organization. It is conventional to call this reality the "universe of discourse" or "domain of discourse".

### 2.2.7. Descriptive, Formal and Formalized Ontologies

Robert Poli has classified three forms of ontologies as descriptive formal and formalized ontology. Descriptive ontology concerns the collection of such prima facie information either in some specific domain of analysis or in general. Formal ontology distills, filters, codifies and organizes the results of descriptive ontology (in either its local or global setting). According to this interpretation, formal ontology is formal in the sense used by Husserl in his Logical Investigations. Being 'formal' in such a sense therefore means dealing with categories like thing, process, matter, whole, part, and number. These are pure categories that characterize aspects or types of reality and still have nothing to do with the use of any specific formalism. Formal codification in the strict sense is undertaken at the third level of theory construction: namely that of formalized ontology. The task here is to find the proper formal codification for the constructs descriptively acquired and formally purified in the way just

indicated. The level of formalized constructions also relates to evaluation of content and context independence: any kind of 'concept' can find its place (Poli, 2003).

## 2.3. Ontologies for Knowledge Representation and Sharing

A body of formally represented knowledge is based on a *conceptualization*: the objects, concepts, and other entities that are assumed to exist in some area of interest and the relationships that hold among them. A conceptualization is an abstract, simplified view of the world that we wish to represent for some purpose. Every knowledge base, knowledge-based system, or knowledge-level agent is committed to some conceptualization, explicitly or implicitly

At the end of 20<sup>th</sup> century, formal ontologies are widely supported for content specification for variety of knowledge-based software and knowledge sharing agreement (on information systems) (Gruber, 1995) (Guarino, 1995) (Guarino, 1998) (Cocchiarella, 1996) (Pirlein, et al., 1995) (Wand, et al., 1990) . Gruber also mentioned many individual business agreements of different information systems where ontological specifications are used or argued for knowledge representation and sharing. Problems in earlier knowledgebase software and services were the motivation behind adopting formal ontological specification for knowledge representation, sharing and building reusable knowledge components.

### 2.3.1. Epistemology and Knowledge

In philosophy, epistemology is the study of the nature, origin, and limits of human knowledge. This field discusses knowledge, and methods to acquire and prove knowledge as theory of knowledge. Epistemology in combination with ontology forms another branch of philosophy known as metaphysics.

Epistemology has a long history, starting from ancient Greeks till today in many different sub branches like knowledge discovery, management, representation and visualization in information systems.

### 2.3.2. Knowledge management

Knowledge management (KM) emerged as a new field of study at start of 21<sup>st</sup> century. It provides a set of procedures for identification, creation, representation and distribution of knowledge. This knowledge as a strategic asset of the organization resides in various forms in organization personnel, roles, documentations, procedures, or strategies formed through experience. This study of creation and compilation in standard representation of knowledge is required for knowledge sharing, integration and continuous improvement of the organization. KM efforts overlap with organizational learning,

and may be distinguished from that by a greater focus on the management of knowledge as a strategic asset and a focus on encouraging the sharing of knowledge.

KM was introduced as an established discipline since 1991<sup>16</sup> by Ikujiro Nonaka. Later on it was included as a subjects taught in the fields of business administration, information systems, management, and library and information sciences (Alavi, et al., 1999). For last two decades, electronic media and information systems have become the main actor in knowledge sharing for internal or external communication of knowledge (e.g. Digital Libraries). Thus, KM research area has gained much contribution from information science and mass communications as well. Web 2.0 has widened the thought by introducing publicly presented information like wiki and feedbacks like blogs. Still, the focal point of Knowledge management efforts is organizational objectives either this knowledge sharing is between in Organization and individual or organization to organization.

In our research we need to explore the study of knowledge that presented through ontologies and need to be visualised through GUI.

#### 2.3.2.1. *Knowledge Discovery*

Knowledge discovery was introduced as sub domain of data mining, an information search in large volume patterns through a database. It talks about search mechanism and the resulting knowledge discovery. This complex topic can be categorized according to 1) what kind of data is searched; and 2) in what form is the result of the search represented. There are many concerns regarding knowledge discovery like context of the search and data ranges. Same query can result in different fashion while activated from different context or roles. It focuses more on methodologies to improve quality of search in data availability and consistency. (Frawley, et al., 1992) (Fayyad, et al., 1996)

Software mining is another artistic application of knowledge discovery in the area of software modernization that analyzes existing software artifacts. It is implementation of reverse engineering concepts. Knowledge Discovery Metamodel (KDM) from Object Management Group (OMG) is a well known implementation of software mining resulting in an ontology (a set of definitions) for system knowledge extraction and analysis.<sup>17</sup>

Semantic Web has changed the scope of knowledge discovery in two major aspects; first a large and heterogeneous data source secondly semantic search. There is a huge revolution in knowledge discovery through search engines semantic search and semantic databases.

---

<sup>16</sup> <http://hbr.org/2007/07/the-knowledge-creating-company/es>

<sup>17</sup> <http://www.omg.org/technology/kdm/index.htm>

### 2.3.2.2. *Knowledge Representation*

Knowledge Representation (KR) is a study of construction and analysis of formal and logical axioms by specific symbol vocabulary that can describe the knowledge as a set of facts. This is an analysis of the quality of reasoning in terms of accuracy, effectiveness and use of symbols for a knowledge domain representation. It is a combination of symbols, operators and interpretation theory. KR provides a logical sequence that can best represent the knowledge. In the KR system, logic is used to deliver the formal semantics for reasoning procedure, operator functions and KR refinement procedures. These operators and operations can include negation, conjunction, adverbs, adjectives, quantifiers and modal operators. The logic works as the interpretation theory.

The quality parameter of a KR is its expressivity. An expressive a KR, is the best fit representation of the domain. However, expressivity is a trade off by complexity, completeness and consistency. The more expressive language used for KR will contain more complex logic and algorithms to construct equivalent inferences. Less expressive KRs may be complete and consistent. Autoepistemic temporal modal logic is an example of highly expressive KR system while Propositional logic is much less expressive but highly consistent and complete with minimal algorithm complexity.

Semantic Web has brought many recent developments in KR systems through XML-based knowledge representation languages and standards development. Resource Description Framework (RDF), RDF Schema, Topic Maps, DARPA Agent Markup Language (DAML), Ontology Inference Layer (OIL), and Web Ontology Language (OWL) are the major examples of XML based KR languages. Several KR techniques like frames, rules, tagging, and semantic networks are fetched from Cognitive Science. The objective of KR is to facilitate reasoning, inferencing and drawing conclusions. A good KR must be both declarative and procedural knowledge.

A suitable choice of knowledge representation simplifies the problem solving task for the field of artificial intelligence. KR makes many problems easier to solve for any represented knowledge domain especially for analytical problems.

Martin (Martin, 2002) has devised five distinct characteristics of KR:

- 1 Substitute of Knowledge domain
- 2 Ontological Commitments
- 3 Set of three components
- 4 Fundamental concept
- 5 Constraints
- 6 Recommendations
- 7 Pragmatic Efficiency
- 8 Formal language



### 2.3.2.3. *Knowledge Sharing*

Knowledge sharing provides the procedure to exchange knowledge (i.e. information, skills, or expertise). This knowledge exchange can be done among people or organizations.

Knowledge comprises a valuable intangible asset for any organization to create and maintain competitive advantages (Fayyad, et al., 1996). Knowledge Sharing is a part of knowledge management system. However, technology is one of the many factors that can affect the sharing of knowledge within organizations like organizational mores, trust levels and provided incentives. Human factor, if exists, is a big obstacle in knowledge sharing in organization. Another obstruction can be the ownership of knowledge thus very important.

That human factor problem doesn't exist in that sense on web or publically available open databases like it is for any commercial or classified information. Publically published data on web (wiki) have other problems of knowledge discovery, copy right issues or integrating different semantically related knowledge shares.

Formal ontology provides standards for creating a shareable KR of the specific domain. Smith Gaurino and Gruber has support the idea of using formal ontology for knowledge representation and sharing (Gruber, 1993) (Gruber, 1995) (Guarino, 1995) (Smith, 2004). There are many open database and knowledge shares defined using Ontology and semantic framework like Linked Open Data (LOD)<sup>18</sup>.

## 2.4. **Computer Applications of Ontology**

Ontologies provide a definition common vocabulary of a specified domain and define, with different levels of formality, the meaning of the terms and the relationships between them. During the last decade, increasing attention has been focused on ontologies. Ontologies are now widely used in knowledge engineering, artificial intelligence and computer science; in applications related to areas such as knowledge management, natural language processing, e-commerce, intelligent information integration, bio-informatics, education; and in new emerging fields like the semantic web. Ontological engineering is a new field of study concerning the ontology development process, the ontology life cycle, the methods and methodologies for building ontologies, and the tool suites and languages that support them.

---

<sup>18</sup> <http://linkeddata.org/>

### 2.4.1. Ontological Concept Modeling

Gruber's definition of ontology "*formal, explicit specification of a shared conceptualization*" and many writings from other researchers have argued using ontology for specification of concepts and reality representation (Guarino, 1995) (Smith, 2004). These concepts are the human brain model of any other concept of real world. The concept can be modeled as Knowledge Representation of the specific domain by ontology and semantic framework.

### 2.4.2. Ontology Engineering

Ontology engineering is a sequence of tasks for the development of ontological definition of a particular domain. This process also referred to ontology designing for the domain concept. Resulting domain ontology is a KR of the concept. It states the axioms stating the fundamental concept, its properties and relationships. It is a metadata design for a database semantically defined and categorized by taxonomy. Thus it is more conceptual representation, with logically specified rules and constraints.

Ontology engineering assists solution to the inter-operability problems brought about by semantic obstacles, i.e. the barriers regarding the definitions and sharing of business terminology and software classes.

#### 2.4.2.1. *Design criteria for ontologies*

Design decisions are required to make for building and ontological representation of any domain concept. This ontological design need to be verified at some objective criteria that are founded on the purpose of the resulting artifact, rather than based on a priori notions of naturalness or truth. Gruber has defined the objectives to use semantic and ontological framework as design criteria of ontology engineering for any domain concept (Gruber, 1995).

- 1 Clarity
- 2 Coherence
- 3 Extendibility
- 4 Minimal encoding bias
- 5 Minimal ontological commitment

### 2.4.3. Cognitive Science and AI

Christopher Longuet-Higgins coined the term of cognitive science in his 1973 commentary on the Lighthill report<sup>19</sup>. Cognitive science is a scientific study of mind and its processes. It defines cognition

---

<sup>19</sup> [http://www.chilton-computing.org.uk/inf/literature/reports/lighthill\\_report/overview.htm](http://www.chilton-computing.org.uk/inf/literature/reports/lighthill_report/overview.htm)

and its functions. It analyzes the processes of information observation, storing, understanding, processing, representation, and transformation in actions through nervous system in human animal or machine. Cognitive science contains multiple research disciplines like psychology, artificial intelligence, philosophy, neuroscience, linguistics, anthropology, sociology, learning and education. It is expanded to many levels of analysis, from low-level learning and decision making mechanisms to high-level logic and planning, while on physical level from neural circuitry to modular brain organization. AI is a computerized application and inspiration of human or animal cognition to make machines enable for decision making and responding to the nature in the same fashion. AI has introduced formal ontology for understanding the nature with natural cognition procedures and image common sense (Smith, 1995).

#### 2.4.4. Ontology Languages

Ontology languages are formal languages defined for ontologies construction. It permits the building of knowledge about specific domains and often includes reasoning rules that support the processing and evaluation of that knowledge. These languages are mostly declarative, generalizations of frame and based on either first-order logic or description logic. Here are some common

- 1 Common Logic - and its dialects
- 2 CycL
- 3 DAML+OIL
- 4 DOGMA (Developing Ontology-Grounded Methods and Applications)
- 5 F-Logic (Frame Logic)
- 6 KIF (Knowledge Interchange Format) and Ontolingua based on KIF
- 7 KL-ONE
- 8 KM programming language
- 9 LOOM (ontology)
- 10 OCML (Operational Conceptual Modelling Language)
- 11 OKBC (Open Knowledge Base Connectivity)
- 12 Ontology Inference Layer (OIL)
- 13 Web Ontology Language (OWL)
- 14 PLIB (Parts LIBrary)
- 15 RACER
- 16 Resource Description Framework (RDF)
- 17 RDF Schema
- 18 SHOE

## 2.5. RDF /OWL

The Resource Description Framework (RDF) is an open format framework to define knowledge<sup>20</sup>. It belongs to the family of World Wide Web Consortium (W3C) specifications originally designed as a metadata data model<sup>21</sup>. It facilitates building a structure with RDF statements called RDF *triples* these statements are formed as subject-predicate-object expressions. It is similar to classic conceptual modeling approaches like relational modeling or Object Oriented Modeling. The subject denotes the resource, and the predicate denotes traits or aspects of the resource and expresses a relationship between the subject and the object. In an example “Sebastian Lives in Graz”; “Sebastian” is stated as subject while “Lives in” and “Graz” are examples of predicate and object respectively. The RDF defines a specific set of vocabulary defined by the RDF specification is as follows consists of *classes* and *properties*.

W3C's Semantic Web has a major component is resource description that allows automated software to store, exchange, and use machine-readable information distributed throughout the Web. Thus, it enables users to deal with the information with greater efficiency and certainty. RDF's simple data model and ability to model disparate, abstract concepts has also led to its increasing use in knowledge management applications unrelated to Semantic Web activity.

A combination of RDF statements fundamentally represents a labeled, directed multi-graph. As such, an RDF-based data model is more naturally suited to specific kind of KR than the relational model and other ontological models. However, RDF data is usually stored in relational database or native representations also like Triplestores, or Quad stores if context is also persevered for each RDF triple. As RDFS and OWL demonstrate, additional ontology languages can be built upon RDF.

### 2.5.1. Serialization formats

RDF has two common serialization formats in use:

**XML format:** This format is often called RDF/XML because it was introduced among the W3C specifications while defining RDF. However, it is important to distinguish the XML format from the abstract RDF model itself. Its MIME media type, application/rdf+xml, was registered by RFC 3870. It recommends RDF documents to follow the new 2004 specifications.

**Notation 3 (or N3):** It is introduced as a non-XML serialization of RDF models by W3C, for an easier understanding and updates by human. It is based on a tabular notation, with a sequences of rows

---

<sup>20</sup> <http://www.w3.org/TR/PR-rdf-syntax/>

<sup>21</sup> <http://www.w3.org/RDF/>

of triples [Sub, Pred, Object]. N3 is more close to the Turtle and N-Triples formats than RDF/XML. These triples can be stored in a triplestore.

### 2.5.2. Resource identification

The subject, object or predicate are defined as a resource; these resource can be a link to the original resource called Uniform Resource Identifier (URI). Subject or object resources can be a blank node as well, is called anonymous resources. While object resources can also be a Unicode string literal. URI is a link to the original recourse. Predicate URI indicates to a resource, representing a relationship. The URI looks like URL and it can be a URL but not necessarily. In popular Semantic Web applications of RDF like RSS, Linked Data and FOAF resources tend to be represented by URIs that intentionally denote, and can be used to access, actual data on the World Wide Web.

### 2.5.3. Statement reification and context

Reification is defined as use of whole statement in a knowledge modeling is used as a resource through a URI. For example a statement like “X saw that Y is going to place-P” here Subject X is linked by predicate “saw” with complete statement “Y is going to place-P” as object of the statement. Reification can be required in modeling knowledge to create a relationship between a resources and a statement. Reification is sometimes important in order to deduce a level of confidence or degree of usefulness for each statement. In a reified RDF description, each original statement, being a resource, itself, most likely has at least three additional statements made about it: one to assert that its subject is some resource, one to assert that its predicate is some resource, and one to assert that its object is some resource or literal. More statements about the original statement may also exist, depending on the application's needs.

### 2.5.4. Query and inference languages

SPARQL (SPARQL Protocol and RDF Query Language) is RDF query language made a standard by the RDF Data Access Working Group (DAWG) of the W3C<sup>22</sup>. It is predominant SQL-like query language for RDF graphs. SPARQL is a recommendation of the W3C as of January 15, 2008<sup>23</sup>.

Other query languages to query RDF graphs include:

RDQL (RDF Data Query Language) ia a SQL-like query language. It is the predecessor to SPARQL. Versa is compact syntax (non-SQL-like) query language that is solely implemented in 4Suite (Python). RQL is one of the first declarative, semi-structured query languages for uniformly querying

---

<sup>22</sup> [http://www.w3.org/2009/sparql/wiki/Main\\_Page](http://www.w3.org/2009/sparql/wiki/Main_Page)

<sup>23</sup> <http://www.w3.org/TR/rdf-sparql-query/>

RDF schemas and resource descriptions, implemented in RDFSuite. SeRQL (Sesame RDF Query Language) is part of Sesame

XUL has a template element in which to declare rules for matching data in RDF. XUL uses RDF extensively for data binding with user interfaces.

## 2.5.5. RDFS

RDF Schema provides the framework to build an RDF model with application-specific classes and properties. Classes in RDF Schema are much like classes in object oriented programming languages. This allows resources to be defined as instances of classes, and subclasses of classes. It is a schema language to develop RDF/XML specification through the set of schema classes with specific properties using the RDF language rules and grammar. It provides basic elements for the description of ontologies as RDF vocabularies, which has to be structured to present RDF resources. These structured elements with RDFS in a triplestore can be accessible using any semantic query language like SPARQL.

The first version was published by the World-Wide Web Consortium (W3C) in April 1998, and the final W3C recommendation was released in February 2004<sup>24</sup>. Many RDFS components are also part of a more expressive Web Ontology Language (OWL).

### 2.5.5.1. Main RDFS constructs

RDFS constructs of RDFS classes, their associated properties and the utility properties to develop the limited vocabulary of RDF for a specific domain.

#### **CLASSES**

RDFS provides following classes:

*rdfs:Resource* – *rdfs:Class* – *rdfs:Literal* – *rdfs:Datatype* – *rdf:XMLLiteral* – *rdf:Property*

#### **PROPERTIES**

Following belong to RDFS as *rdf:Property*.

*rdfs:domain* – *rdfs:range* – *rdf:type* – *rdfs:subClassOf* – *rdfs:subPropertyOf* – *rdfs:label* – *rdfs:comment*

#### **Utility Properties**

*rdfs:seeAlso* – *rdfs:isDefinedBy*

---

<sup>24</sup> <http://www.w3.org/2001/sw/wiki/RDFS>

### 2.5.6. OWL

The Web Ontology Language (OWL) is a set of RDF/XML-based serialized KR languages for ontologies publishing. These languages are characterized by formal semantics for the Semantic Web. World Wide Web Consortium (W3C) has endorsed OWL. It got much attention from academics, medical and commercial purpose.

In October 2007, a new W3C working group was started to extend OWL with several new features as proposed in the OWL 1.1 member submission. W3C announced the new version of OWL on 27 October 2009. This new version, called OWL 2, soon found its way into semantic editors such as Protégé and semantic reasoners such as Pellet, RacerPro, FaCT++ and HermiT.

The OWL family contains many species, serializations, syntaxes and specifications with similar names. OWL and OWL2 are used to refer to the 2004 and 2009 specifications, respectively. Full species names will be used, including specification version (for example, OWL2 EL). When referring more generally, OWL family will be used.

## 2.6. Semantic Web

Tim Berners-Lee, the inventor of the World Wide Web and director of the World Wide Web Consortium ("W3C"), coined the term of the "Semantic web". He defines it as "*a web of data that can be processed directly and indirectly by machines*"<sup>25</sup> by combining data with data description details at web.

The Semantic Web is a collaborative work led by the W3C that promotes common formats for data on the World Wide Web. According to the W3C, "*The Semantic Web provides a common framework that allows data to be shared and reused across application, enterprise, and community boundaries*"<sup>26</sup>. This aim of portability and sharing is achieved by introducing semantic content in web pages. It builds on the W3C's Resource Description Framework (RDF) in mostly RDF/XML format.

Semantic Web had lot of challenges regarding many properties of data available at web like vastness, vagueness, uncertainty, inconsistency, and deceit. Any automated reasoning and analysis systems have to deal with all of these issues to make this huge heterogenous data-ware portable and shareable through the Semantic Web.

The World Wide Web Consortium (W3C) Incubator Group for Uncertainty Reasoning for the World Wide Web (URW3-XG) final report lumps the problems together regarding "uncertainty" stating

---

<sup>25</sup> <http://www.scientificamerican.com/article.cfm?id=the-semantic-web>

<sup>26</sup> <http://www.w3.org/2001/sw/>

“better defining the challenge of reasoning with and representing uncertain information available through the World Wide Web and related WWW technologies.”<sup>27</sup>

## 2.6.1. Standards

W3C provides Semantic Web standards in the context of Web 3.0.

### 2.6.1.1. Components

The Semantic Web Stack demonstrates the architecture of the Semantic Web. The functions and relationships of the components are summarized in Figure 2.3:

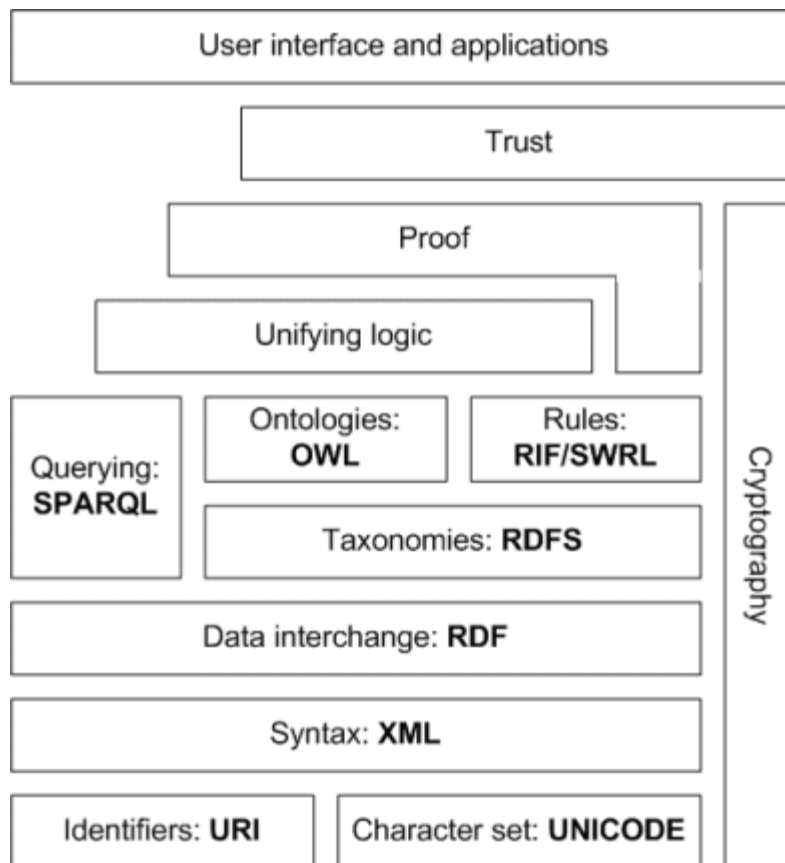


Figure 2.3 The Semantic Web Stack

The term "Semantic Web" is most of the times referred precisely to the adopted formats and technologies. The technologies and format used to increase utility of the available knowledge by enabling advanced searching browsing and evaluation (Hitzler, et al., 2009). It provides formal

---

<sup>27</sup> <http://www.w3.org/2005/Incubator/urw3/XGR-urw3-20080331/>



description of concepts, terms, and relationships within a given knowledge domain in linked data. These technologies are specified as W3C standards and include:

- 1 Resource Description Framework (RDF)
- 2 RDF Schema (RDFS)
- 3 Simple Knowledge Organization System (SKOS)
- 4 SPARQL, an RDF query language
- 5 Notation3 (N3), designed with human-readability in mind
- 6 N-Triples, a format for storing and transmitting data
- 7 Turtle (Terse RDF Triple Language)
- 8 Web Ontology Language (OWL), a family of knowledge representation languages

## 2.6.2. Projects

Here is a list of some mostly adopted, used and well known of the many projects and tools that exist to create Semantic Web solutions.

### 2.6.2.1. *DBpedia*

DBpedia is published structured data fetched from Wikipedia in RDF. It is available on the Web under the GNU Free Documentation License. Thus is also allows other Semantic Web agents to provide inferencing and advanced querying for the knowledge sharing.

### 2.6.2.2. *FOAF*

Friend of a Friend (or FoaF) provides a view and opportunity to analyze social relationships in the Semantic Web. It also uses RDF to describe the relationships people have to other people and the "things" around them. FOAF permits intelligent agents to make sense of the thousands of connections people have with each other, their jobs and the items important to their lives.

### 2.6.2.3. *GoodRelations for e-commerce*

The GoodRelations ontology is a popular vocabulary for expressing product information, prices, payment options, etc. It also allows expressing demand in a straightforward fashion. GoodRelations has been adopted by Google, BestBuy, Overstock, Yahoo, OpenLink Software, O'Reilly Media, the Book Mashup, and many others.

### 2.6.2.4. *SIOC*

The Semantically-Interlinked Online Communities project (SIOC, pronounced "shock") provides a vocabulary of terms and relationships that model web data spaces. Examples of such data spaces include, among others: discussion forums, blogs, blogrolls/ feed subscriptions, mailing lists, shared bookmarks and image galleries.

#### 2.6.2.5. *SIMILE*

Semantic Interoperability of Metadata and Information in unLike Environments

SIMILE was a joint project, conducted by the MIT Libraries and MIT CSAIL and funded by the Mellon Foundation, which sought to enhance interoperability among digital assets, schemata/vocabularies/ontologies, meta data, and services. With completion of the project, many of its tools were open sourced and spun out to simile-widgets, a community-managed site.

#### 2.6.2.6. *NextBio*

A database consolidating high-throughput life sciences experimental data tagged and connected via biomedical ontologies. Nextbio is accessible via a search engine interface. Researchers can contribute their findings for incorporation to the database. The database currently supports gene or protein expression data and sequence centric data and is steadily expanding to support other biological data types.

#### 2.6.2.7. *ANTOM*

ANTOM automates the categorization of text documents, and enables the retrieval of information by semantic search. Its name is an acronym derived from "Automated Ontology Manager".

#### 2.6.2.8. *Linking Open Data*

The Linking Open Data project is a W3C-led effort to create openly accessible, and interlinked, RDF Data on the Web. The data in question takes the form of RDF Data Sets drawn from a broad collection of data sources. There is a focus on the Linked Data style of publishing RDF on the Web. It will be discussed in detail in later section in this chapter.

### 2.6.3. Semantic Databases

Chen explicitly contrasts Entity-Relationship diagrams with record modeling techniques of software engineering. He described ER Diagram is the more natural view in a way that describes the real world entities and relationships with some of the important semantic information while structure diagram is a representation only of the organization of records without representation of entities and relationships (Chen, 1975). Many other famous research publication support this argument of concept based data structuring and relations like Kent in "*Data and Reality*" 1998<sup>28</sup>, "*Data Semantics*" by Abrial (Abrial, 1974), Ronald Stamper (Stamper, et al., 1994) , Elmasri, Navathe (Elmasri, 1989), and Michael Jackson (Jackson, 1980). A semantic model is a model based on concepts thus considered as a "platform independent model". Semantic model provide semantic structures and relationships of

---

<sup>28</sup> <http://www.bkent.net/Doc/darxrp.htm>

entities. A semantic web data space contains domain specific portable data provided in human and/or machine friendly structures. Data in a data space can be referenced by an identifier, and is linked with other data across spaces and domains, and thus can be viewed in an object-Oriented fashion. This approach is applicable to Web based systems and also to desktop-based systems.

There is also a not only storing but more important part is data retrieval in semantic fashion. There are some very effective query languages like SPARQL which can provide facility to fetch semantic data securing semantic information about data.

#### 2.6.4. Semantic web data spaces, linked data, and data portability

Semantic Web was required to enhance the data portability, sharing and most important better understanding of available data for human and machine to search in a huge data space of web. Linked data project has done core task for data sharing and availability with its semantic details from different linked knowledgebase. This has the benefit of being a useful point for querying about information across domains, and assists the development of a Web of Data. Any data-space on web should support for data portability by providing an object in a data space should be movable and should also have the ability to be referenced using a standard identifier such as a Uniform Resource Identifier (URI).

##### 2.6.4.1. *Linked Data*

Linked data (also called Linked Open Data - LoD)<sup>29</sup> provide a mechanism to publish data which can be linked to the Link data cloud for sharing. At early stages of LoD was like Figure 2.4 with a very few nodes in the in LoD with few nodes which have grown too rapidly to Figure 2.5 with more than 31 billion triples according to the statistics of September 2011<sup>30</sup>.

---

<sup>29</sup> <http://linkeddata.org/>

<sup>30</sup> <http://www4.wiwiss.fu-berlin.de/lodcloud/state/>



was quite old and was based on the concept of the network model database, citations between scholarly articles, and authority control in libraries. Two major concepts of Linked Data are URI's to identify any resource available at data and second in RDF can be called the language of Linked Data for communication and understanding among nodes about nodes and data. Tim Berners-Lee outlined four principles of linked data in his Design Issues. Linked Data note, paraphrased along the following lines:

- 1 Use URIs to identify things.
- 2 Use HTTP URIs so that these things can be referred to and looked up ("dereferenced") by people and user agents.
- 3 Provide useful information about the thing when it's URI is dereferenced, using standard formats such as RDF/XML.
- 4 Include links to other, related URIs in the exposed data to improve discovery of other related information on the Web.

## **2.7. Ontology Driven Information System**

Ontology and Semantic Framework has change a lot on the Web. Still there is much research work to do in the favour of adopting the same idea in Software Engineering. At the introduction to the Second International Conference on Formal Ontology and Information Systems (FOIS) Smith presents a brief history of ontology as a discipline spanning the boundaries of philosophy and information science (Smith, et al., 2001). He predicted the future collaboration between philosophical ontologists and information scientists for making computational model of real world based on philosophical model specified in ontologies.

Later on, from the same platform in 2008 Uschold also supported the idea and provide support Ontology Driven Information System with ontological presentation and modeling of concept as a base model for software development (Uschold, 2008). Thus central idea for improving software engineering is to increase the level of formal methods and structure in various phases of the software lifecycle. Using formal ontologies to the models the concepts and areal world facts, in a model-driven software development represents the joining together of these two field with their original cores and functions, like Formal Ontology for reality representation in formalized way that can be directly instantiated at computer for an equivalent computational model.

# Graphical User Interfaces

## 3.1. Introduction

A User Interface is a term used for a technical solution providing a space for Human machine interaction. In computer science, It refers to the software that enables the computer machine interact with the user though some specific User Interface devices. These computer user interfaces can be a command based user interface or a graphical user interface.

A command base user interface interacts with the user in text messages: takes a command for some computation in text stream and/or show results in text if required.

A graphical user interface (GUI) is a type of user interface that allows users to interact with electronic devices graphical representation of entities represented at interface. GUIs can be used in computers, hand-held devices like mobile-phones, portable media players or gaming devices, digital household appliances and office equipment. A GUI symbolizes the information and user's possible actions to a user through graphical icons and visual indicators such as menus. It provides direct manipulation of the actions, which is usually performed at the graphical elements presented at user interface.

### 3.1.1. Human-Computer Interaction

ACM SIGCHI has defined Human-Computer Interaction (HCI) as “*Human-computer interaction is a discipline concerned with the design, evaluation and implementation of interactive computing systems for human use and with the study of major phenomena surrounding them*”. A User Interface (UI) can have very diverse applications like mechanical devices user interface. As a domain of research we will discuss UI or GUI in the domain of HCI.

## 3.2. History

Historically, the term GUI is refers to the scope of two-dimensional display screens with display resolutions able to visualize generic information, in the tradition of the computer science research in late 70's at the PARC (Palo Alto Research Center). We keep the boundaries of User Interface scope only to the computing field.

### 3.2.1. Precursors

User Interface as a mandatory component of any electronic device especially with computer will always be required and available in different forms to make use of the machine by human. In our

research we consider about the computer user interfaces precisely. Computer User Interface was made with the first computation machine to communicate by human (to take commands and data and give back calculated results). Earliest ages of computer, all of human computer communication was done in Batch systems, which later on improved to a command lines interface and the last major change in base model of computer user interface is GUI. There are many changes and no doubt improvements and technological level but not in the base concept of user interface. Many fast and highly sophisticated devices are invented to get the images of natural quality in a GUI. With so much changes and rapid work in the field of User Interface devices and implementation methodology that even GUI design decisions got totally dependent to the technology.

### 3.2.1.1. *Batch Computing*

The starting decades of computer history is known as the batch era, when computing power was extremely inadequate and expensive. The human machine communication used to be done in batches, where a computational sequences (software programs) and data has to be loaded and executed in batches. Due to very slow manipulation for different phases to make use of computers like program loading, inking and execution, it was not feasible to perform these tasks separately for each computer program. Programs written in one language were loaded and linked in a batch for processing. Users had to accommodate computers rather than the other way around; user interfaces were considered overhead, and software was designed to keep the processor at maximum utilization with as little overhead as possible. (Raymond, et al., 2004)



Figure 3.1: IBM 029 card punch<sup>31</sup>

As programs were written through tapes and later through punch cards, these inputs were prepared separately to work in batch process on punched cards or equivalent media like paper tape. The output side added line printers to these media. Thus there was no direct interaction of computers with the machines in real time. There was not any option to make any changes in input programs and data once it is loaded at computer.

### 3.2.1.2. Command Line Interface

Command-line interfaces (CLIs) introduced direct interaction of users to the computers. It provides and environment with batch monitors connected to the system console. Real time request-response transactions were modeled in CLI, where requests are expressed as textual commands from a specialized vocabulary and syntax. Computer from this ages were far efficient than of batch systems, increasing from days or hours to seconds. Computation costs were also much decreased in comparison to the batch systems. Command-line systems also allowed the user to alter the later stages of the transactions which are not executed yet in response to real-time or somewhat-real-time feedback according to the initial results. Investigative and interactive software execution made real-time interaction also debugging possible. But these interfaces still require a serious investment of effort and learning time to master over the commands and their syntax.

```
Current date is Tue 1-01-1980
Enter new date:
Current time is 7:48:27.13
Enter new time:

The IBM Personal Computer DOS
Version 1.10 (C)Copyright IBM Corp 1981, 1982

A>dir/w
COMMAND COM   FORMAT COM   CHKDSK COM   SYS COM   DISKCOPY COM
DISKCOMP COM  COMP COM   EXE2BIN EXE   MODE COM   EDLIN COM
DEBUG COM    LINK EXE   BASIC COM   BASICA COM  ART BAS
SAMPLES BAS  MORTGAGE BAS  COLORBAR BAS  CALENDAR BAS  MUSIC BAS
DONKEY BAS   CIRCLE BAS   PIECHART BAS  SPACE BAS   BALL BAS
COMM BAS

                26 File(s)
A>dir command.com
COMMAND COM    4959  5-07-82  12:00p
                1 File(s)
A>
```

Figure 3.2: Screenshot MS Dos

---

<sup>31</sup> <http://www.columbia.edu/cu/computinghistory/129.html>



Command-line interfaces were closely associated with the rise of timesharing computers, the concept introduced in the 1950s. The most significant early experiment was the MULTICS operating system after 1965 and by far the most influential of present-day command-line interfaces is that of UNIX systems that was introduced in 1969. The earliest command-line systems were a combination of teletypes with computers, adapting a mature technology that had proven effective for mediating the transfer of information over wires between human beings. Using of teletypes reuses the typing pattern introduced for normal typing machines.

### 3.2.1.3. Precursor to GUI

Douglas Engelbart, as a lead of researchers at the Stanford Research Institute, designed a precursor to GUIs that uses text-based hyperlinks manipulated with a mouse for an On-Line System and got it patent (Engelbart, 1970). The concept of hyperlinks was further refined and extended to graphics by researchers at Xerox PARC<sup>32</sup>, who are considered as the first Graphical-based hyperlinks and inventors of first GUI.

## 3.2.2. PARC user interface



*Figure 3.3: Xerox Star Workstation*

The PARC user interfaces are considered as the most influential achievement in the history of User Interfaces by invention of a graphical user interface. This user interface was consisted of graphical

---

<sup>32</sup> <http://www.parc.com>

elements like windows, menus, radio buttons, check boxes and icons with a pointing device in addition to a keyboard for user interaction with the computer.

Computer as part of business and official processing tool, these graphical objects were and inspiration of the office objects like files, folders, documents etc. These aspects can be emphasized by using the alternative acronym WIMP, which stands for *windows, icons, menus* and *pointing device*. (Raymond, et al., 2004)

### 3.2.2.1. Evolution

As a result of PARC efforts in the field of Computer User Interface in 1981, the Xerox 8010 Star Information System was invented as the first GUI-centric computer operating model. Apple get in competition soon with the Apple Lisa (which presented the concept of menu bar as well as window controls) in 1983 and the Apple Macintosh 128K in 1984, followed by the Atari ST and Commodore Amiga in 1985 (Johnson, et al., 1989).

Today, the most familiar GUIs to people can be list Microsoft Windows, Mac OS X, and X Window System interfaces for desktop and laptop computers, and Symbian, BlackBerry OS, Android and Apple's iOS for handheld ("smartphone") devices.

Xerox's successfully experimented idea for GUI has been widely used by many other companies like Apple, IBM and Microsoft to develop their products. Later on, IBM's Common User Access specifications (Berry, 1988) produced the basis for the graphical user interface that guided the construction of Microsoft Windows, IBM OS/2Presentation Manager, and the UNIX Motif window manager and toolkit. These ideas evolved to create the interface found in current versions of GUI for many operating systems and various desktop environments. Thus majority of the current GUIs are based on same principles and share largely common idioms.

## 3.3. Structural elements

Different visual conventions mostly from daily office environment are used and integrated to form a GUI for the generic information representation. These images of daily conventions work as elements of graphical user interfaces that build the structure of the static elements on which the user can interact, and define the appearance of the interface. Our research is using the same conventional models to represent the ontological and semantic structures for a specific knowledge domain. Here is a list of common graphical elements used in different graphical libraries like OpenGL (Woo, et al., 1999), QT (Eng, 1996), wxWidgets (Leijen, 2004), Java Swing (Walrath, et al., 2004) etc. Figure 3.4 shows an example GUI structuring some of the elements in a printer properties display dialogue window.

### 3.3.1. Window

A window is an area on the display screen that represents information as complete structure of GUI or complete GUI. Mostly it refers to a representation of specific part of GUI with its contents independent from the rest of the screen. An example of a windows environment, it is what appears on the screen when the "My Documents" icon is clicked in the Windows Operating System. It is an easy to manipulate window for a users with some conventional behavior of dragging, minimize, maximize and cascading with some special common elements like title bar for showing the title of windows and also work as a holder for dragging the window, status bar shows the current status of window like any selected element or proceeding and a set of three buttons to minimize maximize and closing the window.

#### 3.3.1.1. *Container Window*

A Container Window specifically used to visualize the directory structure and data files and executable programs available in the specific locations. These are container windows are invoked through the icon of root or any element of directory structure which can contain other elements like dick storage, directory (folder). It provides an ordered list of other icons that could be again some other icons of new container windows or data in files or links to some other place even executable programs. All modern container windows could present their content on screen either acting as browser windows or text windows discussed later. Their behavior can be personalized in visualizing the automatically according to the choices of the users and their preferred approach to the graphical user interface.

#### 3.3.1.2. *Browser Window*

A browser window allows the user to move forward and backwards through a sequence of documents or web pages. Web browsers are an example of these types of windows invented by Sir Tim Berners-Lee<sup>33</sup>.

#### 3.3.1.3. *Text Terminals*

Text terminal windows are designed for embedding command line interface to the GUI. MS-DOS and UNIX consoles are examples of these types of windows.

#### 3.3.1.4. *Parent Child Window*

A child window opens automatically or as a result of a user activity in a parent window. Pop-up windows on the web browsers can be considered as an example of specific kind of child windows.

---

<sup>33</sup> <http://www.w3.org/People/Berners-Lee/WorldWideWeb.html>

### 3.3.1.5. *Message Window*

A message window, or dialog box, is a type of child window. These are usually small and basic windows that are opened by a program or another window to communicate with the user by displaying information to the user and/or getting information from the user. These windows usually contain at least one push button to resume the parent program or window.

### 3.3.2. Tabs

A tab group is also a list of container that contains other graphical elements. It is typically use to classify the data in different tabs (also called tab-pages) visualized mostly as rectangular small box in a tab group. Many of the dialogue windows showing multiple data options and input from user are group in some categories use tab groups e.g. of When activated the view pane, or window, displays widgets associated with that tab; groups of tabs allow the user to switch quickly between different widgets. This is used in the web browsers Firefox, Internet Explorer, Konqueror, Opera, Safari and chrome. With these browsers, you can have multiple web pages open at once in one window, and quickly navigate between them by clicking on the tabs associated with the pages. Tabs are usually placed in groups at the top of a window, but may also be grouped on the side or bottom of a window. Tabs are also present in the settings panes of many applications. Windows for example uses tabs in most of its control panel dialogues.

### 3.3.3. Menus

To reduce the learning load for users, menus are introduced that allow the user to execute commands by selecting from a list of choices. Options can be selected using a mouse or any other pointing device within a GUI. A keyboard may also be used by provided shortcuts for expert users. Menus are convenient because they show what commands are available within the software.

Conventionally a parent menu bar is displayed horizontally across the top of the screen and/or along the tops of some or all windows. This menu bar is like list of categories of available commands. Sub menu can be pull down by clicking any category to see list of commands or a sub category of the menu. These menus are named as pull down menus.

A popup menu is usually associated with right click button of the mouse that can also provide some other context dependent action where ever on the screen it is called, thus also called context menu. This menu is invisible until the user activate it by right clock or if available keyboard button according to current position of pointer/cursor.

### 3.3.4. Icons

An icon is a small graphical representation of available objects such as a file, program, web page, or command. These are also a quick way to execute commands, open documents, and run available

programs. Icons are also very useful when searching for an object in a browser list, because in many operating systems all documents using the same extension will have the same icon.

### 3.3.5. Controls (or Widgets)

A computer user interacts through different controls available in GUI usually refers to Widget. These interface elements have their specific behavior (functionality) and interaction method (use actions e.g. click, double click, drag and move etc.)

#### 3.3.5.1. *Window*

A paper-like rectangle that represents a "window" into a document, form, or design area.

#### 3.3.5.2. *Button*

A button or a push button is a control to perform some command that can be triggered by user. It is a convention of a mechanical or electronic instrument's push-button.

#### 3.3.5.3. *Pointer (or mouse cursor)*

The spot indicating the current referred position of the pointing device is called the pointer like mouse pointer or cursor. Conventional graphics are used to provide not only information but current status of the system like showing an hourglass for the busy (or wait) state or blinking cursor at any document editor showing ready state.

#### 3.3.5.4. *Single Element Text Data Control*

##### TEXT BOX

A text box is a text editing control to add/get textual box information from/to the user.

##### HYPERLINK

Text with some kind of indicator (usually underlining and/or color) that indicates that clicking it will take one to another screen or page. URL is a hyperlink for a web page or web content.

##### LABEL

Label is a non-editable text control only to show some textual data to the user

#### 3.3.5.5. *List Items*

List items are used to present a list of elements or data items. It can also be used to get some selection(s) by the user from a given list of elements.

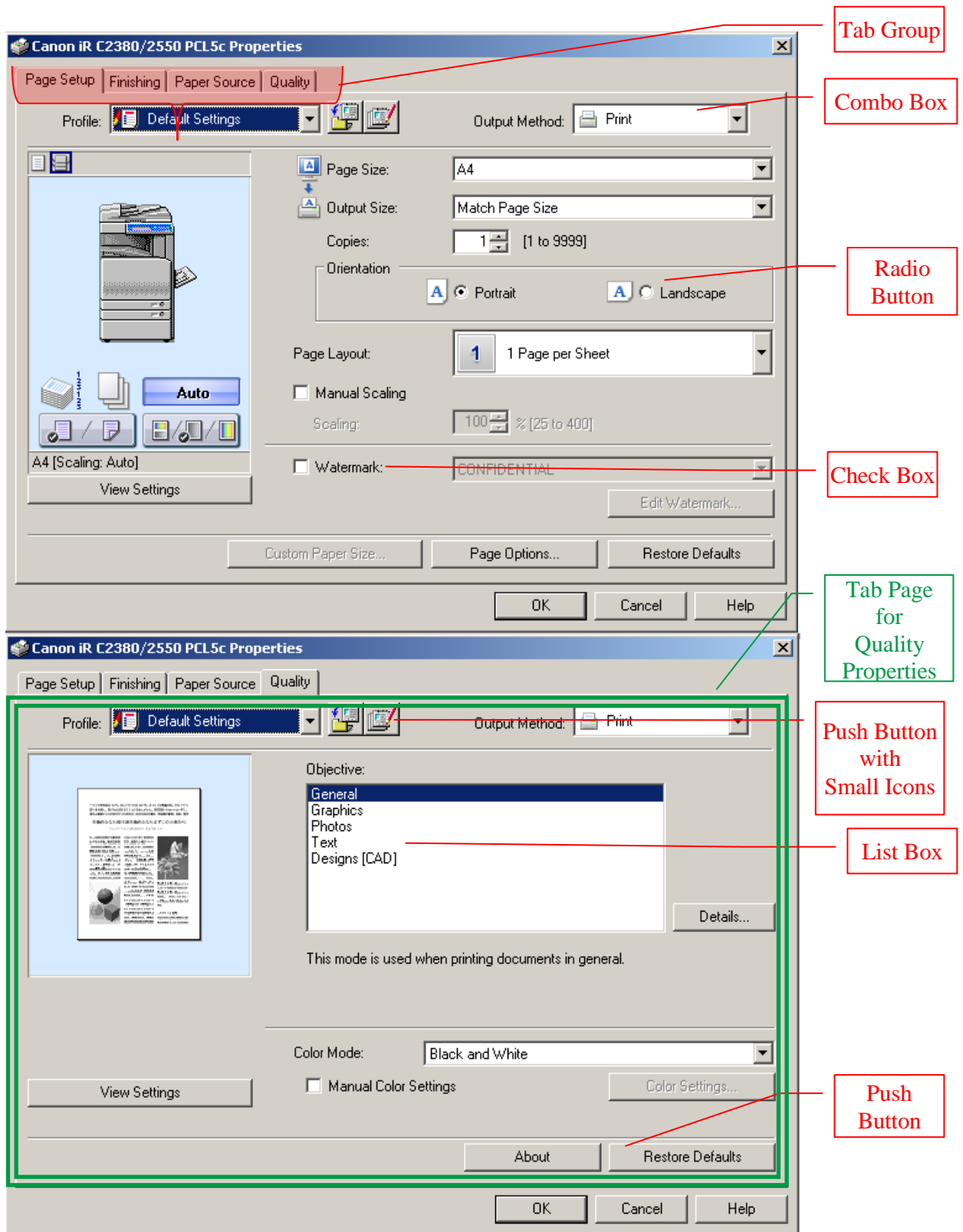


Figure 3.4: An Example of GUI structure for Printer Properties for MS Office

## LIST BOX

A list box provides an option to the user of selection from the given list of elements. User can select more than one item from the list. Some of the dialogue windows also work as list boxes with a list of files and folder to select to perform a specified operation using this GUI control.

## DROP-DOWN LIST

A drop down list provides a list of items for a user selection. The list normally only displays items when a special button or indicator is clicked. A Drop down or popup menu is also an example of drop down list.

## COMBO BOX

A combo box is a combination of a drop-down list and a single-line textbox usually for single element selection from the list. It also allows user to type a value directly into the control from the list of available options.

### *3.3.5.6. Datagrid*

A spreadsheet-like grid, called Datagrid, allows numbers or text to be entered in rows and columns. It is used to visualize any tabular form. It can also be considered as a list of data or records where conventionally each row grid present a record/like a tuple of relational database.

### *3.3.5.7. Option Buttons*

Option buttons are use for a binary input to present the state of “on” or “off” of available options. It is mostly triggered by clicking available option

## RADIO BUTTON

A radio button is an option button that allows user to select one of the available options. It is named over the mechanical push-button group on a car radio receiver. Selecting a new item from the group's buttons also deselects the previously selected button.

## CHECK BOX

Check boxes are used to facilitate user for multiple selection form the given options. Conventionally, it is visualized through check mark  or a cross  or empty box thus called check box.

## 3.4. Interaction elements

Data structuring elements of GUI are used to receive and send information from or to user. There are still necessary elements used for indications about the current state and position of GUI. While Human Computer Interaction at GUI the currently focused of element must be known or indicated to the user. In contrast to the command line interface, GUI provides and assists users by providing only possible actions at specific location at screen at some specific focused, selected or pointed element of GUI.

### 3.4.1. Cursor

A cursor is a visual indicator about the current position on a computer monitor or other display device. Cursor was also used in command line interface to show the current position within the text to edit by a blinking “\_” (called underscore). Cursor in GUI refers to indicator in the both environments whether it input from a text input like keyboard or a pointing device like mouse. At GUI, current position of the cursor is mostly different from the mouse pointer at desktop or text editing environment, where cursor can be shown at some focused graphical element or an I-beam in a text editor.

### 3.4.2. Pointer

One of the most common indicators provided at GUI on the any personal computer is a pointer. It a graphical image indicates the current position of the pointing device. There are many user interaction operations associated with the pointing device through the pointer as a combination a click and move like click, double click, select, drag and drop etc. A pointer commonly visualized as an angled arrow, but it can vary within different programs or operating systems. As an example it can be an I-beam pointer within text-processing applications, or a gloved hand with outstretched index finger at any hyperlink mostly in web browsers. Thus it also provides an indication about the currently selected or pointed data element. It also provides an indication of the busy state of the machine showing some hourglass or a wristwatch.

### 3.4.3. Selection

A selection is a list of items on which user operations will take place. The user typically adds items to the list manually, although the computer may create a selection automatically. The selected items are usually grayed in convention of the grayscale displays.



## 3.5. Post-WIMP GUI

The reason WIMP environment was a big breakthrough that was very much visible at user interface screen comparative to command line interface. They also make use of an analogous paradigm of documents as paper sheets or folders from the a normal office environment than a computational model by abstracting workspaces, documents, and their actions, thus had easy to introduction to novice users and much attraction from the end users. Moreover, the basic representations as rectangular regions on a 2D screen make them a good fit for system programmers, thus support the profusion of commercial widget toolkits in this style.

However, early WIMP interfaces were not at their best to provide most favorable capacity for working with complex tasks such as computer-aided design, working on large amounts of data simultaneously, or interactive games. Applications for which WIMP is not well suited include those requiring continuous input signals, showing 3D models, or simply portraying an interaction for which there is no defined standard widget. It was required to make necessary improvements in WIMP environment by development of customized interface according to the application requirements. Interfaces based on these considerations, now called "post-WIMP", have made their way to the general public. Examples include the interface of the classic MP3 player iPod and a bank's automated teller machine screen.

Jakob Nielsen (Nielsen, 1999) proposed post-WIMP interfaces in "Non Command User Interfaces" followed by "The Anti-Mac Interface" in 1993. In 1997, Andries van Dam (Van Dam, 1997) argued about updated proposals that are discussed in "Post-WIMP user interfaces". Later on a framework proposed by Michel Beaudouin-Lafon (Beaudouin-Lafon, 2000) in 2000 called instrumental interaction. This framework proposed a design a design space for Post-WIMP interaction techniques and a set of properties for comparing them, such as reality-based interaction and 3D interaction.

## 3.6. User Interface Iceberg Analogy

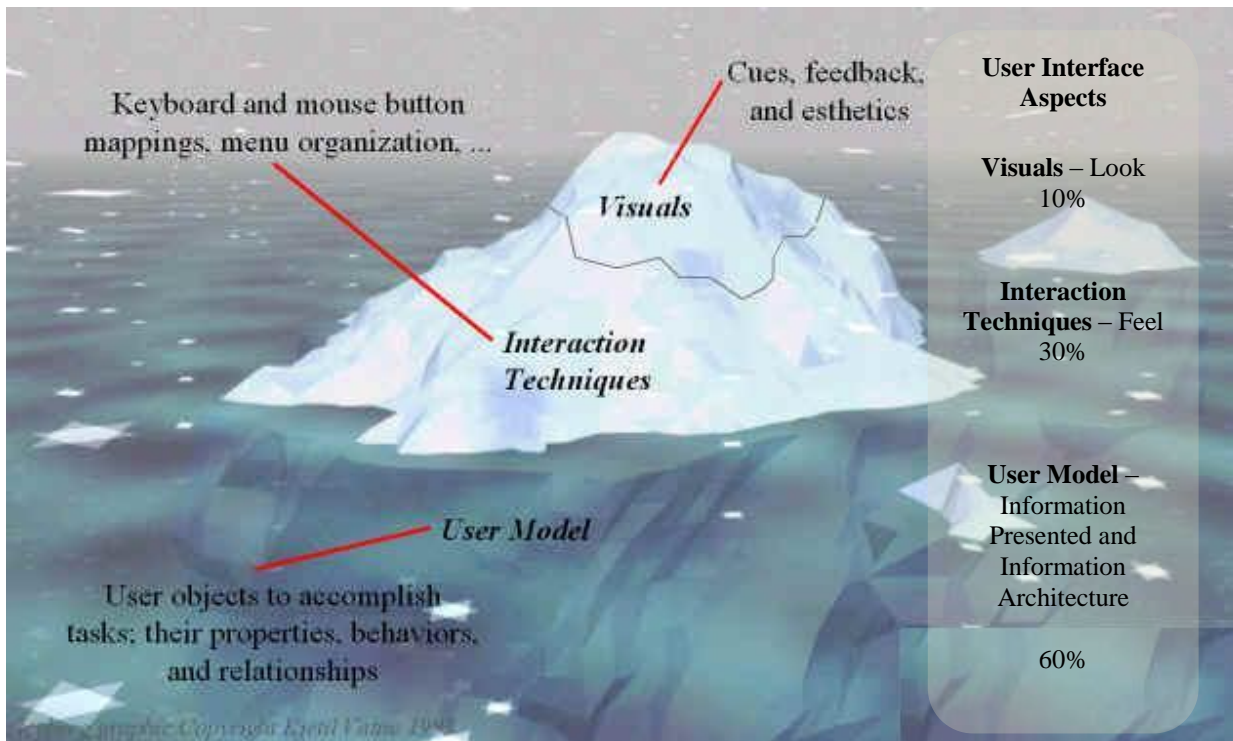


Figure 3.5: The iceberg analogy of usability by Dick Berry

Dick Berry has argued about the important hidden part of a user interface through the iceberg analogy of user experience of a computer application through a user interface<sup>34</sup>. He defined user interface as an iceberg with a smaller visible part of 40% of look and feel and 60% is of user model presenting the information including information architecture. This model also argues about the importance of usability and user understanding depending more on the user model that consists of interaction objects and their structure at presented GUI. We have used this model to identify the three layers of user interface and their weights and consideration. Current approaches and research area for rapid user interface development tools and methodologies are mostly focusing on the technical details and improvements at look and feel of the GUI while ignoring the larger part that need to be considered for information architecture and presentation tools. Many of the critics at modern development have pointed out the challenge to make GUI using conceptual model to make a user model (Puerta, 1996) (Szekely, 1996) (Alexander, et al., 2003) . We have adopted the analogy to show the importance of semantic representation of the domain concept at user model that can truly reflect the domain concept

<sup>34</sup> <http://www.ibm.com/developerworks/library/w-berry/>

at user interface. Any Post WIMP software development tool that allows automatic GUI generation must carry the properties and constraints defined at the concept level to the user model. Moreover for any model driven software development also require a conceptual model mapping to the use model to ensure the concept delivery at user interface.

### 3.7. Element of User Experience

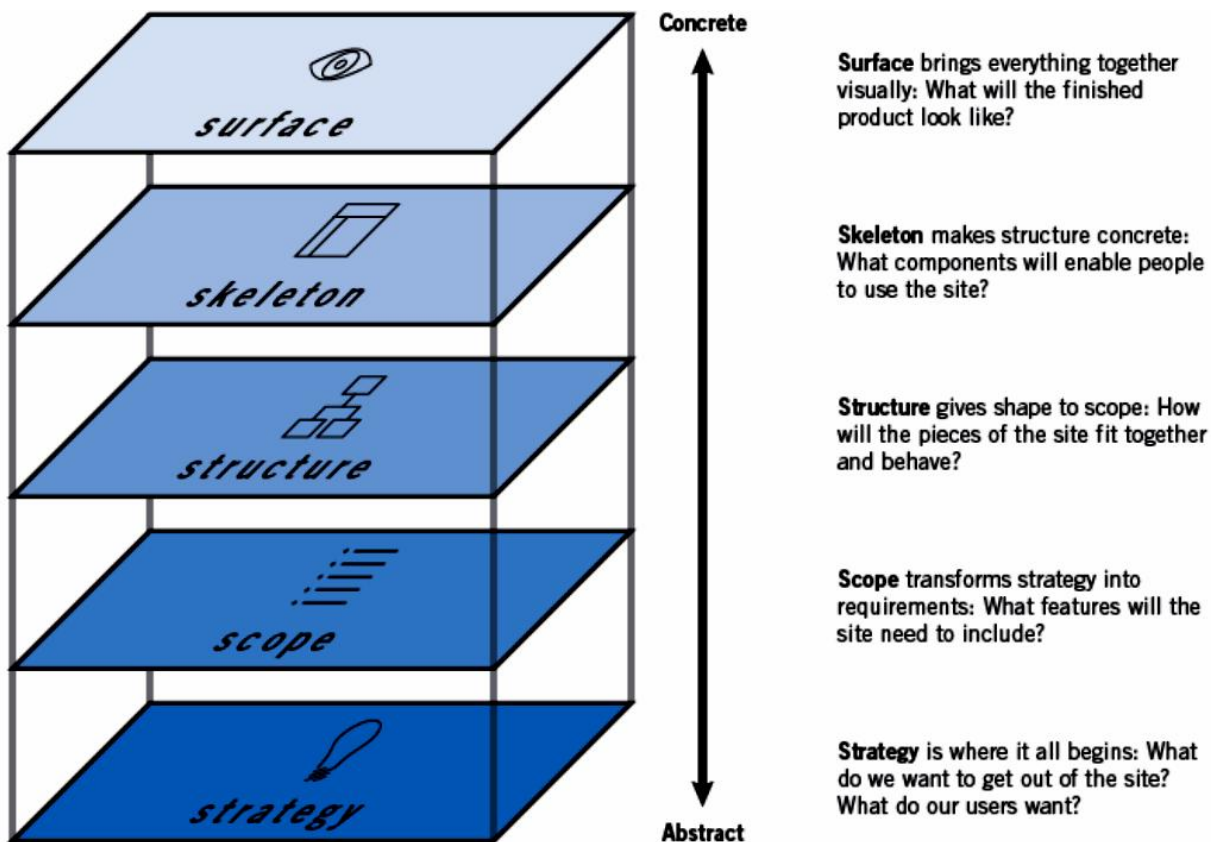


Figure 3.6: Element of user Experience by David Garret (Garrett, 2002)

We have used arguments from another famous analogy of user experience model by David Garret. Garrett has mentioned five elements of user experience of information from abstract level to concrete structure represented at computer through a software interface or a website. These elements collectively build an image of the concept presented at computer (Garrett, 2002). With the reference of Garrett’s model, we argue that representation and direct mapping of ontology from the base layer (i.e. strategy) to the presentation layer (i.e. surface) can help user to perceive the targeted concept. These elements can also be considered as layers of UI design and implementation. User perceives following elements:

1. Strategy is an abstract and non-formalized picture of concept needed to represent a computer model. This computer model can be software or website for providing and/or processing information to achieve specific goals.
2. Scope provides the features and functions included. It provides the boundary of the software or website. The question of whether a feature is within the boundaries or not.
3. Structure talks about the way in which these features fit together.
4. Skeleton gives the representation of structured function and features. It provides the placement and use of buttons, tabs, textboxes, labels, etc. at UI.
5. Surface provides the concrete GUI through a structure of Web pages or widgets representing the structured contents to the user.

## 3.8. User Interface Design

Graphical User Interface design or Graphical User Interface engineering refers to the process of designing an interface element presenting information and communication function with visual tool for a software application represented at computer or any other electronic device. It needs a complex translation to cater the differences in languages and behavior of human and machine. User Interface designing is the core part of our research for designing user interface model through domain ontologies.

### 3.8.1. Research – past and ongoing

Graphical User interface design has been a topic of substantial research, including its aesthetics. In the past standards have been developed, as far back as the eighties for defining the usability of software products. There are many heuristics based experiments provided many user interface designing guidelines like Gnome<sup>35</sup>, KDE<sup>36</sup>, android developers<sup>37</sup>, Apple Mac OS X<sup>38</sup> and Java Look & Feel Design Guidelines (Sun Microsystems, Inc., 2011) and many research publications like Mayhew (Mayhew, 1992). One of the structural basis has become the IFIP reference model by IFIP Working

---

<sup>35</sup> <http://developer.gnome.org/hig-book/stable/>

<sup>36</sup> <http://techbase.kde.org/Projects/Usability/HIG>

<sup>37</sup> [http://developer.android.com/guide/practices/ui\\_guidelines/index.html](http://developer.android.com/guide/practices/ui_guidelines/index.html)

<sup>38</sup>

<http://developer.apple.com/library/mac/#documentation/UserExperience/Conceptual/AppleHIGuidelines/Intro/Intro.html>

Group (2.7 / 13.4) for user Interface engineering<sup>39</sup>. The ifip model proposes four dimensions to structure the user interface:

- 1 The input/output dimension (the look)
- 2 The dialogue dimension (the feel)
- 3 The technical or functional dimension (the access to tools and services)
- 4 The organizational dimension (the communication and co-operation support)

This model has appreciably influenced the development of the international standard (ISO 9241) stating the interface design requirements including “*ISO 9241-210:2010 provides requirements and recommendations for human-centred design principles and activities throughout the life cycle of computer-based interactive systems*”<sup>40</sup>. In early 80’s, majority of user interface design guidelines had objective to provide a generic solution for all knowledge domain related computer application and users, thus were not very clearly defined in details for every kind of application and information representation.

With all efforts from researchers and technology development, User Interface design process became much independent to the technology. Many researchers and critics to the user interface standards and guidelines have pointed out the issue of technology dependence of user interface designing and especially development due to financial, technical feasibility (Bastien, et al., 1995) (Reeda, et al., 1999) (Dzida, 1995). The diversity in knowledge domain, data formats and user requirements with some commercial bindings bring many constraints to the environment which makes many user interface guidelines impractical in use. These requirements led to domain specific user interface design approaches.

#### 3.8.1.1. Domain Specific GUI Designing

To achieve better usability, the context knowledge, including the domain of the software application and targeted user, based user interface designing got much attention (Bauersfeld, et al., 1991) (Carlsen, 1992). Context analysis became an unavoidable part of system analysis for designing and development of a user interface (Thomas, et al., 1996).

The aim to understand domain-specific GUI concerns during software development process, introduced the requirement to explore GUI rapid prototyping tools that might present some convincing simulations to the user of how an actual application might behave in production use. Some of the research work has shown that a wide variety of programming tasks for GUI-based software can, in fact, be specified through means other than writing program code. There are also some adaptive and

---

<sup>39</sup> <http://www.se-hci.org/index.html>

<sup>40</sup> [http://www.iso.org/iso/iso\\_catalogue/catalogue\\_tc/catalogue\\_detail.htm?csnumber=52075](http://www.iso.org/iso/iso_catalogue/catalogue_tc/catalogue_detail.htm?csnumber=52075)

multimodal interfaces that allows user to switch to different modes according to their needs. There is also research on generating user interfaces automatically, to match a user's level of ability for different kinds of interaction (Mayhew, 1999).

Domain Specific GUI designing process include a detailed context analysis which can be divided into three subtasks as Task analysis and User analysis and Environment detail Analysis. (Thomas, et al., 1996)

#### TASK ANALYSIS

Task analysis refers to the data collection and analysis about the functions needed to be performed by the computer system.

#### USER ANALYSIS

It the analysis about the user abilities, job description, authorities and organizational role to design the most suitable interface according to the user's context

#### ENVIRONMENT ANALYSIS

In process of user interface development, an environment analysis refers to the study of the facilities, abilities and constraints of following:

- 1 Organizational Environment
- 2 Technical Environment
- 3 Physical Environment
- 4 Control options

### 3.8.2. User Centered Interface design

User-centered design (UCD) or pervasive usability<sup>41</sup> is a design a process and a philosophy that is based on the requirements, desires, and limitations of end users of the information system. These elements are given attention across-the-board at each stage of the application specifically user interface design process.

Norman (Norman, 1988) defined user-centered design as "*a philosophy based on the needs and interests of the user, with an emphasis on making products usable and understandable*" (p. 188). He defined usability and understanding measures of the software in user understanding with two aspects

- 1 The user can figure out what to do
- 2 The user can tell what is going on.

---

<sup>41</sup> <http://www.sitepoint.com/planning-uncertain-future/>

Rubin (Rubin, 1994) described user-centered design as techniques and procedures for designing usable systems keeping the user as the focal point of the designing process. Many other researchers agree that principles of user-centered design philosophy or methodology to create more usable and useful products by focusing on the user throughout the design process (Dumas, et al., 1993) (Eason, 1988) (Gould, et al., 1985) (Shackel, 1991). Gould and Lewis (Gould, et al., 1985) described three principles which were extended to four by Gould (Gould, 1995) as stated:

- 1 EARLY - CONTINUAL - FOCUS ON USERS
- 2 Direct contact: through interviews, observations, surveys, participative
- 3 Design: To understand cognitive, behavioral, attitudinal, Anthropometric characteristics of users and their jobs.
- 4 EARLY - CONTINUAL - USER TESTING
- 5 Early on, intended users do real work with simulations and prototypes; their performance and reactions are measured qualitatively and quantitatively.
- 6 ITERATIVE DESIGN
- 7 System (functions, user interface, help system, reading material and training approach) is modified based upon results of user testing.
- 8 Testing cycle is repeated.
- 9 INTEGRATED DESIGN
- 10 All aspects of usability evolve in parallel
- 11 All aspects of usability under one focus

User-centered design methodology is defined as a multi-stage process iterative process that not only requires designers to analyze and anticipate usage of the product by user, but also to validated of their assumptions according to the user behavior in real world tests with actual users. This iterative design methodology is also refers to the Spiral model for software engineering (Boehm, 1986).

### *3.8.2.1. Cognitive Dimensions*

Cognitive dimensions or Cognitive dimensions of notations are design principles for notations, user interfaces and programming language design, described by researchers Thomas R.G. Green and Marian Petre (Green, et al., 1996). The dimensions can be used to evaluate the usability of an existing information artifact, or as heuristics to guide the design of a new one. Cognitive dimensions are designed to provide a lightweight approach to analysis of a design quality, rather than an in-depth, detailed description. They provide a common vocabulary for discussing many factors in notation, UI or programming language design. Also, cognitive dimensions help in exploring the space of possible designs through design maneuvers.

### 3.8.3. User Interface Designs for Web

A *web browser* is a software application for retrieving, presenting, and traversing information resources on the World Wide Web. An information resource is identified by a Uniform Resource Identifier (URI) and may be a web page, image, video, or other piece of content. Thus, it enables users to access, retrieve and view documents and other resources on the Internet. Hyperlinks present in resources enable users easily to navigate their browsers to related resources. Although browsers are primarily intended to access the World Wide Web but they can access any data source from specified location e.g. Physical disk storage, data-store or a knowledgebase. In 1990, Sir Tim Berners-Lee invented the first web browser<sup>42</sup>. It was named first as *worldwideweb* later changed to *NexusA*. Firefox, Google Chrome, Internet Explorer, Opera, and Safari are most commonly used web browsers.

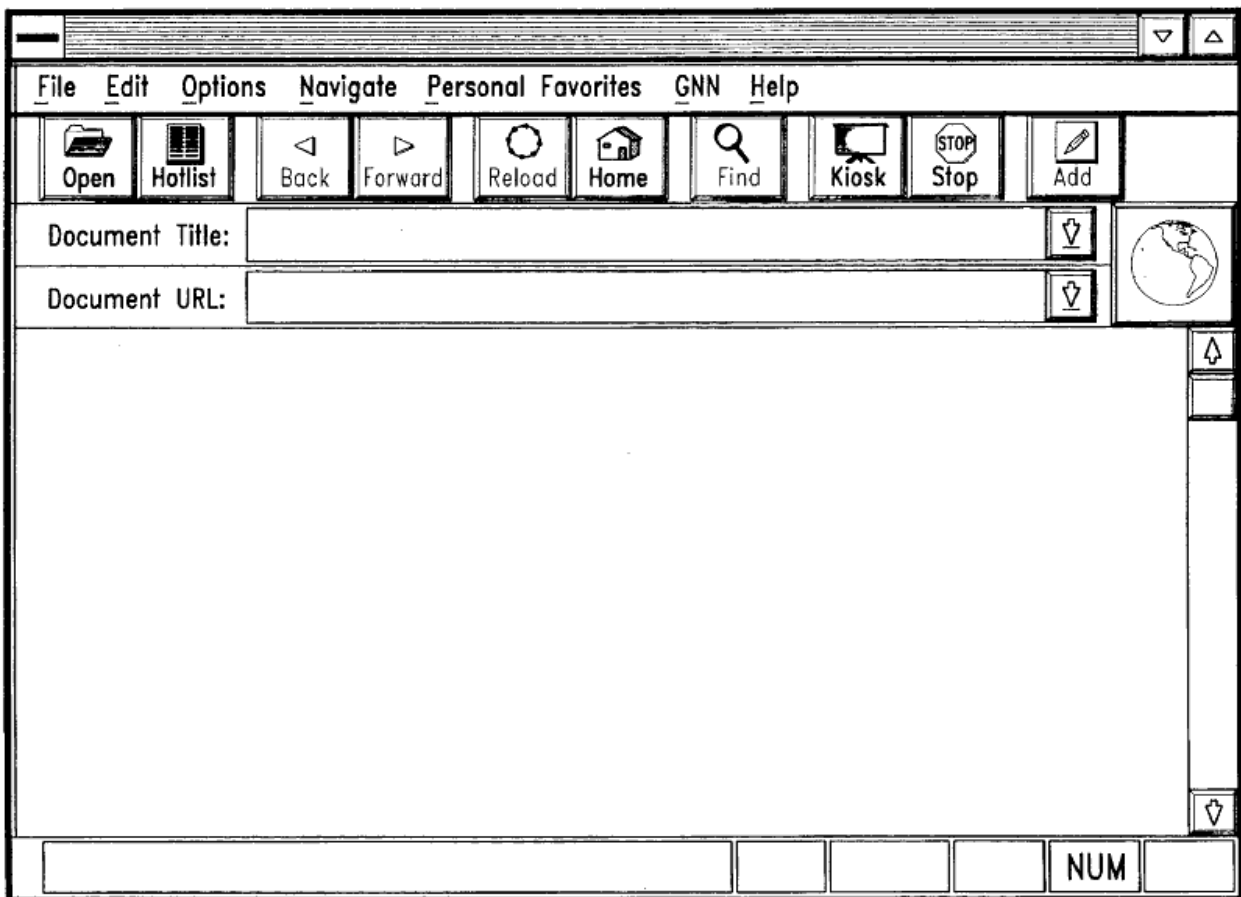


Figure 3.7: A Browser User Interface United States Patent

<sup>42</sup> <http://www.w3.org/People/Berners-Lee/WorldWideWeb.html>



These browsers have a specific environment for information visualization and interaction mechanism for any online or web application (Judson, 1996). Judson suggested standard user interface as the patent specific for a web browser for a dynamic display of information objects shown in Figure 3.1Figure 3.7.

These buttons for back, forward, reload and home and address bar (Document URL) are clearly visible nearly at same location of the user interface in all of the known web browsers.

Similar to other design guidelines for a user interface development, researchers have also specified some designing guideline for User Interface for web applications and web pages. Still these are not must rules due to diversity of information ad contents representation at web for a vast range of users (Corry, et al., 1997) or online search interface guidelines (Bates, 1989). Nielsen has identified five of the major issues regarding online application in 1999 (Nielsen, 1999). Two of the identified problems, data download speed and scrolling at web pages, are dependent at technologies which are very much resolved. Three important issues are presented content and its structure and searching methods. Last three are purely representation design issues that can be catered by an intensive context analysis and design procedure.

### 3.8.4. User Interface Design Processes

User Interface Design is a multiphase process including planning and data gathering and designing – testing iterations. This process also involves user analysis, system prototyping and prototype evaluation (Sommerville, et al., 1994).

#### 3.8.4.1. *Task Analysis*

Task analysis is a process of assembling the list of the required system functionality to accomplish the developing system objectives and fulfilling the potential user needs. This analysis also attempt to answer following questions:

- 1 What goals do users want to achieve by using the application and what set of user tasks is the application intended to support?
- 2 Which tasks are most important, and which ones are least important? and Which tasks are common, and which ones are rare?
- 3 What are the steps of each task and what are the result and output of each task?
- 4 What problems do users have performing each task? What sorts of mistakes are common? What causes them and how damaging are mistakes? (Sommerville, et al., 1994) (Kieffer, et al., 2010).

#### 3.8.4.2. *User and Context analysis*

User and Context analysis consist of information gathering and study about the potential users, their need and their technical abilities and facilities of the physical and organizational environment. system

either through discussion with people who work with the users and/or the potential users themselves. Typical questions involve:

- 1 What would the user want the system to do?
- 2 How would the system fit in with the user's normal workflow or daily activities?
- 3 How technically savvy is the user and what similar systems does the user already use?
- 4 What interface look & feel styles appeal to the user?

Information architecture – development of the process and/or information flow of the system (i.e. for phone tree systems, this would be an option tree flowchart and for web sites this would be a site flow that shows the hierarchy of the pages).

#### *3.8.4.3. Prototyping*

In other words, prototyping is a core activity in design across different domains as Moggridge (Moggridge, 2007) regards a prototype as “a representation of a design made before the final solution exists.” Lichter et al. (Lichter, et al., 1994) argued that Prototyping involves producing early working versions in the user interface design process of the future application system and experimenting with them. These prototypes are stripped of all look & feel elements and most content in order to concentrate on the interface.

#### *3.8.4.4. Usability testing*

While conducting usability testing, two major considerations have to be taken in to account. First thing to ensure is that the best possible method is used when participants interact with representative scenarios, such as quantitative data and qualitative observations information are provided. Secondly, to ensure that iterative approach is used. Usability testing is performed mostly at working prototypes. Testing of the prototypes on an actual user often using a technique called think aloud protocol where you ask the user to talk about their thoughts during the experience.

#### *3.8.4.5. Graphic Interface design*

Actual look & feel design of the final graphical user interface (GUI). It may be based on the findings developed during the usability testing if usability is unpredictable, or based on communication objectives and styles that would appeal to the user. In rare cases, the graphics may drive the prototyping, depending on the importance of visual form versus function. If the interface requires multiple skins, there may be multiple interface designs for one control panel, functional feature or widget. This phase is often a collaborative effort between a graphic designer and a user interface designer, or handled by one who is proficient in both disciplines.

User interface design requires a good understanding of user needs.

### 3.8.5. User Interface Design Requirements

The dynamic uniqueness or characteristics of a system are explained in terms of dialogue prerequisites that restrained in 7 principles of part 10 of the ergonomics standard - the ISO 9241. This system setup a framework of ergonomic "principles" for the dialogue methods, such as, illustrative applications, examples of the principles and high-level definitions. Hence, the principles of the dialogue signify the dynamic aspects of the required interface and considered as the "feel" of the interface. The 7 dialogue principles can be defined as follows (Sommerville, et al., 1994):

- 1 Suitability for learning
- 2 Suitability for individualization
- 3 Self-descriptiveness
- 4 Suitability for the task
- 5 Conformity with user expectations
- 6 Controllability
- 7 Error tolerance

The ISO 9241 defines the concept of usability in Part 11 of standard by efficiency, effectiveness and satisfaction of the human user. These three standards can be measured as quality factors of usability. These standards can be evaluated by decomposing them into sub-factors, and finally, into usability measures.

On the other hand, Part 12 of the ISO 9241 defines the information presentation in standard for the organization of information such as: location, grouping, arrangement, labels and alignment. This information display graphical objects and coding of information by seven attributes, for instance, color, shape, size, abbreviation and visual cues. These attributes are defined in the recommendations mentioned in the standard to supports one or more of the 7 attributes. These attributes are:

- 1 Detectability
- 2 Discriminability
- 3 Clarity
- 4 Consistency
- 5 Legibility
- 6 Conciseness
- 7 Comprehensibility

The Part 13 of the ISO 9241 standard defines that the user guidance information should be specific for the current context of use and should be distinguishable from other displayed information by the following means:

- 1 Prompts indication
- 2 Feedback
- 3 Status information

- 4 Error management
- 5 On-line help

### 3.8.6. Prototyping

Software prototyping refers to the activity of creating dummy, partially functional or fully functional sample of software applications, i.e., incomplete versions of the software program being developed. These prototypes are built to verify the designing and development process and getting feedback specifically from the end users. A prototype typically simulates only a few aspects of the final solution, and may be completely different from the final product but we refer here to software prototypes only (Myers, et al., 2000).

Prototypes have been used for not only for product and process evaluation but also for estimation of cost and effort spent and required to complete the application. Thus these prototypes work as millstones like any other deliverable for project management in software engineer paradigm. A prototype is more trusted milestone where the semi-finished application can be measure easily than any process. (Landay, et al., 1995)

## 3.9. Usability

Usability is a key question in the field of UI design and development and the big part of its answers lies in the content and structure of the computer application or webpage presented to user. Majority guidelines provided by research work are mainly focused on the usability of the system (Gould, et al., 1985). These user interface guidelines are mostly based on logic and reasoning called as common sense (Goodwin, 1987). In our research we have spotlight on logical representation of information structure and representation, though usability and it usability measures will not be the main focus of the research.

## 3.10. User Interface Modeling

User interface modeling is a design and development methodology in a model driven software engineering paradigm that enables user interface designers and computer programmers can design and implement a UI in a professional and systematic way (Da Silva, 2001). Today's user interfaces (UIs) are multifaceted software components, which play a vital role in the usability qualification of an application. The development of UIs requires therefore, not only guidelines and best practice reports, but also a precisely defined development process including the elaboration of visual models and a standardized notation for this visualization. Many other researchers have argues in favour of model driven information system development process have given much importance of using models for

User Interface development (Frank, et al., 1993) (Uschold, 2008) (Petrasch, 2010) . User Interface models are defined through the user-centered design iterations to refine and match users requirements and achieve maximum usability of the application (Viswanathan, et al., 2010). There user interfaces are design and developed through an intense analysis of users and their feedback. This analysis involved not only interaction with users by development team but also their behaviour and working efficacy at semi functional user interface models (Frank, et al., 1993). These user model also work for context aware models that can enable UI designers to design models and reuse them with some alteration acceding to the use context (Van den Bergh, et al., 2004) . These user Interface model when verified by user can be converted or extended to fully functional GUI through automated GUI generation procedures (Puerta, 1996).

Model Based User Interface Development Environment (MB-UIDE) is an environment that facilitates the UI a designer to design a UI model and extend it to a complete GUI according to the designed model.

### 3.10.1. Modeling Languages

#### 3.10.1.1. *UML*

Unified Modeling Language (UML) is mainly design to make models for business logic and data storage components of software engineering process. Still some aspects of user interface modeling can be realized using UML.. Due to extensive use of UML in other components of software development Majority of researcher and attempts are made to make is usable for UI modeling as well (Brockmans, et al., 2004) (Cranefield, et al.) . The language is not mainly intended for this kind of modeling, thus may render the models somewhat synthetic.

#### 3.10.1.2. *UMLi*

UMLi is an extension of UML, with added features for representation commonly occurring in user interfaces. UMLi aims to address this problem of designing and implementing user interfaces using a combination of UML and MB-UIDE (Da Silva, et al., 2003).

#### 3.10.1.3. *DiaMODL*

DiaMODL combines a dataflow-oriented language (Pisa interactor abstraction) with UML Statecharts that has focus on behaviour. This combination provides a capability of modeling the dataflow as well as the behavior of interaction objects. It can also be used for documentation of the UI design including the structure and functions of user interfaces.

## 3.11. Semantics aware Interfaces

Since semantic and ontological framework has changes the software industry and related work especially web applications and converted online data-stores to knowledgebase. This semantic application and data contains information about data like semantic categories, relations and constraints of data. These semantic and ontological models have given a new information system development base as discussed in earlier chapter Ontology Driven Information System (ODIS) (Wand, et al., 1990) (Uschold, 2008). Osterwalder have used ontological modeling approach for modeling e-economics with socioeconomic metaphors modeling (Osterwalder, et al., 2002) . Thus, it requires these semantic and ontological rules reflection at user interfaces too. There are some attempts to translate semantic web content to the user interface of web applications (Alexander, et al., 2003). They have also exploited the consequences requires to reflect at user interfaces from semantic search (Garcia, et al., 2003) (Latif, et al., 2009). Ontological model not only provide data classification and structuring but also validation measures for data and model itself (Aljawarneh, et al., 2009) (Shanks, et al., 2003).

ODIS specifically states the requirement of ontology and semantic relationship and constraints delivery at user interface in Knowledge representation and sharing rule to the user through user interface (Guarino, 1995) (Guarino, 1997) (Georgiev, 2005) (Uschold, 2008). Ontology can be a base to provide development of adaptive interface in Knowledge based User Interface environment (Sukaviriya, et al., 1993).

Till now there are developments of semantic aware user interfaces carrying semantics in parallel to conventional user interface development approach or adding semantic concerns to user interfaces (e.g. semantic tags at web-interface) (Paulheim, 2009). There is still need to use these semantic and ontological information structure and data classification constraints as a base information model for user interface in addition to user, context and device properties at upper layers to the base concept.

### 3.11.1. Data Formats and Semantic Classification

In this research exercise, we have adopted ontology base user concept modeling to define a computational and logical prototype that can be validated and verified not only specified data ranges of represented knowledge but also the relationships among them.

### 3.11.2. Input / Output Data Validation

These ontological classes and constraints are also proposed to provide data validation services based on semantic classification of data (Aljawarneh, et al., 2009).

The motivation behind semantic aware user interface designing is a base line for our current research. The aim to make semantic and ontological contribution at user interface design and development is to

get the targeted concept delivery at user interface through their semantic meanings, structure and classification.

## **3.12. Intelligent User Interface**

Advanced applications are characterized by large amounts of information to be conveyed knowledgebase with trillions of records and millions of data structures like linked data and additive understanding and reasoning abilities, complex task structures, real-time performance characteristics, and incorporation of autonomous or semiautonomous agents. A user interface is required to address all these complexities as well as exploit the additional metadata and semantic information with data. Intelligent user interfaces are defined as a human-machine interfaces that aim to improve the efficiency, effectiveness, and naturalness of human-machine interaction by representing, reasoning, and acting on models of the user, domain, task, discourse, and media (Woods, 1991) .The area of intelligent user interfaces covers a variety of topics concerned with the application of Artificial Intelligence and knowledge-based techniques to issues of human-computer interaction.

## **3.13. User Interface Development**

The thought base for User Interface Development in our research focus is the user interface model of a domain ontology that can be directly mapped to the GUI. ODIS has mentioned the same idea to explore to make a 60% of user interface (according to the iceberg model) then develop or introduce the user interaction and graphical representation concerns to the user interface model driven by ontology (Uschold, 2008) (Kristiansen, et al., 2007).

# Functional Programming Approach

As discussed in earlier chapters we made some base lines to our experiments regarding concept representation. Semantics and Ontological framework can define standards for concepts presented through computational models. This framework can also provide standards for the concept representation at User Interface level. A user model as the base of user interface can be built using formal ontologies. Language supporting Higher Order Functions can be used to build Graphical User Interfaces by mapping the user defined model with typed graphical user interface controls. This part of the work discusses a logical way using mathematical functions implemented using Haskell to present the standard concepts of computational model to user interface through ontological user model.

## 4.1. Introduction

As initial experiment of the research we explore the answer for the feasibility and possibilities to generate a GUI that preserves ontological essence at user output devices. We have explored the answer for the first research question “*How semantically defined concepts can be delivered at GUI?*”. Mathematical illustration of ontological rules can be directly represented to the computational model in Haskell, thus we adopted functional programming approach.

A computer application is a computational model of a real world concept. The concepts conceived by the user depend on the computational model. The computational model is built by human and represented concepts are based on the human understanding. These concepts are delivered to the user through the user interface of the computer application (Smith, 2004).

With reference to the “*The iceberg analogy of usability*” discussed earlier, the major part of GUI (user model) provides the concepts of computer application to the user discussed in section 3.6. This underwater (hidden) part of iceberg includes the objects, properties and relations between them. The majority of recent work in user interfaces is focused on improving only the look and feel. Although many guidelines to improve the look and feel of a user interface have already been devised (Smith, et al., 1998) yet no specific rules exist for modeling a user interface (Alexander, et al., 2003).

On the other hand semantics and ontological frameworks are being used by many knowledge based systems to formalize the semantics of a domain via concepts and relationships. But in current computer applications, the same concepts are represented in many different forms on the visual component of a user interface and provide much displaced picture of the same concept. Hence, semantically similar concepts are perceived differently depending on the actual implementation of the user interface.



In order to consistently harmonize domain semantics with its visual representation requires standardization of the domain semantics, i.e. concepts and relationships, as well as their mapping onto standard visual components, so called widgets. Such an Ontological User Model (OUM) provides a direct mapping of the user model, component 3 of the iceberg, to the look and feel. Ontological user modeling can lower the cost of not only user interface development and maintenance but also for application testing (Alexander, et al., 2003).

This chapter discusses a strategy for representing and mapping ontological user model to a Graphical User Interface in the domain of spatial information systems. We will experiment some strategies to exploit the self-explanation of ontologically specified domain to map directly to user interface. Ontology and semantic framework here provide us a user model with semantic and ontological structure of data with ontological funtions. We have used, as an example, spatial data domain to be presented at GUI as a 2D graph.

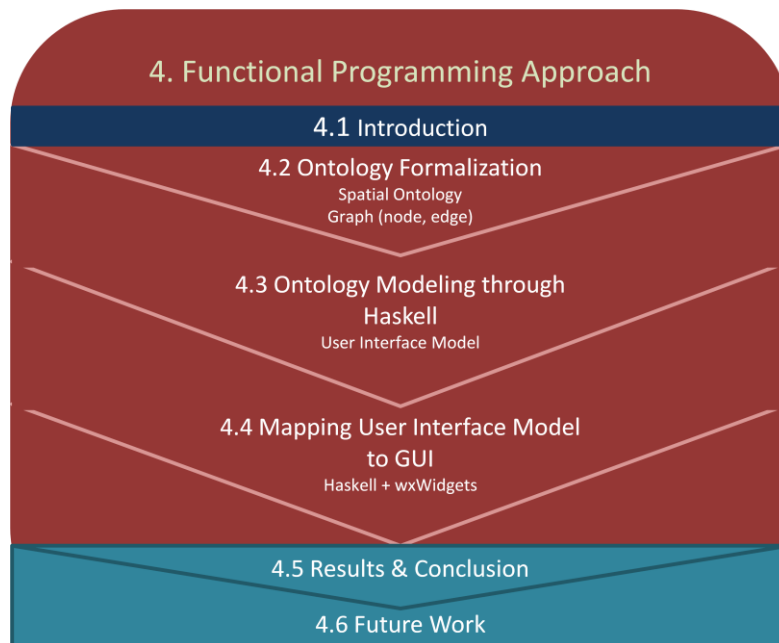


Figure 4.1: 1.4 organization

Hence a 2D graph is discussed as an example implementation of GUI based on spatial ontology and its standardized visual representation. Based on related work, we formalize ontology for 2D graph and define a relational algebra in order to define manipulations and axioms. We implement the relational algebra; using the functional programming language (Haskell) to create a user model, as in terms of the so called user model of domain ontology. This functional programming language also provides us a facility to incorporate functional ontology that provides the ontological functions. In a last step, shown in Section IV, we discuss the mapping of the Haskell implemented base ontology (presented to user model) to GUI with a 2D graph example.

Next section talk about ontology formalization where we have formalized semantic data and ontological function for a graph as an example of spatial ontology. Third section provides a user interface base model for the using Haskell (functional programming language). In the last stage of experiment we developed a GUI through (one to one) mapping of Haskell to wxWidget GUI component to build a GUI. At the end we have discussed the outcomes of the experiment though detailed results in comparison to other experiments will be provided in 1.7 for the results and conclusion.

## 4.2. Ontology formalization

Specification and prototyping is essential for standardization of data and process modeling (Frank, et al., 1995). Ontology is an explicit specification of a conceptualization. Current research is exploring the use of formal ontologies for specifying content specific agreements for a variety of knowledge-sharing activities (Gruber, 1995). In our example ontological formalization specifies the conceptual model for a 2D graph as an example of ontology. Spatial ontology formalization is needed to get the specification of the conceptual model. The formal ontology can be then represented in relational algebra.

### 4.2.1. Formal Ontology

Good ontology and good modeling can be achieved in a specific domain through a precise representation of entities that exist in reality (Smith, 2004). Here this domain is narrowed down to the discipline of spatial ontology for a 2D Graph model.

Formal Ontology can be intended as a theory of priori distinction among the entities of the world and among the meta-level categories (Guarino, 1995). Entities represented here are a 2D Graph, and Node and Edges as a part of graph. Every entity has an identity function as the identification for that entity.

Formal ontology deals with the categories and relations which appear in all domains and which are in principle applicable to reality under any perspective (Grenon, et al., 2004). We state the entities and relation among them.

Our domain model is given as follows: Graph is called by a Title here referred as Graph name. The graph consists of some points and connection between them referred as Node and Edges respectively. Each node represents a location in 2D space and Edge represents the connection between two nodes. It is a straightforward definition of a graph to formalize entities and relationship among these entities. These specifications are stated using algebraic specification.

An example of simple 2D Graph:

- Table 4.1 present the properties and structure of a graph
- This is an example of 2D graph just to describe how axioms and categories can be formalized using relational algebra and functional programming language. It is not any standard ontology for a graph.
- Figure 4.2 present an example instance of graph representation.

Graph Properties and Structure

Graph	Graph ID	
	Graph Name	
	Node	Node ID
		Number
		Position
Edge	Edge ID	
	First Node	
	Second Node	

Table 4.1: Ontology Description of a Graph

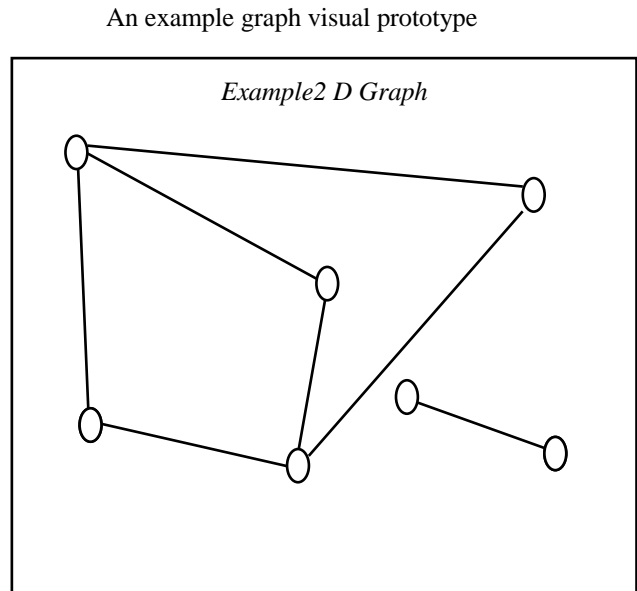


Figure 4.2: An Example representation of Graph

### 4.2.2. Relational Algebra

The term "algebraic specification" originally referred to the use of algebras to model programs, and to use equatorial axioms to write specifications (Car, et al., 1995). Here spatial ontology is formalized using formal ontology and relational algebra. Algebra represents spatial categories and relations in space dimension. Relations also defined potential functions of any object from a specific category.

Axioms can be specified as property functions for any object of a category X like:

```

class X
  p1(x) -> q1
  p2(x) -> q2
  ..
  pn(x) -> qn
where
  p = property
  
```

```
q = property value
x = Object of Class X
```

Relations for 2DGraph graph examples are:

Graph contains a Name, List of Node and List of Edges. Graph is identified by GraphID

```
id :: Graph → GraphID
graphName :: Graph → Name
graphEdges :: Graph → [Edge]
graphNodes :: Graph → [Node]
```

Here Node contains a 2D location and a Number. Node identified by NodeID

```
id :: Node → NodeID
nodeNumber :: Node → Number
nodePostition :: Node → Position
```

and Edge is a connections between two nodes. Each Edge contains a pair of Node referred by their NodeID while Edge is identified by EdgeID

```
id :: Edge → EdgeID
startNode :: Edge → ID
endNode :: Edge → ID
```

Potential Operations on Graphs also provide relations in the entities for the graph. Relational algebra can also define potential processes.

```
class X
  f1(x) -> s1
  f2(x) -> s2
  ..
  fn(x) -> sn
where
  f = Function
  s = Behavior of function
  x = Object of Class X
```

### 4.2.1. GUI States

A concept is presented to a user interface through the meta-data including information components and their structure, while the data and quantities provided by data provided by user or and other data source at user interface provides the current state of user interface. A user interface is a continuous process that provides the current state through loaded quantities at user interface. With same concept representation, these states are changed with data updates by user events or some software code. Here we will discuss different functions that update or change states of Graphical User Interface. Any potential process can update the measurement/values of the properties resulting in state update.

Initiation of any potential process of the concept reads the current state of the user interface and responds accordingly by updating the state. Potential processes for this example provide following relations:

```
newGraph :: Name → [Node] → [Edge] → S(GUI)
deleteGraph :: GraphID → Database → S(GUI)
updateName :: GraphID → Name → Database → S(GUI)
updateGraph :: (GraphID → Graph ) → S(GUI)
```

Any node and Edge function can update a state of Graph being a part of Graph. Graph object will propagate the signal of change of state to GUI.

```
addNode :: Number → Postion → S(Graph)
deleteNode :: NodeID → S(Graph)
updateNodeNumber :: NodeID → Number → S(Graph)
updateNodePosition :: NodeID → Position → S(Graph)
addEdge :: NodeID → NodeID → Graph → S(Graph)
deleteEdge :: EdgeID → S(Graph)
updateStartNode :: EdgeID → NodeID → S(Graph)
updateEndNode :: EdgeID → NodeID → S(Graph)
```

The algebraic specification is implemented using functional programming language. Here Haskell is used for implementation for this relational algebra.

## 4.3. User Model by Haskell programming

### 4.3.1. Why Haskell

Haskell is an advanced purely functional programming language. It provides features like higher-order functions, lazy evaluation, equations and pattern matching, strong static typing and type inference, and data abstraction (Hudak, 1989). Haskell allows types (inferred and functional types) through axioms to form classes and category as a domain concept.

wxHaskell with a powerful second order language can be used to generate a user interface based on ontologies. A functional programming language allows the designer to examine whether the model is a correct representation of reality with respect to the ontology, and if the objectives and expectations are fulfilled (Cardelli, et al., 1985). Algebraic representation of ontology can be written down directly to Haskell. Representation of these relations can be done with a mapping function which can map functions (ontological functions) to a function (I/O function). In this experiment Haskell (functional programming language) is used for representation of relation algebra for a basic vector graph representation (function graph (node, edge)) as an example of a Spatial Ontology. Haskell represent relational algebra by its typed structure. Moreover State Monads facilitate for representation of states.

### 4.3.2. Representing Algebra in Haskell

Graph consists of nodes and connection between nodes referred as edge. Representation of Algebra is achieved in Haskell through

Simple types  $t$

First order function types  $f :: t \rightarrow t'$

Higher order functions  $g :: f \rightarrow f'$

Here is data-structure implementation in Haskell for:

#### *Node*

Nodes are defined as some points attached with an number. `nodeNumber` is an identity for the node and `nodePosition` defines the position of node in 2D space.

```
data Node = Node {
    nodeNumber :: Number
    , nodePosition :: Position
} deriving (Eq, Ord, Show)

where
newtype NodeNumber = NodeNumber Int | NodeNumber unknown
type Position = Point
```

#### *Edge*

Edges are defined with two nodes `startNode` and `endNode`

```
data Edge = Edge {
    startNode :: Number
    , endNode :: Number
} deriving (Eq, Ord, Show)
```

#### *Graph*

Graph is container or aggregate for lists of nodes and edges with a name. `graphName` is an identity for the graph.

```
data Graph = Graph {
    graphName :: Name
    , nodeList :: [Node]
    , edgeList :: [Edge]
}

where
type Name = String
```

### 4.3.3. Manipulating States (State Monads)

Business logic of software provides a flow of information and a structure to process and manipulate the values base on the concepts. The GUI provides information in measurements of properties to the user and operations for the user. Operations available at the user interface allow user to see and update the current measurements. Current values of instances of any concept form a picture a state at User use interface. Querying and updating operations thus result in reading and update the current state.

Haskell represents states using state monads<sup>43</sup>.

```
newtype State s a = State { runState :: (s → (a, s)) }
```

where 's' is a state type and 'a' is an observation from the state as a simple type or a data structure of an object.

All observations are used to read any measurement at resulting state of state monad. The state is stored in shape of measurements in quantity or quality to data storage. State monads are used here to read and write the current state of Data Store to get and change state of Graph through ontological processes. The ontological processes are stated as relations in user model.

Here states are used as a computer model of reality.

```
class classX a where
  f → State a objectX
```

Data Storage is instantiated for any class CategoryX. f is the function to update the current state or read an objectX as shown in Table 4.2.

e.g. manipulating DB states

```
f → State DB object
```

The same way ontology is instantiated for UI Model

```
f → State userModel object
```

Here any function can read or update the state of the database model or the user interface model. For 2D graph examples, categories are defined on the base of behavior. All potential processes (for Nodes, Edges and Graphs classes) result in the reading or the update in current state of graph. Haskell defines manipulation of state 'a' for mentioned graph processes as following:

```
class a
  newGraph :: Name → State a GraphID
  deleteGraph :: GraphID → State a Bool
  graphbyName :: Name → State a (Maybe Graph)
```

---

<sup>43</sup> <http://cvs.haskell.org/Hugs/pages/libraries/mtl/Control-Monad-State.html>

```

setGraphName :: GraphID → Name → State a Bool
newNode      :: Number → Position → Name → State a NodeID
deleteNode   :: NodeID → Position → State a NodeID
updateNumber :: NodeID → Number → State a Bool
updatePosition :: NodeID → Position → State a Bool
getNumber    :: NodeID → State a Number
getPosition  :: NodeID → State a Position
addEdge      :: NodeID → NodeID → Name → State a EdgeID
deleteEdge   :: EdgeID → State a EdgeID
setStartNode :: EdgeID → Number → State a Bool
setEndNode   :: EdgeID → Nr → State a Bool
edgebyNode   :: NodeID → State a [Edge]

```

Type of an object to some state either changes in state or observation of current state.

#### 4.3.4. Data types and Typed Classes

Objects keep properties and potential process collectively called behaviour of an object. Categories are made of the base of commonalities in behaviour as criteria of the class.

Using Haskell, a type is not only a simple type, there are also types defines using axioms. Ontological axioms can be used to specify the exact domain concept as a type of any object. Also functional type can be defined as a class to represent behavior of an object or category.

In our example of 2D Graph,

NodeNumber is a type which is associated with positive integers but is it neither an integer type nor a positive integer. NodeNumber cannot be added, subtracted or manipulated like integers. It have only two functions = or  $\neq$ . Haskell define these types with scales as functors.

```

NodeNumber = f (x)
where      x ∈ Z+

```

Haskell wrap these types to manipulate it different from simple integer types.

#### 4.3.5. User Model

The core of this experiment is building a base user model that can represent formalized domain ontology in software programming language directly without any translation. User model is a representation of ontological structure of conceptual model which can be directly mapped to User interface. This user model works as the base layer of user interface as provided in ice berg analogy discussed in section 3.6. Here User model provide information architecture, classification through domain ontology. In short, the user model defines an ontological structure from an application and maps it to the GUI elements.



Here user model also works as the middle layer between the business logic of software and GUI. It represents ontological model that presents user interface as a GUI state at upper layer and at lower layer it is connected to business logic and database communicating through the read and write functions.

As described in earlier section these functions are triggered automatically by software to update the current state of GUI or User model works using the formalized ontology to represent the behavior of a computer application. It is then directly mapped to a GUI and forms the base model for a user interface. It is based on ontological definition in axioms defining properties and relations. It ensures the data values from or to the user interface are values fulfilling the ontological constraints.





GUI	Populate GUI (Drawing Canvas and Panel)	User Interface Events (f)
	 Read Graph State	Update Graph State 
User Model	Ontological Model (Graph)	Potential Processes (f') (Graph State)
	 Read Graph	 Update Graph
Business Logic		
DBMS		

Table 4.2: The mapping of User Model to GUI through Data Flow (Haskell programming)

## 4.4. User Model in Haskell

We got following declaration details from the implementation of user model in Haskell implementation:

### 4.4.1. Graph

```
data Graph = Graph { graphID :: GraphID
                    , graphName :: Name
                    , nodeList :: [Node]
                    , edgeList :: [Edge]
                    } deriving (Eq, Ord, Show)

newGraph :: Name -> State NameGraphDB Bool
deleteGraph :: GraphID -> State NameGraphDB Bool
graphbyID :: GraphID -> State NameGraphDB (Maybe Graph)
graphbyName name = State $ \db -> if isGraph db (identifyName db name)
setGraphName :: GraphID -> Name -> State NameGraphDB Bool
```

### 4.4.2. Node

The Haskell programming declaration of node data type and functions defined in mathematical model:

```
data Node = Node { nodeID :: NodeID
                  , nodeNumber :: Nr
                  , nodePosition :: Location
                  } deriving (Eq, Ord)
```

Here “*a*” is the User Model “*UM*” that called of these functions:

```
class Nodes a where
newNode :: Nr -> Int -> Int -> IntScale -> a -> State NameGraphDB Bool
deleteNode :: NodeID -> a -> State NameGraphDB Bool
setNodePos :: NodeID -> Int -> Int -> IntScale -> a -> State
NameGraphDB Bool
setNodeNr :: NodeID -> Nr -> a -> State NameGraphDB Bool
```

### 4.4.3. Edge

```
data Graph = Graph { graphName :: Name
                    , graphSelected :: Bool
                    } deriving (Eq, Ord, Show)
```

Here “*a*” is the User Model “*UM*” that called of these functions:

```

class GraphOps a where
  addGraph :: Graph -> State a Graph
  graphbyName :: Name -> State a Graph
  deleteGraph :: GraphID -> State a GraphID

```

Here “*UM*” defines user model that contain either a Graph with list of Nodes and Edges or an empty model:

```

data UM = UM Graph [Node] [Edge] | UEmpty

```

“NodeID” is different from “*EdgeID*” or “*GraphID*”. Thus that ensures there will not be any element which is processing any data without semantic representation.

```

newtype GraphID = GraphID ID
  deriving (Eq, Ord, Show)

```

```

newtype EdgeID = EdgeID ID
  deriving (Eq, Ord, Show)

```

```

newtype NodeID = NodeID ID
  deriving (Eq, Ord, Show)

```

## 4.5. Mapping User Model to GUI

Mapping the domain ontology of graph presented in User model to GUI provides the concept of representation of meta-data and possible operations. These relations were stated in user model. A transparent interface is the best interface where User interacts directly to the conceptual model entities and performs potential processes. GUI can have a bijective mapping to the formalized conceptual model. The bijective mapping create real picture of direct interaction to the conceptual model.

Here GUI shows the current state of Graph at drawing canvas and provides controls to initiate any potential process. Graph data structure provides meta-data of Graph. Haskell provides higher order functions to map ontology to GUI. We have used simple graph example, this graph can be any weighted or directed graph, or a more sophisticated applications like Shortest Path. For every different application, there are different concepts presented in axioms and different categories. The user model provides the framework for any application and can be mapped to the GUI.

### 4.5.1. GUI construction tools

Different graphical libraries like Wxwidgets, OpenGL, QT and GTK available to plug with Haskell. Different GUI development libraries are also available based on above mentioned graphical libraries available. Here wxHaskell and wxGeneric is used for construction of GUI based on formalized ontology. WxGeneric allows building customised controls based on the data type.

#### 4.5.1.1. Composability and Typed User Interface

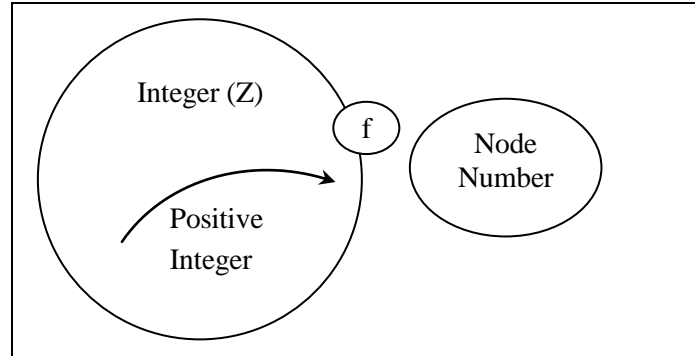


Figure 4.3: Node Number as a Functional Types to represent ontology

Mapping a semantic data types to GUI requires semantic data typed user interface controls. WxGeneric provide facility to make specific typed controls. Typed controls by wxGeneric represent the conceptual structure of categories to GUI. Panels can group simple type objects and these simple type objects can be represented with typed controls. These are composite controls like text box for Number type<sup>44</sup>. For 2D Graph example typed controls are used through composability

```
type IntEntry = Composite (TextCtrl ()) (IO Int, Int → IO ())
type NodeIDEntry = Composite (TextCtrl()) (IO NodeID, NodeID → IO ())
type NodeNumberEntry = Composite (IntEntry ()) (IO NodeNumber,
                                           NodeNumber → IO ())
```

#### 4.5.2. Drawing at Canvas

Current state of reality for a 2D graph can be shown to user as 2D picture. It is the ontological show method or the representation of vector values. This picture can be mapped to GUI by draw function. Draw function draw graph to a display screen. The graph structure represents user model. Drawing can visualize all objects from user model at a specific scale. For example of 2D graph, the drawing is a graphical representation of 2D graph at specific scale.

#### 4.5.3. GUI

GUI reads the current graph values from “NameGraphDB” a binary file containing graphs

```
mainGui :: NameGraphDB -> IO()
```

A typed panel can be created to manipulate a structured data type. Typed panel is composed of typed controls. Each simple type can be mapped to GUI through typed text boxes, list boxes, radio buttons,

---

<sup>44</sup> <http://lindstroem.wordpress.com/2008/03/16/proposal-adding-composability-to-wxhaskell/>

check boxes etc. The mapping function have domain of types and data structure in user model and co-domain is the set of GUI data controls. Current example creates a very simple interface panel, which is based on a user model. It ensures the consistency and standardization.

Here GUI has two major parts:

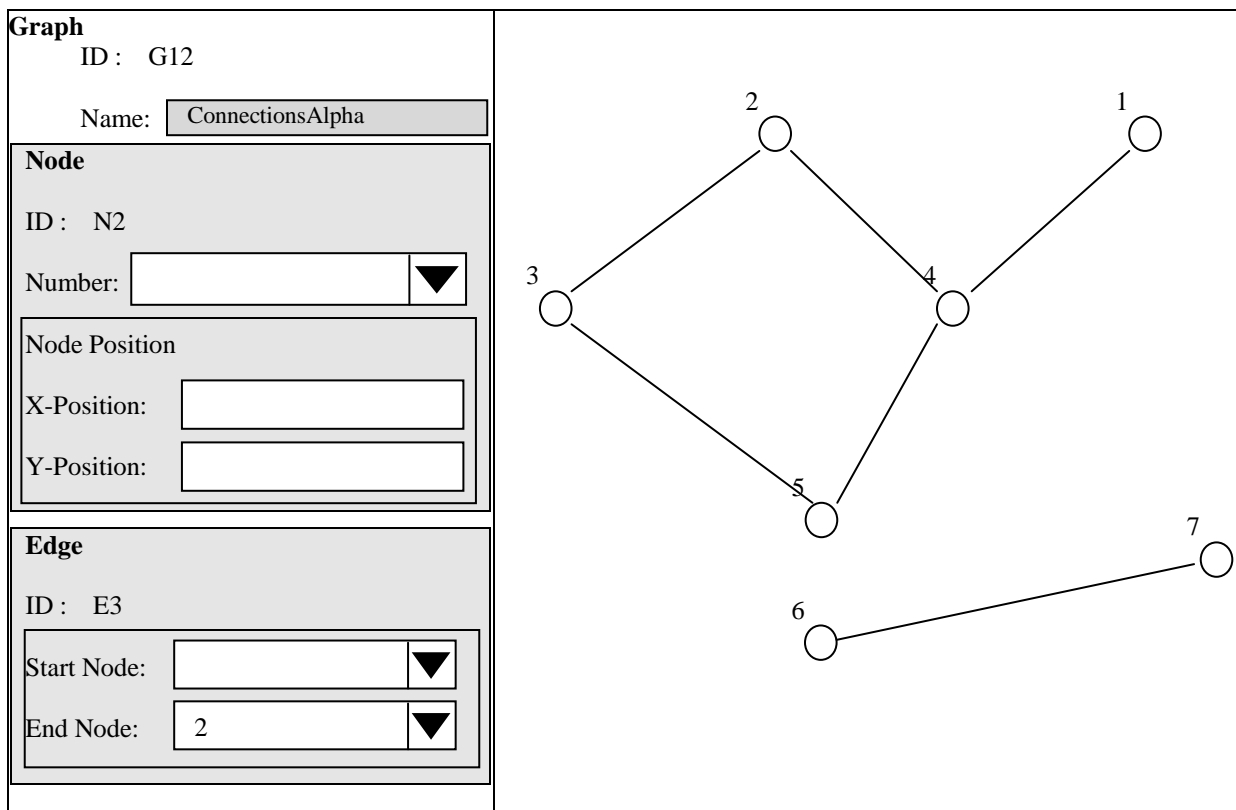


Figure 4.4: Panel and canvas of the graph properties through typed controls

### 1 Panel

Panel consist of the customized controls that can read and write Haskell defined types associated to the GUI controls. wxGeneric provide facility to create a customized user interface control.

```
$(derive ['Graph, 'Edge, 'Node])
instance WxGen Graph
instance WxGen Edge
instance WxGen Node
```

We have used customized controls as “Panel()”, “Text()”, “List()” and “Combo()”

### 2 Canvas

Canvas draw vector graph on the canvas it is also customized to read and write a “Graph”:

```
drawNode :: (DC a) -> Scale -> Graph -> IO()
drawNode :: (DC a) -> Scale -> Node -> IO()
drawEdge :: (DC a) -> Scale -> Edge -> IO()
```

here we have define many types which are similar in implementation but tagged or wrapped for making difference in semantic meanings like ID all of the IDs are numeric integers but semantically Mapping Processes to GUI

Mapping of processes to GUI is more complex than mapping of data structure to panel or drawing. Processes results in Change of graph state. So it is a function type which needed to be mapped. Comparative to imperative programming styles, Haskell can provide a higher order mapping functions to map these processes. Still it is bijective morphism though mapping function is a higher order functions.

The mapping function has the domain of User Interface events and the co-domain is the set of processes in user model. The processes can be mapped to the events of GUI controls. Implementation details of these mapping using different controls and clipping initiation of processes at the specific event was not supported with many of the graphical libraries especially the graphical libraries or their components compatible to functional programming. The majority of GUI designing guidelines and aim of transparent GUI can be achieved by these mapping functions.

Here the issue is with mapping processes having parameters and modes (specific GUI State); the GUI needs to provide some control to select objects to initiate any processes on them. Specific processes can be done to objects of specific category. In graph example the setGraphName can change name of a graph, and parameter here is the GraphID.

At any operation which requires a user input can be represented as a panel or dialogbox for getting input parameters. Graph example use notebooks for each class containing buttons for operation at each tab. At button click event panels appears for input data form user. This process input panel contains typed objects for every simple type or identity type for a data structure like node which is identified by the nodeID. Each process panel has an Ok button to run process in the user model which is further mapped to Data Storage using Haskell State monads for the data store.

#### 4.5.4. Limitations and constraints

There are many issues related to GUIs which are not discussed here like

- 3 Synchronization of sequence of user actions with ontological processes.
- 4 Data storage and buffering at user model level.
- 5 Database error handling.
- 6 Handling GUI Modes; like selection of objects bounding domain of objects to perform few operations on selected component e.g. deleteNode process can be done only visible nodes at user model. These constraints are not on category but at the set of values.
- 7 Semantic data transfer; data input and output through User Dialogues also have semantic types controls

## 4.6. Results and Conclusion

As a result for the experiment we got a user interface for the vector graph containing edges, nodes and their relationship in graph. Spatial Ontologies have been mapped to GUI by a function using an intermediate layer of user model containing all ontological vocabulary and structure of the graph with additive information of user interface elements. This layer works as a user model for a user interface. It provides the application semantics in form of an ontological framework of the concepts to be represented at the GUI. It provides relations to specify meta-data and potential processes for different categories mapped to GUI elements. This mapping from domain ontology to GUI is bijective to present an exact view of conceptual model at the GUI.

Higher order mapping functions can clip ontological processes to the GUI controls. Typed control can be used to read or provide objects of specific category for any potential process of the category. Any graphical control and its behavior can be mapped with the ontological definition of objects which includes the processes as a typed control. This bijective mapping using typed user interface elements carry all semantic and ontological constraints with the vocabulary and relationship provided in domain concept. Thus, this structure presented as GUI through typed controls (mapped to the semantic classification) provides true picture as presented in the base concept at user interface level.

Here we got answer to our first research question (“How can semantically defined concepts be delivered at GUI?”) that a bijective or one to one mapping of all domain entities along with semantic structure and constraints of domain concept can give. Still there are semantic data driven graphical library is required which allows semantic typed data flow. Secondly it is needed to implement the methodology using some proper standard domain ontology from the big list of standard ontologies (discussed in section 2.2) and semantic programming library like Jena (from java)

## 4.7. Future work

In imperative style languages much translation and complex systems are involved to map from the business logic to the GUI. There is a need to implement the idea using standard ontologies provided by many knowledge bases.

This experiment gives WxGeneric works on the data structure to construct an automated GUI. Where potential processes need to be implemented separately from data structure. There is still need of an abstract level GUI construction tool which can implement the typed controls with the classes. Where processes in classes can be handled as events handlers for events of UI ontology to update GUI state.

Mapping of Temporal ontology with GUI is also needed to be discussed in future. This mapping will require synchronizing user generated event through IO devices with temporal ontological processes.

In this experiment user action or sequence of actions are not discussed but for the next layer of ontological process is ontological definition of user action to perform certain tasks like clicking at canvas for selection of nodes or edges. The meta-data or ontology at User Interface should include ontological description to access and activate the possible process at any object (User-Interface Ontology).



# Ontology based User Interface Development: User Experience Elements Pattern

The user experience of any software or website consists of elements from the conceptual to the concrete level. These elements of user experience assist in the design and development of user interfaces. On the other hand, ontologies provide a framework for computable representation of user interface elements and underlying data. Last chapter discussed the logically defined methodology for introduction of ontological framework driven user interface. In this chapter, a strategy introduced for introducing ontologies at different user interface layers with a standard ontology of vCard given in RDFs/XML. These layers are adapted from Garret's model of user experience elements (Garrett, 2002). These layers range from abstract levels (e.g. User needs/Application Objectives) to concrete levels (e.g. Application User Interface) in terms of data representation. The proposed ontological framework enables device independent, semi-automated GUI construction which we will demonstrate at a personal information management research application.

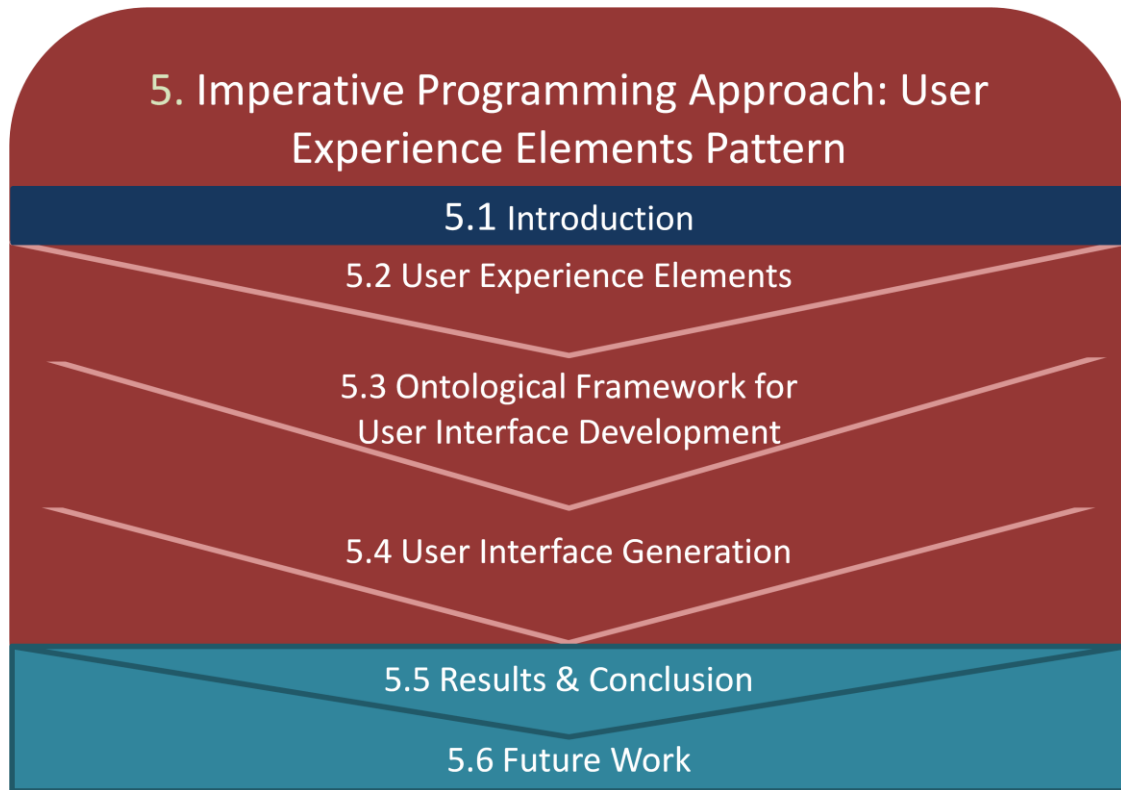
## 5.1. Introduction

Guarino, Smith and Gruber introduced the formalization of abstract domain concepts by using ontologies (Guarino, 1995) (Smith, 2004) (Gruber, 1993). By combining ontologies with logical theory, machines can be enabled to compute formalized conceptual models using shareable, domain specific ontologies as results showed from previous experiment results in 4.6. Moreover, formal ontologies ensure the meaning and consistency of the presented concepts and allow to unambiguously sharing domain specific concepts across decentralized information systems.

Ontologies have been used in information systems on several levels, for example for integrating databases, as business logic or for constructing Graphical User Interfaces (GUI). Additionally, ontologies have been exploited in software development processes. Uschold for example combined formal ontologies with model driven software development (Uschold, 2008). Uschold outlined that ontological model driven software development improves the development of complex software systems. Ontology based modelling carries the domain concept to the entire software development process.

In our work we present an approach for mapping formal ontologies to GUI automatically, so that the GUI preserves the domain specific properties of the underlying domain ontology. Such a mapping

supports device independent GUI construction as well as semi automatic GUI modeling. We discuss our approach along a Personal Information Management (PIM) example. We used vCard/hCard as a standard ontological model for PIM. We also employed Jena and SWT for mapping vCard ontologies to GUI.



*Figure 5.1: 1.5 organization*

However, the use of ontologies within the GUI development process also provides technology independent modeling. Moreover, it allows adding context-aware properties at the development level depending on the targeted devices and user roles.

The next section discusses elements of user experience providing the layers of user's perception. In section 3, we outline the properties of the formal ontology which should be presented at the GUI level and conclude our work by combining User Interface Ontology (UIO) and Personal Information Management.

## 5.2. User Experience Elements

Garrett introduces five elements of user experience by concepts underlying software or a website (Garrett, 2002).

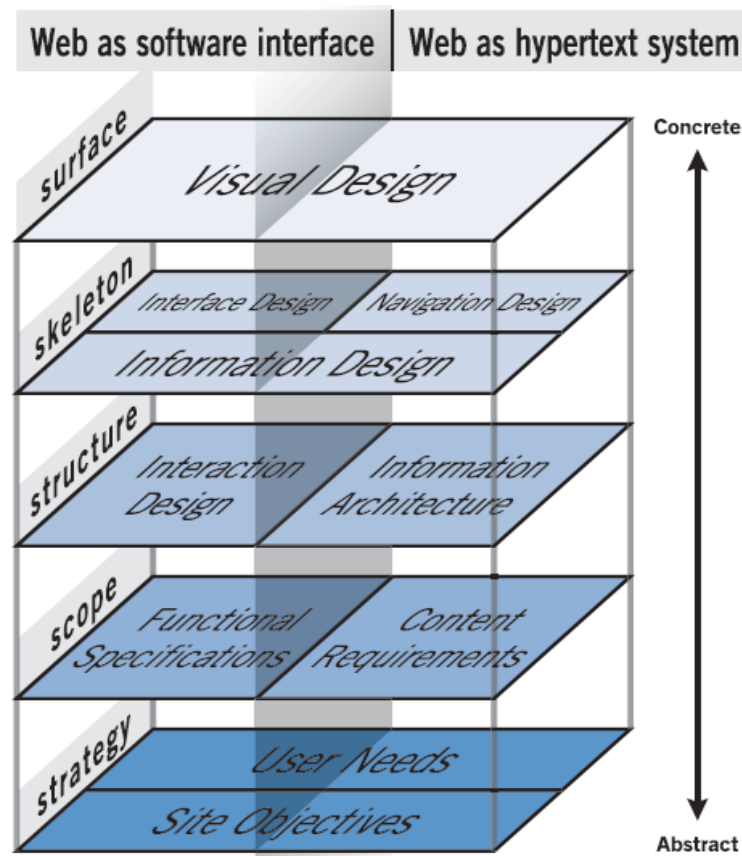


Figure 5.2: Adapted User Experience Elements by Garrett

These application concepts summarize the goals a software system should pursue. Garrett's elements collectively introduce different levels of such application concepts represented in an information system and will be described in the Table 5.1.

By referring to Garrett's model, we argue that representation and direct mapping of ontology from the base layer (i.e. strategy) to the presentation layer (i.e. surface) can help users to perceive the targeted domain concept. These elements can also be considered as layers of UI design and development. We adapted Garrett's layered approach to GUI development to follow through the elements of user perception. The ontological framework provides the capability for a formal specification of the concepts. It also enables the designer and developer of the website to implement ontologically specified domain concepts. Following user psychology also enables the designer to make a user

centred design. Separating the different roles in GUI construction on the different levels of user experience reduces development time and allows for independent optimization of each level.

Level of User Experience	Ontological Implementation	Example Application using Ontological Framework
Surface	Graphics Look and Feel	UIO Implementation at Graphical Library SWT, OpenGL, GTK, wxWidgets, QT
Skeleton	User Interface Ontology	Customized Textbox, List box, Selection Box, Date/Time tool, Containers, Buttons
Structure	Domain Ontology	vCard, hCard
Scope	Vocabulary (for Entities and Relations) Relations also represent Functions	Name, Address, Date of Birth, Email, Phone Number, Family Name, Zip Code
Strategy		Personal Information Management

Table 5.1: Adapted User Experience Elements by Garrett (Garrett, 2002)

Ontologies can be introduced to each layer for integrating the different user experience levels as follows (Table 5.1):

Garret represented the strategy as an abstract concept which the user abstractly defines. It specifies the user needs and objectives of the targeted GUI. At this level, there is no explicit ontological formulation in order to represent this abstract strategy.

- 1 Scope talks about what exists within the boundary of the domain. In this work, it refers to the details of the concepts and sub-concepts within the domain. It merely addresses the vocabulary used, but not its structure, which is part of the next level.
- 2 Structure defines the logical structure of the user interface. Domain ontologies specify this via relationships among concepts and sub-concepts forming the logical structure of the user interface. It provides a hierarchical structure of whole and part, but also associated concepts.

This ontological structure makes a map of information arrangement and navigation in related concepts.

- 3 The skeleton level provides the representation and interaction methods of an user interface. Ontologically formalized structures represented as User Interface Ontology allows specification of such a skeleton and translates the logical domain structure of step 3 into a user interface layout which is rendered in the next step.
- 4 Surface is the concrete implementation of the skeleton. It is the User Interface Ontology implemented through any computer graphical library.

In the following example of mapping vCard onto GUIs via those 5 steps, we have dealt with visualization of vCard to the GUI only. Interaction methods, update functions and initiation with UI events will be discussed in future research through mapping functional ontology to User Events.

### 5.2.1. Conceptualization through Ontological Modelling

Gruber suggested the properties of formal ontologies for conceptualization (Gruber, 1993). Uschold introduced ontological modeling for model driven software development to use the properties of conceptualization (Uschold, 2008). We use formal ontologies as a base structure for GUI development.

Formal ontology specifies the structure. UI Ontology will use the vocabulary of each entity concept and sub-concept, specified by formal ontology. These concepts and sub-concepts can be carried to the structured layer using ontological relations. Automatic mapping functions from formal ontology to GUI can maintain properties of conceptualization at GUI.

### 5.2.2. Personal Information Management (vCard/hCard)

Internet Mail Consortium (IMC)<sup>45</sup> has defined a standard for Personal Data Interchange as vCard (RFC2426). This standard has been broadly implemented (e.g. Apple's "Address Book", Microsoft Outlook) ensuring interoperability. vCard provide an ontological structure for personal information storage and representation. We use vCard as example ontology for personal Information.

---

<sup>45</sup> <http://www.imc.org/pdi/>

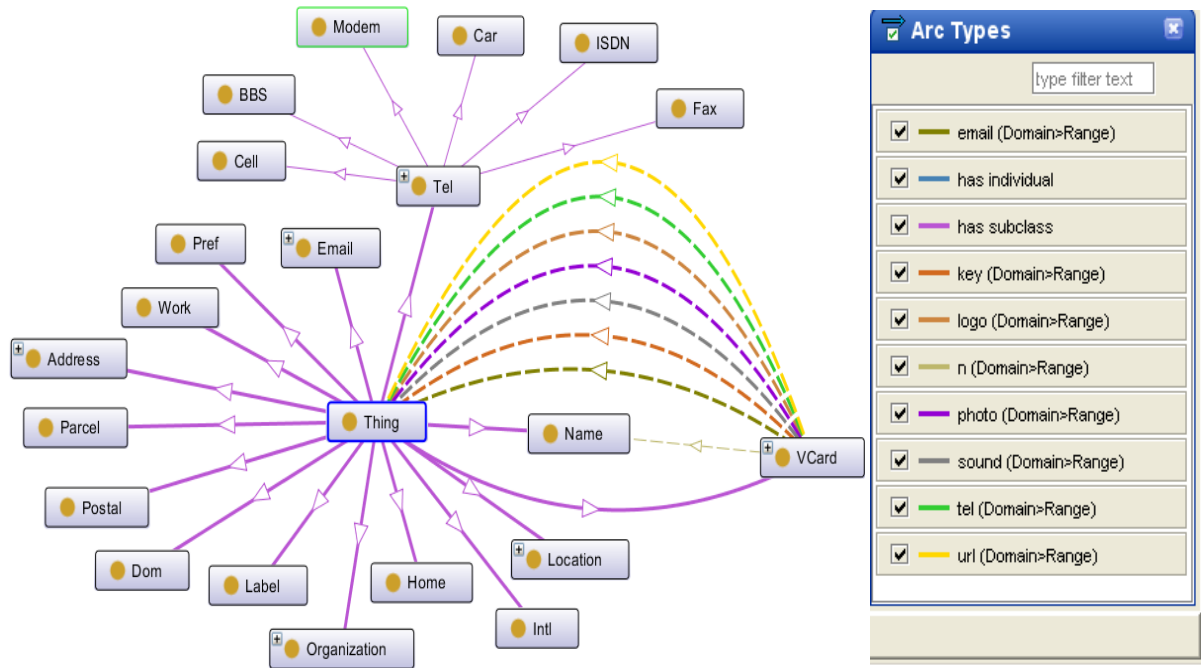


Figure 5.3: Structure of vCard classes

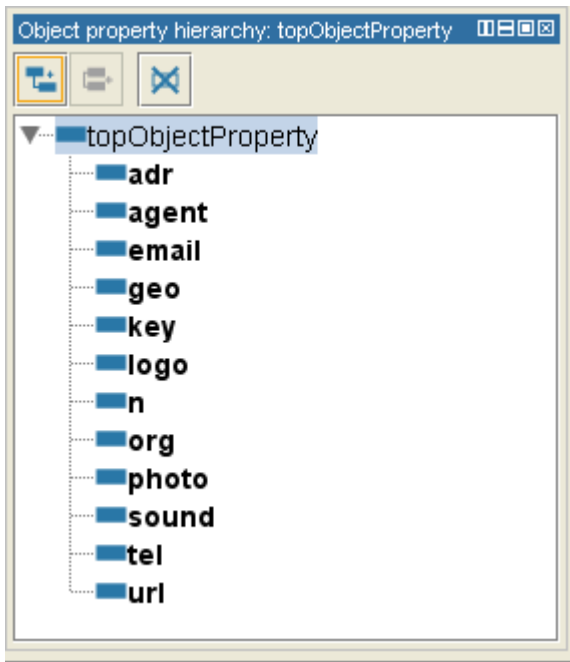


Figure 5.4: vCard Object Properties

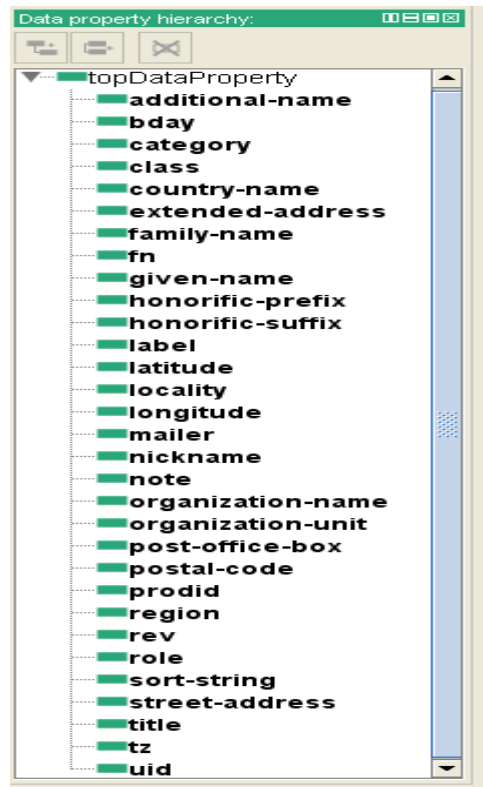


Figure 5.5: vCard Data Properties

The Internet Mail Consortium states that vCard can be used to forward personal data to an electronic mail message. While integrating vCard support into an application, an implementer must consider a number of UI implications. Most applications provide some levels of support for interacting with other applications. RFC2426 mentioned three ways to use vCard the File System, the Clipboard, and Drag/Drop techniques. It also provides two kinds of grouping: first grouping multiple vCard and second grouping related properties within the same vCard. The IMC White paper also argues that full potential of the vCard technology can be better utilized by an application that supports the vCard specification in each of these UI forms.

The grouping of vCard elements provides an ontological structure forming domain ontology. As outlined above, such domain ontology can be used at the structure level to form the logical structure of the GUI. There are also some implicit relations within vCards realized as different groups or sets of attributes.

vCard properties can be grouped and prioritized according to the context and relationships. The values of grouped properties can be represented as one value but with different joining structures and delimiters.

## **5.3. Ontological Framework for User Interface Development**

The ontological framework includes three major sections (i.e. vocabulary, Domain Ontology and User Interface Ontology). These sections are the layers of development procedure. First two layers of development procedure are combined in ontology parser which make a vocabulary table as scope and associated ontology structure. Third layer is the skeleton layer that introduces user interface properties to the elements in vocabulary table and with the arrangement provided in structure layer.

### **5.3.1. Ontology Parser**

This section of our research application parses the ontology in resulting vocabulary table and its structure. This layer read the ontology in any RDF or N3 tripple. Jena is used to read the ontology.

#### *5.3.1.1. Vocabulary (Scope)*

The section Vocabulary contains the concepts and sub-concepts that exist in the domain without stating relationships. It is list of attributes of the domain ontology. vCard standard (RFC2426) provides the list of attributes being used for PIM.

Thought Jena can read and provide the ontology graph and class structure, still it was needed to parse it further to create a vocabulary table

1. Simplify the compound classes and intermediate (anonymous classes) made by the relations like

```

if (class.isUnionClass()){
    s = "unionOf";
}
if (class.isIntersectionClass()){
    s = "intersectionOf";
}
if (class.isEnumeratedClass()){
    s = "oneOf ";
}
if (class.isComplementClass()){
    s = "complementOf";
}

```

2. Specification of associated RDFs: Types with the specified property. These types are defined as the range of the property. These types define the visualization methodology of an `OntProperty` provided in `vCard`

`vCard`, as research example is parsed into a table specifying the elements in a table entry specifying as:

ElementTable is as follows:

Element ID	Element URI	Type	Comments
0	<a href="http://www.w3.org/2006/vcard/ns# vCard">http://www.w3.org/2006/vcard/ns# vCard</a>	OntClass	BaseClass
1	<a href="http://www.w3.org/2006/vcard/ns# Name">http://www.w3.org/2006/vcard/ns# Name</a>	OntClass	
3	<a href="http://www.w3.org/2006/vcard/ns# family-name">http://www.w3.org/2006/vcard/ns# family-name</a>	OntProperty	
4	<a href="http://www.w3.org/2006/vcard/ns# given-name">http://www.w3.org/2006/vcard/ns# given-name</a>	OntProperty	

Properties Table

Element ID	Element URI	Type	DomainID	RangeID
3	<a href="http://www.w3.org/2006/vcard/ns# family-name">http://www.w3.org/2006/vcard/ns# family-name</a>	Data Type Property	1	
4	<a href="http://www.w3.org/2006/vcard/ns# given-name">http://www.w3.org/2006/vcard/ns# given-name</a>	Data Type Property		



ClassTable

Element ID	Element URI	Type	List
6	Date	Unionof	[7,7,7]

vCard has specified date as a combination of three floats.

5.3.1.2. Domain Ontology (Structure)

The section Domain Ontology specifies the relationship of the attributes. It provides a structure for arranging attributes at GUI. In our example vCard provides taxonomy of information (Table 5.2). It provides the way personal information should be arranged and grouped, e.g. first name and last name should be together, city and country information will be in address group. Though it is most of the time arranged in the same way, it is not a rule by the base model in designing and development of the user interface.

<b>Identification Properties</b>	<b>Name</b>	Given
		Family
<b>Address Properties</b>	Street Address	
	City	
	Country	
	Zip Code	
<b>Telephone Number</b>	Country Code	
	Network Code	
	Phone Number	
....		

Table 5.2: Structure of vCard properties

This structure defines the taxonomy as a rule which cannot be avoided in GUI development.

5.3.2. User Interface Properties Mapping:

Next part of user interface designing application facilitates user interface properties association with the domain ontology. These user interface properties are associated to the listed elements filtered through the ontology parser.

This process provides a base skeleton of user interface called here User Interface Model (UIM). UIM is then directly instantiated to a concrete GUI.

### 5.3.2.1. User Interface Model (Skeleton)

Uschold encourages automated code generation for UI from ontological models (Uschold, 2008). But most of the research talks and presents automation and translation of ontologies to GUI (Alexander, et al., 2003) (Furtado, et al., 2002) . Paulheim provides a method to implement ontologies to UI through plug-ins (Paulheim, 2009)

We used the User Interface Ontology (UIO) as one part of the core development process and a base for designing UI, rather than introducing ontologies separately or in parallel to the main design procedure. UIO describes concepts and relationship of GUI objects and interaction methodology. UIO considers that the domain ontology also specifies the representation of each entity in the vocabulary depending on the type of entity, the relationship provided in the domain ontology and the role of the user. UIO considers the role of users to generate GUI according to the context.

UIO contains domain ontology user interface concepts and their properties at the user interface level. Domain ontology defines a vocabulary and a structure. At UIO level, vocabulary is associated with conceptual visualization according to its type (textual, image, multi-media, map or a group) and structure provides arrangements of the visualization at User Interface.

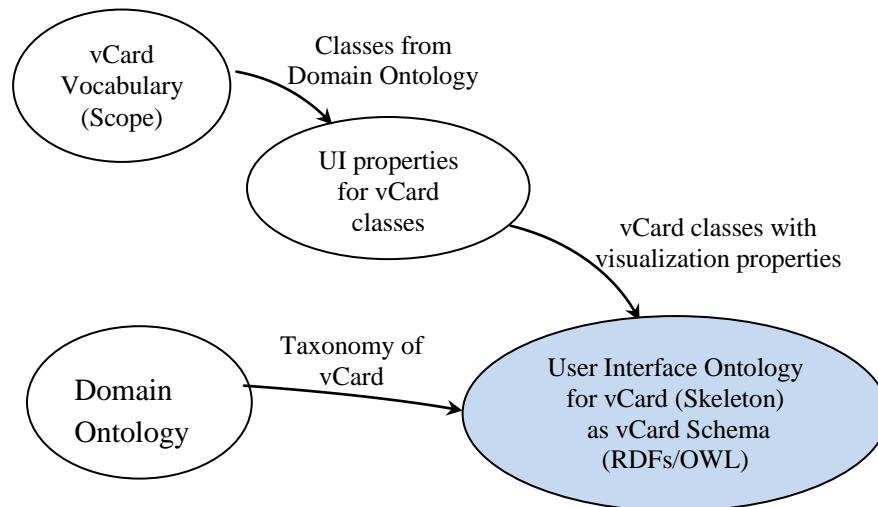


Figure 5.6: User Interface Ontology for vCard (Skeleton)

User Interface properties were added to the Domain Ontology classes carried out in the ontology. These properties specify the visualization mechanism based on the class and type of the instance. Literals, groups, sets and lists can be presented accordingly (Figure 5.6).

In our example of PIM, vocabulary and taxonomy can be read from vCard provided as RDFS. In vCard, Name is a group of two items, “Given and Family”. These sub-parts of names are literals (textual data). So visualization properties can be associated at this small part and class and sub-class. Additional UI properties based on context, like font, colour or size, can also be added later on at skeleton level via the UIO.

UI is two-way communication “from and to the user”. The concepts discussed in UIO are the presentation of a domain concept, sub-concept and the relation to the user as to the user communication; it also discusses the interaction methodology and user actions and ontological responses. Functional Ontology is part of future work and conceptual user actions to concrete event handling will also be part of future work.

### 5.3.3. GUI Development

This is the final process GUI design and development which results in a concrete GUI. This process follows three major steps:

- 1 Take UIM as input
- 2 Instantiate all the UIM elements according to the specified user interface properties through customized user interface controls
- 3 Arrange all the GUI controls according to the structure in one GUI

#### 5.3.3.1. User Interface (surface)

Here we get a simple GUI for vCard where we used Tab pages for groups for a desktop interface. Here we can have choice to select one of the containers to visualize the groups like tab-pages, frames widgets or web-pages depending at the user context

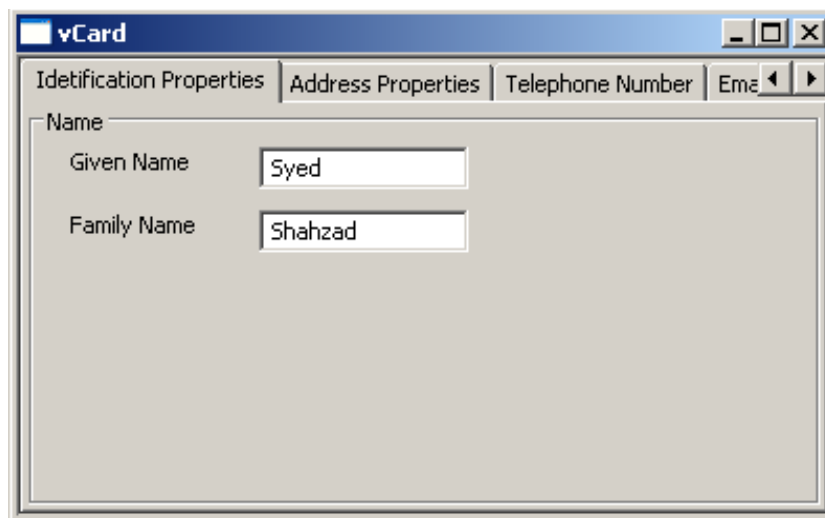


Figure 5.7: Example UI for a vCard

## 5.4. Customized User Interface Control

GUI controls can be customised and associated with the domain concepts which are going to be represented at GUI. So, text box, list box, combo box, radio button or even a frame are associated with a domain concept presented in vocabulary according to their UI properties. IBM Semantic Layered Research Platform<sup>46</sup> has also worked on RDF-driven application development using JFace/SWT components. It provides examples of RDF-driven JFace widget, tooltip window and viewers. Each item within the scope level can be assigned a viewer, widget or container (like frames or tabs for representing a relationship of part and whole).

## 5.5. User Interface Generation

Customized UI controls are then joined together according to the provided skeleton. We did direct mapping from skeleton to graphical objects.

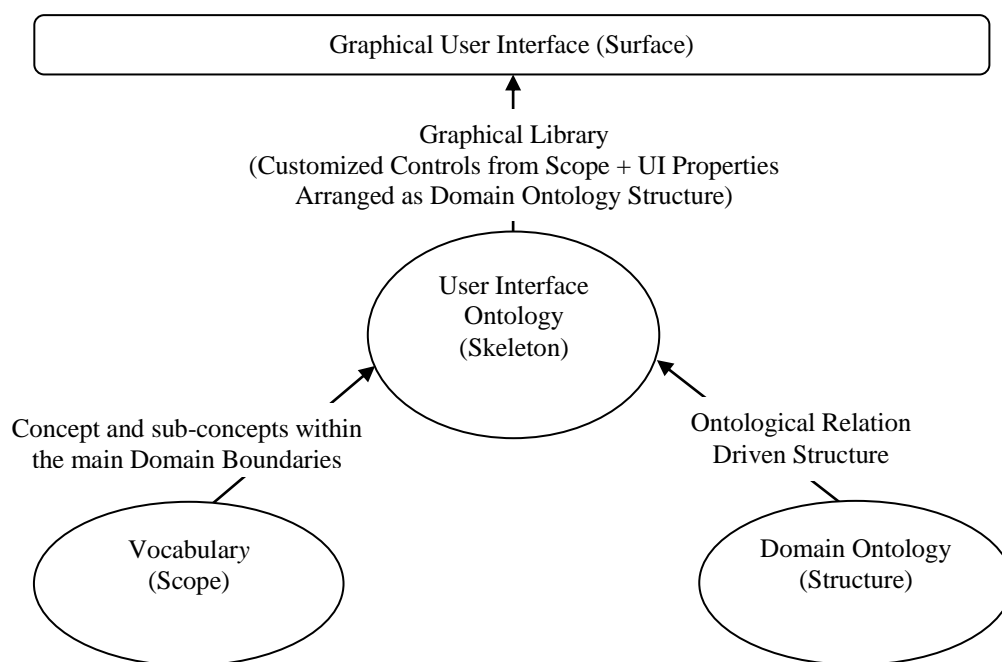


Figure 5.8: Ontology representation at GUI

GUI development is an instantiation of UIO provided as skeleton. Vocabulary (with user interface properties) is instantiated as customized controls. These controls for literals (e.g. textboxes, labels, list box etc.) are arranged in group and provided to containers (e.g. Frame, widget, tab pages, web pages

<sup>46</sup> <http://ibm-slrp.sourceforge.net/wiki/index.php/Com.ibm.adtech.telar.ui.swt>

etc.) This structure of UI objects belonging together provides a complete GUI instance based on Domain ontology. Taking the Garret's model in consideration, the mapping process follows the steps scope (vocabulary), structure (Domain Ontology) and skeleton (User Interface Ontology) (Figure 5.8). Skeleton provides a base design for user interfaces that can be instantiated by any graphical library for any platform.

As an example application of PIM we created a dictionary for vocabulary through reading RDF.

## 5.6. Ontology Modelling based on Context

Domain ontologies structuring and arrangement is based on relationships. In our example of vCard, properties groups and vCard groups are made on the basis of criteria provided by the whole system. vCard is based on the concept of business cards. Thus the arrangement and groups of properties will be different from personal data ontology in social network. Individual properties (like name, address, date of birth etc.) represent the same domain ontology. These domain ontologies are arranged by different rules in an organization than in a social network.

DOLCE ontology has discussed the calculus of individuals in an organization based on their roles. The organizational structure does provide structure for grouping properties and persons.

In a network of friends, level of trust is not dependent on the official designation or department. Social networks provide different criteria to arrange and group properties and persons. All of these structures are discussed and available in relationships, axioms and calculus in current research [ (Bottazzi, et al., 2009) (Gangemi, et al., 2002) (Clarke, 1981) (Gruber, 1993). These structures assist us to develop UI accordingly.

## 5.7. Results

This research experiment resulted nearly similar to the last research experiment in very simple user interface for a vCard, ensuring the structure and semantic classification of each entity of domain ontology presented at user interface. This experiment was more focused on the process and methodology of UI development. There are many alternatives to represent the same concept at surface level. A group of concepts can be aggregated by a group, tab or frame. In this example we group different vCard properties by tabs as containers to show groups. The vCard can be shown in a business card view or as personal data page. Microformats also provide two views at hCard creator; the first

view is a web form for data entry and the second is a preview of a vCard<sup>47</sup>. It is an example for an UI based on modes (i.e. read mode and edit mode). Microsoft Outlook also creates the same kind of UI, but with a more complex structure for vCard edit mode.

Here we get not only a methodology of GUI development through ontology, but it also provides semantically consistent testified, reusable and flexible parts of ontologies at each level of abstraction in the whole process. These parts like vocabulary, structure and skeleton can be verified through logic to ensure a consistent process of GUI designing and development.

## 5.8. Conclusion

Ontological modeling and formalization used in model driven software engineering can make concept specification and representation more consistent. It helps in ensuring proper concept delivery at user interface. A proper layered approach being based on layers of user perception can make the process of GUI design and development more user-centered. User Interface Ontology is a conjunction of the domain ontology, context and look and feel. Website designs and models can also be tested at any stage through formalized concepts, as they carry the concept throughout the development process. In our approach, representation methods were specified as default for textual data. In general, knowledge representation depends on the knowledge domain specified. User interaction and response methods are based on the use and affordances of the knowledge domain. These interaction methods can be specified through ontological functions. In comparison to other methodologies to GUI development for a web-based application, mobile device application or desktop application different approaches are defined and adopted. That makes these development approaches application context dependent rather than concept dependent. Here in this research experiment, the technology or device context is not required till building a skeleton (User Interface model) device. This makes this UIM reusable, flexible and consistent concept delivery at any device. Though context properties can be further added to UIM still these properties are not the base for user interface modeling.

Future work will consider axiomatic constraints or validation functions that can be attached to a model as a part of the ontology rather than through implementation at the GUI level. Further, user Interface properties can form a generic ontology with the UI control vocabulary and relations among them. This UIO can be fused with any domain ontology to introduce UI properties of the domain ontology.

---

<sup>47</sup> <http://microformats.org/code/hcard/creator>

# Ontological Model Driven GUI Development: User Interface Ontology Approach

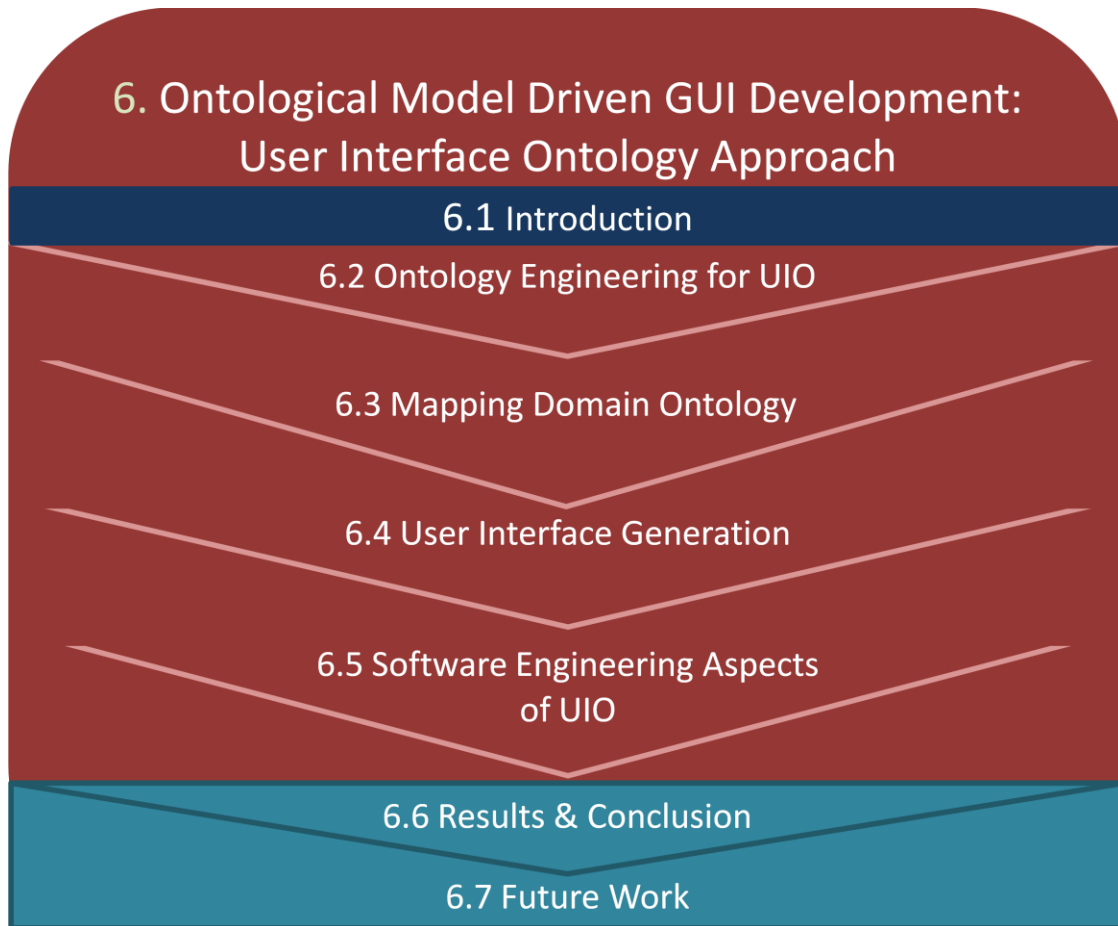
Ontology and Semantic Framework has become pervasive in computer science. It has huge impact at database, business logic and user interface for a range of computer applications. This framework is also being introduced, presented or plugged at user interfaces for various software and websites. However, establishment of structured and standardized ontological model based user interface development environment is still a challenge. In previous experiments added raw user interface aspects to domain ontology but without making a proper ontology for user interfaces. This chapter talks about the necessity of such an environment based on User Interface Ontology (UIO). To explore this phenomenon, this research focuses at the User Interface entities, their semantics, uses and relationships among them. The first part focuses on the development of User Interface Ontology. In the second step, this ontology is mapped to the domain ontology to construct a User Interface Model. Finally, the resulting model is quantified and instantiated for a user interface development to support our framework. This UIO is an extendable framework that allows defining new sub-concepts with their ontological relationships and constraints.

## 6.1. Introduction

Semantics and ontological framework defines computational model for concepts (Guarino, 1995) (Gruber, 1995). This framework can provide a base for modeling concepts representation at user interface to the end users. Uschold (Uschold, 2008), Schlungbaum (Schlungbaum, 1996), Lui (Liu, et al., 2005), IBM<sup>48</sup>, Shahzad (Shahzad, et al., 2010) argued in their research for the model driven user interface development. This research experiment also argues for model based user interface development using semantics and ontological framework. Here, we construct ontology defining some basic user interface classes, properties and their relationships in extendable framework as User Interface Ontology (UIO). In an incremental process, UIO and targeted domain mapping presents the base user interface model. This model is quantified and instantiated to develop a Graphical User Interface (GUI).

---

<sup>48</sup> <http://www.ibm.com/developerworks/library/w-berry/>



*Figure 6.1: 1.6 Organization*

Next section focuses at the motivation behind this research to support our arguments and our work. Third part discusses the UIO development process. Forth section provides the association of UIO with domain ontology. Final section of the chapter describes one of the methods for instantiation and development of the Graphical User Interface (GUI), followed by conclusion and future work.

In this research, an example of vCard as a domain ontology for personal information management system has been used to be represent at GUI. The impetus of this experiment targets the area of modeling and development of a consistent graphical user interface that ensures the semantic properties and constraints defined in the domain ontology.



### 6.1.1. Motivation

Since last decade, Semantic and ontological framework has been widely used for information retrieval and knowledge representation (e.g. Linked Open Data<sup>49</sup> and Web3.0<sup>50</sup>). Various GUI facilitate the representation and visualization of the information retrieved from semantic web at heterogeneous platforms. These GUI have been built in variety of flexible development environments, which allow interface for developers to provide visualization and user interaction mechanism that is disjoint to the semantic relationship and constraints. This phenomenon results in many inconsistent representations of the concepts at similar or different platforms.

Thus, there is a need to explore a methodology that ensures the semantic rules (relationships and constraints) at user interface level for a consistent GUI development in various environments. There is also a need to use these rules for defining data validators, information visualization and user interaction techniques at user interfaces level.

## 6.2. Ontology Engineering for User Interface Ontology (UIO)

Here we discuss in detail about ontology engineering for UIO. In our previous work (Shahzad, et al., 2009) (Shahzad, et al., 2010), we introduced the term of UIO as user interface aspects of computer applications and their mapping to domain ontology. In this work, we build UIO as UI specification to maintain the qualities of UI specification and to ensure proper concepts of user interfaces and targeted domain in GUI development process.

### 6.2.1. Modeling User Interface Aspects

DaSilva (Da Silva, 2001) defines model-based user interface development technology to provide an environment where developers can design and implement user interfaces (UIs) in a systematic way. He also stated three qualities of UI specifications, such as:

1. To model user interfaces using different levels of abstraction;
2. To incrementally refine the models;
3. To re-use UI Specifications

---

<sup>49</sup> <http://linkeddata.org/>

<sup>50</sup> <http://www.w3.org/2001/sw/>

In UIO, User Interface will be the domain concept to be presented through ontology. All of the user interfaces are composed of same GUI controls. End User Interfaces are much more dependent on the targeted domain ontology of the software.

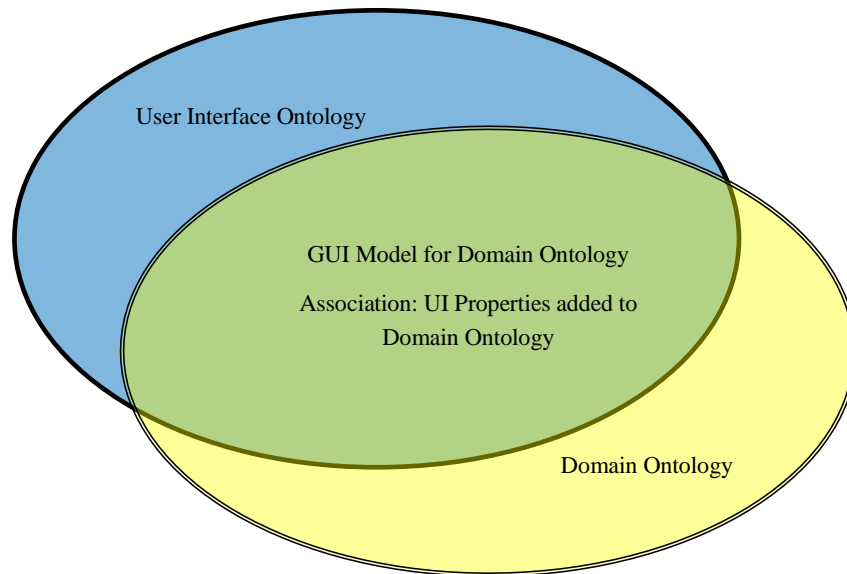


Figure 6.2: Association: Relating Domain ontology with User Interface Ontology

## 6.2.2. Ontology Engineering for User Interfaces

In this stage, we have defined UIO at different level of abstractions to deal with the first property defined by DaSilva (Da Silva, 2001). The Xerox Palo Alto Research Center (Alexander, et al., 2003) and IBM's<sup>51</sup> "The iceberg analogy of usability" discuss three levels of abstraction of user interfaces. Moreover, nearly similar but detailed levels of abstraction for user interfaces have been defined in Garrett's (Garrett, 2002) "The elements of user experience". Garrett's research has been used as level of abstraction for user interface in our previous research. In our approach, we have defined nearly same levels of abstraction for user interface modeling. We have added context dependent properties which can be quantified at the GUI development process. These levels are:

1. Data Modeling
  - a. Data Formats
  - b. Data Structures (Information Architecture)
2. User Interaction Properties
3. Graphical Properties

---

<sup>51</sup> <http://www.ibm.com/developerworks/library/w-berry/>

#### 4. Context aware properties

Data Modeling: Data modeling classes are defined as purely abstract to the implementation details. It specifies only the data domain and its hierarchy. For the research purpose we deliver a paradigm based on ontological classes for data formats from the set controls and user events applied in java (SWT<sup>52</sup>). It provides a base data model which has to be represented at user interface with specific user interaction methods and graphics detail.

Classes and sub classes made for data format and data structures criterion:

1. Visualization Classes:
  - a. Group
    - i. Widget
      - i. Frames
      - ii. Tabs/Pages
  - b. Textual
  - c. Temporal
  - d. Multimedia
    - i. Picture
    - ii. Audio
    - iii. Video
2. Data Structure based Classification:
  - a. Single Entity
  - b. List
  - c. Tuple
  - d. Table-Grid

Visualization classes characterize the representation methodology of a concept, depending of its semantic class and in our example RDF/XML data type. Data Structure based Classification specify the information architecture at user interfaces. Mereology also helps user interface designing by providing the relations to give a consistent representation of concept in their semantic groups.

User Interaction Properties: At lower level of abstraction to data modeling classes, we have defined user interaction properties. These properties are dependent to the data modeling classes which specify the user interaction methodology for a specific data model in general or in specific architecture.

Graphical Properties: At next level of abstraction to user interaction properties, we have defined graphical properties. These properties are also dependent at data modeling classes.

---

<sup>52</sup> <http://www.eclipse.org/swt/docs.php>

Context aware Properties: At the lowest level of abstraction, we defined context aware properties. These properties are also dependent at data modeling properties and user environment specification. This specification consists of technological details and user information.

This ontology was made in OWL-DL and RDF as meta-data for the GUI. There were also some rules defined as basic user interface rules that specify relationships. For instance, a Group is a container that is a set of other classes like textual-data, lists and/or other groups. UIO also provides constraints that ensure consistency at user interface, for instance e.g. duration property (property of audio/video format) cannot be associated to textual data. This ontology is extendable to add more features and aspects related to UI.

### 6.3. Mapping Domain Ontology with UIO

UIO mapping to domain ontology is performed in two major steps. Firstly, data modeling classes from UIO are associated with the domain ontology properties to provide a structure for information visualization and architecture. Secondly, user interaction and graphical properties from UIO are added to the mapping. Any user interface concept specified in UIO is linked to the domain ontology properties in three ways:

1. Link to all properties having same range
2. Link to all properties having same domain
3. Link to the specific relation (Reification)

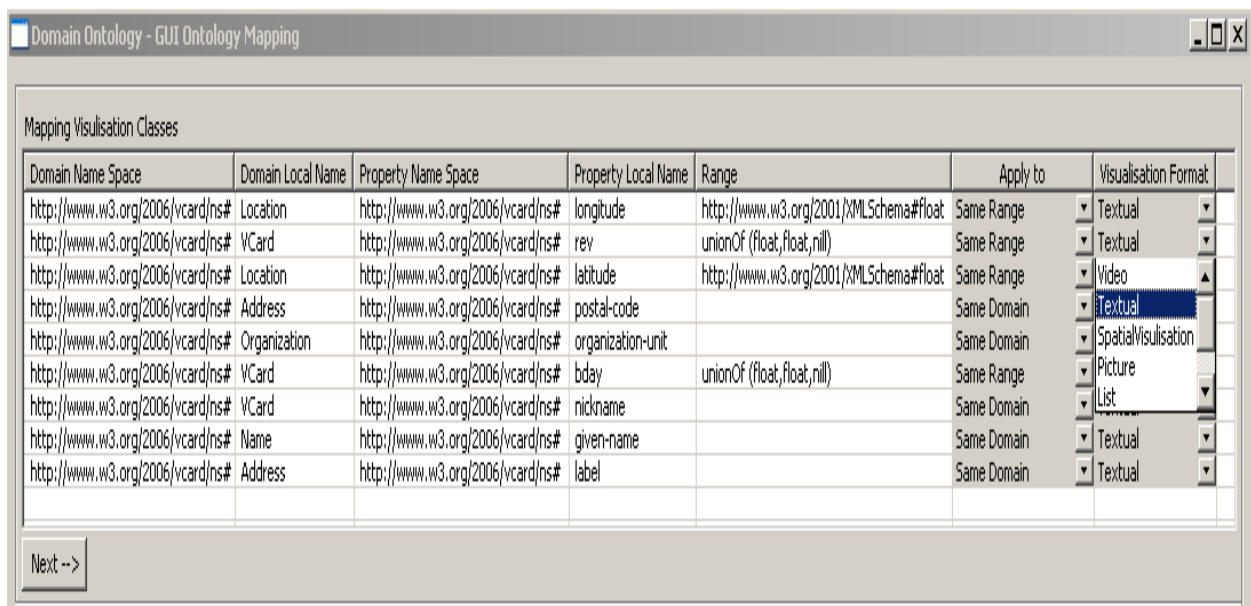


Figure 6.3: Portion of mapping application screenshot

This mapping proceeds through the UIO levels of abstraction from top to bottom.

### 6.3.1. Mapping Visualization Classes

At the first step of mapping process, representation based classes are mapped to the properties of target domain ontology to specify the visualization method. It is mostly dependent on the range class of the entity that specifies the data format, but the procedure also allows mapping based on domain or to the specific relation of the domain and range.

All next level properties are mapped according to the representation specified in visualization class mapping to any domain ontology property.

### 6.3.2. Mapping User Interface Properties

User interface properties are mapped to domain ontology properties depending on their visualization classes, e.g. Editable is a user Interface property for Textual data, while multi-selection can be a property of list or a Table-Grid. UIO provided rules ensure semantic association of user interface properties to every data type. UIO also ensure data consistency at quantifying these properties e.g. if a container is disabled for user interaction and performing any operation all entities within the container will be disabled as well. Rather than depending at graphical libraries, semantic rules are defined in UIO and carried to GUI for consistent representation.

### 6.3.3. Mapping Graphical Properties

Graphical properties are mapped to domain ontology for providing a structure of interface entities. It can also map properties which can be quantified at the time of GUI development. These properties has to provide relative scaling of user interface entities in mapping process e.g. font sizes for title, headings and body text. These relative scaling collectively provide a theme or representation styles.

### 6.3.4. Mapping Context aware properties

These properties are mapped to the domain ontology properties but can be quantified only at GUI development process.

### 6.3.5. User InterfaceModel (UIM)

This mapping of DO and UIO results in a User Interface Model (UIM). This Model is base for the user interface generation. Similarly, the main target of this research is also to create such a technology and context independent model for any given domain concept representation.

Semantic knowledge representation model contains a combination of semantic rules (relationship and constrains) from UIO and domain ontology that ensure:

- 1 Proper concept representation at GUI
- 2 Consistent concept representation
  - a. By GUI developed with different technologies
  - b. At heterogeneous platforms

### 6.3.6. vCard Ontology for Personal Information Management

In this stage, we continued working on the same Personal Information Management case study/example 5.2.2. vCard is a standard and widely adapted (Apple's Address book, MS Outlook, TUGraz Staff and Student<sup>53</sup>, Mobile Phones) ontology. It is defined by Internet Mail Consortium (IMC)<sup>54</sup> and Personal Data Interchange as vCard (RFC2426). This experiment focuses on factors and issues in mapping of vCard to GUI. vCard namespace<sup>55</sup> provides a vocabulary and schema of a vCard. This schema can be associated with UIO according to the RDF/XML data type entities.

Apparently, the presented model for semantic description of associations looks complex, however, the ontology engineering software will not allow making any inconsistency in ontology. Introducing new relations and adding new relations can only be done by HCI expert and Ontology engineer. We have done manual association in this study, such as associating First name with a textual class. It is like making a relations or RDF statement for each relations. We have added UI aspects as objects like

[<http://www.w3.org/2006/vcard/ns#Name>, Displayby, "UIO/ns#Textual"]

Futhermore, some UI aspects were added as to be quantified at later stage such as:

[<http://www.w3.org/2006/vcard/ns#Name>, Font, "0"]

## 6.4. User Interface generation

### 6.4.1. Quantifying Context aware properties

Before proceeding to the instantiation we need to quantify the UI aspects which are dependent on the context or technology. Introduction of context also make the user interface sketch more detailed and concrete. These context variables depend at the device for which GUI is going to be made e.g. iPhone,

---

<sup>53</sup>

[https://online.tugraz.at/tug\\_online/visitenkarte.show\\_vcard?pPersonenId=73A4CED7D8FE1923&pPersonenGruppe=3](https://online.tugraz.at/tug_online/visitenkarte.show_vcard?pPersonenId=73A4CED7D8FE1923&pPersonenGruppe=3)

<sup>54</sup> <http://www.imc.org/pdi/>

<sup>55</sup> <http://www.w3.org/2006/vcard/ns#>

desktop or web application. Information architecture, and navigation remains same for this device but size and wrapping can be different depending on the context.

### 6.4.2. Instantiating UIM

All Domain ontology associated with UIO that come up with a complete sketch of user interfaces in RDF/XML format (UIM). Any graphical library can be used to instantiate the UIM. We have used SWT (Java) for UIM instantiation of vCard. By using RDF/XML, UIM facilitates to represent the data in a simple text format without making any GUI.

## 6.5. Software Engineering Aspects fo UIO

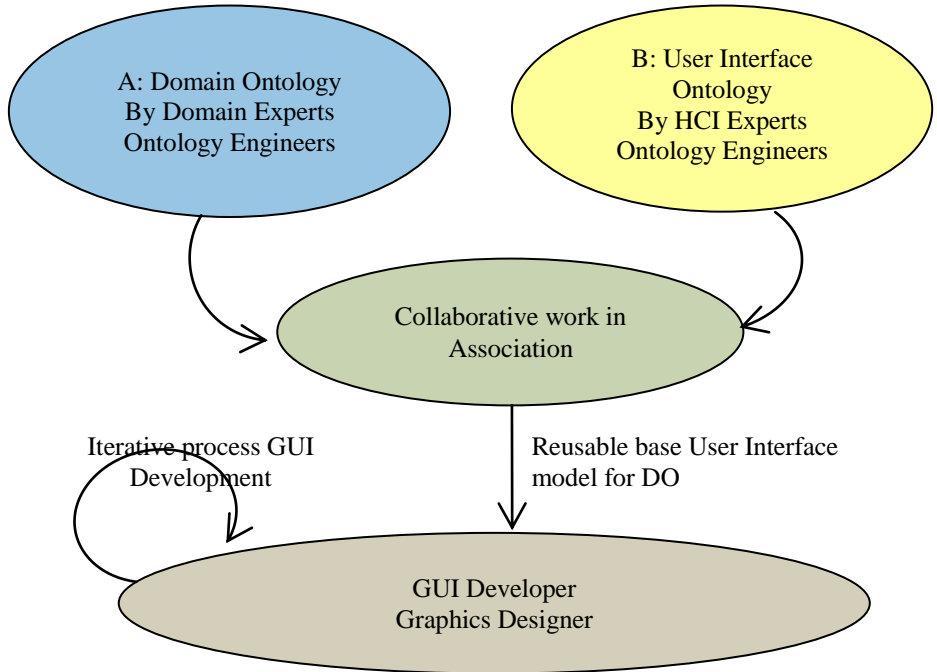


Figure 6.4: Knowledge Experts contribution breakdown and collaboration stages

This model also separates the individual and collaborative work knowledge experts. It is also an economical solution for software development firms. However, every firm cannot bear the cost of HCI consultants for each of the software development. There are mostly reusable libraries are made for some specific domain. That does not fit or get associated with every domain. Secondly, all HCI concerns are done only for customized or specific software.

- 1 In Figure 6.4 HCI expert build at B. UIO experts and Domain expert can work at A. These ontology are

- a. Independent of context details
  - b. Independent to any technical and implementation details
  - c. Reusable for association with any domain
  - d. Extendable with new vocabulary
  - e. Shared, Standardized also extendable by other skilled resources
2. At association level a sketch can be made for a user interface
    - a. Independent to the technology details
    - b. Context can be introduced
    - c. Association is done at this layer considering basic requirement of each knowledge domain.
    - d. Provide reusable User Interface Model (UIM) as a base for GUI development for the associated domain ontology
  3. At GUI generation level GUI developer and graphics complete sketch of the picture which they can colour with any graphical library, themes style-sheets etc. At the end a GUI is instantiated from this level which
    - a. All Technical aspects are considered rather than design aspects
    - b. Multiple GUI can be created with the same sketch for different devices which keeps a consistency.

## 6.6. Results

This research focused at process of user GUI and consistent concept delivery to the end user through GUI. The end result of the process is just a normal GUI for a vCard, but main targeted result is UIM - stated as RDF/XML files containing the tags of domain ontology and UIO. It is like a complete ready recipe just to put in machine to start working. Further results show that UIO mapping process read vCard schema from RDF file. The intelligence of this research was UIO engineering and mapping of UIO that can be applied to any domain ontology.

## 6.7. Conclusion

Rather than using raw user interface development guideline, it was evident from our work that user interfaces properties and their relationships can be defined through semantic and ontological framework as UIO. We also illustrated that UIO mapping with any domain ontology gives an abstract base model for user interfaces for the domain ontology. The long term goal of this experiment is to demonstrate a model guide and to restrict user Interface developer to develop any semantically consistent GUI for heterogeneous development and execution environment.



## 6.8. Future Work

It is increasingly observed that research results usually come up with new questions. Similarly, this work attempted to recommend some approaches and model to the development of ontological model based user interface, however, many questions still have to be explore in this domain. We are still working on to use UIO for different standard domain ontology which is extending UIO for many new formats of data and properties and their relationships e.g. multimedia properties, vector data. There are more areas to explore in this regard such as:

- There is a big question of usability for resulting GUI which will explored in future research.
- There is also need to discuss functional ontology for domain ontology and UIO (user actions and events)
- There is also a need to make some base UIO and browser which can read and display UIM.
- AI and Interactive learning from end users can make a mechanism for auto extend and align UIO.

# Results and Conclusion

## 7.1. Introduction

This part presents the experiments done for GUI development based on ontology and semantic framework. Our research work consisted of three different major methodologies to achieve GUI development based on semantic and ontological framework.

We made some baseline logical principals for GUI designing and development process to ensure a consistent delivery of semantics and ontology concern at user interface:

- 1 Semantic and Ontology framework provides concept representation with following properties
  - a. GUI will be developed on the technology independent base user interface model (skeleton) deduced from domain
  - b. Domain ontology will without adding new rules to domain ontology except additive constraints depending on context before development of GUI
  - c. At base model, Represented knowledge classification and structure at GUI should contain the semantic and ontological properties and relationships to keep it complete.
- 2 There should not be any information structure at user interface model that contradicts any domain ontology relationship and constrains to keep it consistent
- 3 A reasoning engine should validate the structure and classification of entities represented at user interface to ensure (i) and (ii)
- 4 There should be direct mapping of domain ontology driven User Interface model to the GUI to make an exact visual image of domain ontology
- 5 Ontology Driven GUI development will be done in following three phases
- 6 Adding semantically defined GUI properties and constraints according to semantic data formats and classification
- 7 Adding semantically defined GUI properties and constraints without according to the user and context without voiding any domain ontology rule, thus verified by reasoning engine
- 8 Instantiation of base model (merger of domain ontology with visualization semantics) to concrete GUI by direct and one to one mapping of base model components to Graphical library objects.
- 9 Data formats, ranges and information structure of the mapped graphical components at GUI.

Other than these specified rules the data validation should be possible according to the ontology and semantic classification.

## 7.2. Functional programming approach

With our first experiment discussed in 1.4, we have used a functional programming approach to directly code mathematical model to software programming language. We made up example ontology of vector graph as a graph (id, node, edge) function containing other functions like node and edge stating their relationships and data ranges.

We used an example of a Graph (Node, Edge) a function of Nodes and Edges. We have used high order functions on abstract level seems as

```
gui (graph) → IO()
```

gui is the main function take graph functions as input and produce gui as IO() function. These experiments results in a simple gui consists of a panel for data input output and canvas for graph visualization.

As the first experiment of the research stream it was aimed to for this experiment is to construct a methodology that provides a consistent delivery of ontologically define concepts to user interfaces. Here bijective morphisms and higher order functions ease the task to develop a gui directly from mathematically defined ontological functions.

## 7.3. Impreative Programming Approach

In our second experiments discussed in 1.5, standard vCard ontology is used to develop GUI. Java as an imperative programming language with a semantic application development library Jena and SWT graphical library are used in this experiment. As an extension to previous research experiments a more detailed layered structure for GUI modeling and development have been used with reference to the on iceberg analogy and Garret's user experience elements model shown in Table 5.1: Adapted User Experience Elements by Garrett . Here we also ensure semantic and ontological constraints delivery to user interface from abstract layer to concrete GUI. This experiment ensures semantic consistency at each layer of process and user interface design and model.

### 7.3.1. Experiment Structure

This experiment consists of three important phases for GUI design and development. First two phases are ontology parsing and user interface model generation are part of GUI designing process that is independent of context and technology details. While the third process of GUI development requires context and technology details for GUI development.

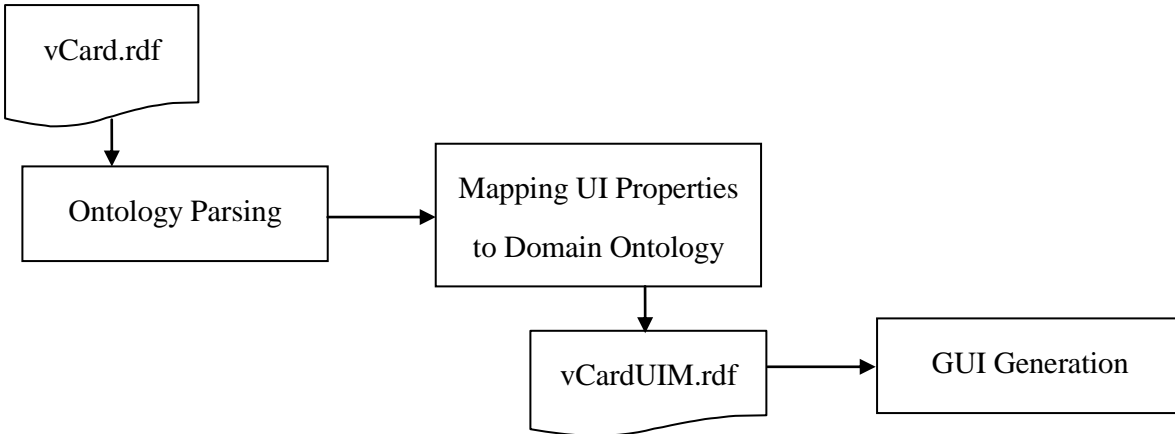


Figure 7.1: Major steps for User Interface Design and Development

### 7.3.1.1. *Ontology Parsing*

At the first part of the experiment the ontology parser parse the domain ontology to buildup an entities table as vocabulary dataset and ontology structure as taxonomy of the ontology. This parser use Jena API for parsing domain ontology.

### 7.3.1.2. *Mapping UI properties to Domain Ontology*

Second step is associating visualization properties with the entities vocabulary accordingly. These properties specify the visualization method for all of the properties that has to be displayed at user interface it is always assisted by semantic data types “*RDFS:Datatype*” defined in domain ontology. User Interface properties are added according to the following steps to in the fashion of adding layers

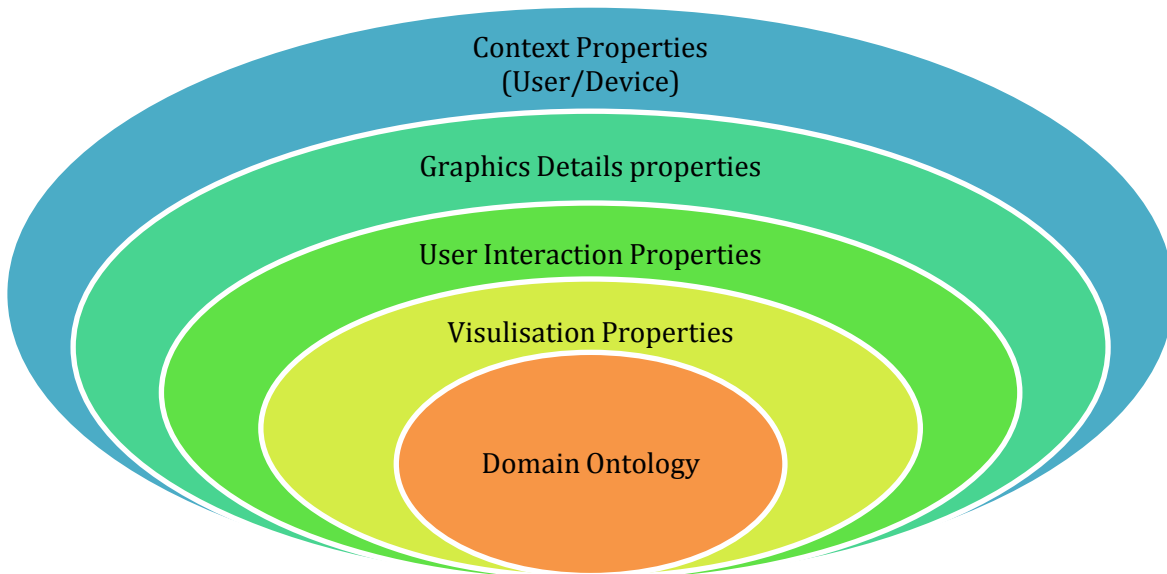


Figure 7.2: User Interface Model Layers

These additional user interface properties to the vocabulary are added to the domain ontology according to the taxonomy. This mapping results in a “*User Interface Model (UIM)*” for the domain ontology which carries the relationship and constrains from domain ontology. All of these properties are quantified at user interface generation phase.

### 7.3.1.3. *GUI generation*

User Interface generation also use the ontology parser. It builds a user interface with specified user interface properties in user interface model. Properties, which are part to the same class, (mereotopology) are presented in one container. Here UI developer makes choices of using one container type (we used Tab groups) to group visualize these classes. Editable text entries are mapped to text boxes while non-editable text entries are mapped to labels.

## 7.4. User Interface Ontology

In this experiment we have used nearly the same design and development methodology as we have used in previous experiment except introduction of User Interface Ontology (UIO). Here we have worked to develop a UIO considering same layers approach. We have user OWL DL to specify these classes and properties. We have used the common WIMP environment elements and behavior to introduce them in ontology. This structure is in three major classes

### 7.4.1. Ontology Engineering

#### 7.4.1.1. *Visualization Properties*

These classes and properties provide ontological structure of data visualization according to the RDFS data types there are two views for classification of data visualization based on:

- 1 Data Structure
- 2 Data Types

#### 7.4.1.2. *User Interaction*

User Interaction classes and properties are user context dependent properties that can associated to the domain ontology but quantified at GUI generation phase. In our example we have used some user interaction methods like Editable, Movable and Drag-Drop.

These properties are dependent to data types (or Visualization classes).

### 7.4.1.3. *Graphical Properties*

Graphical properties are also dependent to the data type and device context. These properties can be associated in mapping procedure. User Interface developer can quantify these properties at GUI generation phase according to the device context.

## 7.4.2. GUI Generation

The companies SystemeticBytes<sup>56</sup> and Experflow Pvt Ltd<sup>57</sup>. testing the methodology have developed GUI using JFace/SWT for GUI development. The Expert Flow team has made a Contacts Management tool (Telephone Directory) for as a part of their client's web interface of telecommunication solution. They have added some other audio properties for providing voice dialing facility. Systematicbytes have tested this methodology for developing a Contact Directory for desktop, web and mobile application.

They have introduced some common GUI features

### 7.4.2.1. *Mandatory Field*

vCard ontology doesn't provide or specify any mandatory or identity fields. At UIM development phase the testing teams has introduced the identity field and some compulsory fields added like Name and any one of the contact information. For the phone directory it was the phone number as compulsory to have data

### 7.4.2.2. *Edit Mode*

They also have introduced an edit mode to add new vCard entries or edit current vCard data.

### 7.4.2.3. *User Dialogues*

User Dialogues are provided to upload or download vCard as vCard file

### 7.4.2.4. *List View and Thumbnails of vCard*

In a directory with a list of vCard there were two mode of view one is detailed view showing the vCard data details and other is a thumbnail view of all listed vCard entries in the directory.

The testing teams have made multiple GUI solution with some of their commercial GUI development libraries that allows customization and association to the RDFS data type much easier. Systematicbytes.com team has reused the UIM to develop three interfaces for different platforms.

---

<sup>56</sup> <http://systematicbytes.com/index.html>

<sup>57</sup> <http://www.expertflow.com/index.php>

## 7.5. Conclusion and Future Work

### 7.5.1. Research Targets

In our research work we have targets to explore the methodologies for user interface development that can ensure semantic and ontological concerns at user interface. There we have started with logical solutions and promote that solution for standard ontologies provided as shared standard.

We used ontology as based model of the process of designing and development of user interface and user interface itself. Building user interface over the domain ontology ensures the process and product dependency at ontology rather than technology (development tools or devices). Moreover these can be validated by any reasoning software at any stage of designing and development. These themes and validation procedures provide us the grounds to experiment and explore the methodologies. We have also formalized research questions in 1.2.1 on the same grounds.

### 7.5.2. Experiment Results outcomes

In our experiments we started with to build a GUI based on domain ontology. Gradually we introduced semantic and ontological consistency in depth to each step of the procedure (designing and development) and product (GUI).

- 1 First experiment was aimed to provide a mathematical modeling based GUI development to verify the ontology based GUI development. Though it doesn't use any standard ontology, but it provides mathematically defined example of graph containing nodes and edges. Haskell programming language is used that can directly code the mathematical equations to the programming language without any additional translations or any intermediate layers.

Domain Ontology  $\rightarrow$  Domain Ontology<sub>Haskell</sub>  $\rightarrow$  IO (Domain Ontology<sub>Haskell</sub>)

This experiment provides the result of our first research question that *“How can semantically defined concepts be delivered at GUI?”*

2. Still there was need to apply this solution to the standard ontology. That we have experiments in our second experiment using vCard as standard ontology we also introduced a base model as User Interface Model to develop a GUI over it. This base model is specified in RDF/XML that can be validated for any semantic and ontological consistency. Here we introduced user interface aspects to the domain ontology with a semantic relationship using RDFS. We explored the second and forth research question in this experiment i.e. *“Can Semantics and Ontological frameworks provide a based model for GUI?”*

Instantiation of UIM to the GUI using customized user interface controls provide answer for the fourth research *“How a base model for UI can be instantiated to GUI ensuring that all IO Operation preserves semantic properties and allow data flow according to the semantic data classification using:”*

3. Still we need to explore the methodology to make it consistent at all the levels, in second experiment we added some classes and properties according to the RDFS data types but there was no specific semantic introduction to these classes and properties. These UI concerns are properly introduced in our fourth experiment that introduced User Interface Ontology. This ontology ensures semantic consistency to both areas that forms UIM. Here get answers to the question regarding consistency of the process and product i.e. *“How can a base model for UI contain and Domain Ontology relations and constraints?”* and *“How a base model for UI can be semantically consistent?”*

### 7.5.3. Future Work

As the introduction to a new methodology regarding user interface development, our research opens many new questions about the methodology and related fields. Some of the fields which need to be explored are directly connected to our field of research are as:

1. This research explored the area of semantically consistent process of user interface design and development without discussing issues of usability. The most important issue is to discuss usability issues regarding ontology driven GUI development.
2. There is a need to build some smaller parts of the computational libraries that we have developed on a small scale for experimentation like
  - a. Graphical Library associated to semantic data types that can enable communication from user to machine in semantically defined and validated data values.
  - b. More generic and big UIO need to be established that can be used to make a UIM for more standard ontologies.
  - c. Recommended UIM attached with the standard ontologies should be provided by the knowledgebase as recommendations for the visualization of the specific ontology.
  - d. The universal browsers are needed which have ability to provide a user interface for standard ontology and their specified UIM.
3. There is also a need to explore ontologies of the functions that can be mapped to the recommended user actions (GUI events) in UIM as well as at GUI.
4. In addition, there is a need to evolve a proper methodology for ODIS design and development life cycle.



# Bibliography

**Abrial Jean-Raymond** Data Semantics [Conference]// IFIP Working Conference Data Base Management. - 1974. - pp. 1 - 60.

**Alavi Maryam and Leidner Dorothy E.** Knowledge management systems: issues, challenges, and benefits [Journal] // Commun. AIS. - [s.l.] : {Association for Information Systems, 1999. - 2 : Vol. 1.

**Alexander Kleshchev and Valeriya Gribova** From an ontology-oriented approach conception to user interface development [Journal]// Information Theories & Applications. - [s.l.] : Institute of Information Theories and Applications FOI ITHEA, 2003. - 1 : Vol. 10. - pp. 87 - 93. - ISSN: 1313-0463.

**Aljawarneh Shadi and Alkhateeb Faisal** Design and Implementation of New Data Validation Service (NDVS) Using Semantic Web Technologies in Web Applications [Conference]// World Congress on Engineering 2009. - London : [s.n.], 2009. - ISBN: 978-988-17012-5-1.

**Almeida Mauricio, Souza Renato and Fonseca Fred** Semantics in the Semantic Web: A Critical Evaluation [Journal] // Knowledge Organization Journal. - 2011. - 3 : Vol. 38. - pp. 187 - 203.

**Bastien J. M. C. and Scapin D. L.** How usable are usability principles, criteria and standards ? [Journal] // Advances in Human Factors/Ergonomics, Volume , , Pages . - [s.l.] : Elsevier, 1995. - Vol. 20. - pp. 343-348.

**Bates Marcia J.** The Design of Browsing and Berrypicking Techniques for the Online Search Interface [Journal] // Online Review. - Los Angeles, CA : Graduate School of Library and Information Science, 1989. - 5 : Vol. 13. - pp. 407 - 424. - <http://pages.gseis.ucla.edu/faculty/bates/berrypicking.html>.

**Bauersfeld Penny F. and Slater Jodi L.** User-oriented color interface design: direct manipulation of color in context [Conference]// Proceedings of the SIGCHI conference on Human factors in computing systems: Reaching through technology. - New Orleans, Louisiana : ACM, 1991. - pp. 417 - 418. - doi: <http://doi.acm.org/10.1145/108844.108981>.

**Beaudouin-Lafon Michel** Instrumental Interaction: An Interaction Model for Designing Post-WIMP User Interfaces [Conference]// Proceedings of the SIGCHI conference on Human factors in computing systems. - The Hague : ACM, 2000. - pp. 446 - 453. - doi: 10.1145/332040.332473.

**Berry Richard E.** Common User Access—A consistent and usable human-computer interface for the SAA environments [Journal] // IBM Systems Journal. - [s.l.] : IBM, 1988. - 3 : Vol. 27. - pp. 281 - 300. - doi: 10.1147/sj.273.0281.

**Boehm Barry W.** A spiral model of software development and enhancement [Journal] // ACM SIGSOFT Software Engineering Notes. - [s.l.] : ACM, August 1986. - 4 : Vol. 11. - pp. 14 - 24. - doi: <http://doi.acm.org/10.1145/12944.12948>.

**Bonatti P., Deng Y. and Subrahmanian V.S.** An ontology-extended relational algebra [Conference] // Proceedings of IEEE International Conference on Information Reuse and Integration, 2003. IRI 2003.. - Las Vegas : IEEE Systems, Man, and Cybernetics Society, 2003. - pp. 192 - 199. - doi: [10.1109/IRI.2003.1251413](http://doi.acm.org/10.1109/IRI.2003.1251413) .

**Bottazzi E. and Ferrario R.** Preliminaries to a DOLCE: Ontology of Organizations [Journal] // International Journal of Business Process Integration and Management (IJBPM). - [s.l.] : Inderscience Enterprises Ltd, 2009. - 4 : Vol. 4 . - pp. 225 - 238. - doi: [10.1504/IJBPM.2009.032280](http://doi.acm.org/10.1504/IJBPM.2009.032280).

**Brey Philip** The Epistemology and Ontology of Human-Computer Interaction [Journal] // Minds and MACHines. - [s.l.] : Springer Netherlands, November 2005 . - 3 - 4 : Vol. 15. - pp. 383-398. - doi: [10.1007/s11023-005-9003-1](http://doi.acm.org/10.1007/s11023-005-9003-1).

**Brockmans Saartje [et al.]** Visual Modeling of OWL DL Ontologies Using UML [Conference] // Proceedings of International Semantic Web Conference. - [s.l.] : Springer, 2004. - pp. 198 - 213. - doi: [10.1007/978-3-540-30475-3\\_15](http://doi.acm.org/10.1007/978-3-540-30475-3_15).

**Buxton W. [et al.]** Towards a comprehensive user interface management system [Conference] // Proceedings of the 10th annual conference on Computer graphics and interactive techniques, SIGGRAPH '83. - Detroit : ACM New York, NY, USA ©1983, 1983. - doi: [10.1145/964967.801130](http://doi.acm.org/10.1145/964967.801130).

**Car Adrijana and Frank Andrew U.** Formalization of conceptual models for GIS using Gofer [Journal] // Computers Environment and Urban Systems. - [s.l.] : Elsevier, 1995. - 2 : Vol. 19. - pp. 89 - 98. - doi: [10.1016/0198-9715\(95\)00013-X](http://doi.acm.org/10.1016/0198-9715(95)00013-X).

**Cardelli Luca and Wegner Peter** On Understanding Types, Data Abstraction and Polymorphism [Journal] // Computing Survey. - December 1985. - 4 : Vol. 17. - pp. 471 - 522. - doi: [10.1145/6041.6042](http://doi.acm.org/10.1145/6041.6042).

**Carlsen Niels Vejrup** Towards a Common Context for User Interface Management System Design [Conference] // Proceedings of the IFIP TC2/WG2.7 Working Conference on Engineering for Human-Computer Interaction. - Amsterdam : North-Holland Publishing Co., 1992. - pp. 13 - 34. - ISBN: 0-444-89904-9.

**Carroll Jeremy J. [et al.]** Jena: Implementing the Semantic Web Recommendations [Conference] // Proceedings of the 13th international conference on World Wide Web - Alternate Track Papers. - New York : ACM, 2004. - pp. 74 - 83. - doi: [10.1145/1013367.1013381](http://doi.acm.org/10.1145/1013367.1013381).

**Carroll Jeremy J. [et al.]** Named graphs, provenance and trust [Conference] // Proceedings of the 14th international conference on World Wide Web. - Chiba : ACM, NY, USA, 2005. - pp. 613 - 622. - doi: [10.1145/1060745.1060835](http://doi.acm.org/10.1145/1060745.1060835).

- Chen Peter Pin-Shan** The entity-relationship model—toward a unified view of data [Conference] // Special issue: papers from the international conference on very large data bases. - Framingham : ACM, New York, NY, USA, 1975. - pp. 9 - 36. - doi: <http://doi.acm.org/10.1145/320434.320440>.
- Clarke Bowman L.** A calculus of individuals based on connection [Journal] // Notre Dame Journal of Formal Logic. - [s.l.] : Project Euclid, 1981. - 3 : Vol. 22. - pp. 204 - 218. - doi: 10.1305/ndjfl/1093883455.
- Cocchiarella Nino B.** Conceptual Realism as Formal Ontology [Book Section] // Formal ontology / book auth. Roberto Poli Peter M. Simons. - [s.l.] : Kluwer Academic Publishers, 1996.
- Corry Michael D., Frick Theodore W. and Hansen Lisa** User-Centered Design and Usability Testing of a Web Site: An Illustrative Case Study [Journal] // Educational Technology Research and Development, . - 1997. - 4 : Vol. 45. - pp. 65 - 76.
- Cranfield Stephen and Purvis Martin** UML as an ontology modelling language [Report]. - [s.l.] : University of Otago.
- Da Silva Paulo Pinheiro and Paton Norman W.** User interface modeling in UMLi [Journal] // IEEE Software. - [s.l.] : IEEE Computer Society, July-Aug 2003. - 4 : Vol. 20. - pp. 62 - 69. - doi: 10.1109/MS.2003.1207457.
- Da Silva Paulo Pinheiro** User interface declarative models and development environments: a survey [Conference] // DSV-IS'00 Proceedings of the 7th international conference on Design, specification, and verification of interactive systems. - Limerick : Springer-Verlag, 2001. - pp. 207 - 226. - ISBN: 3-540-41663-3.
- Decker Stefan [et al.]** Ontobroker: Ontology Based Access to Distributed and Semi-Structured Information [Conference] // Proceedings of the IFIP TC2/WG2.6 Eighth Working Conference on Database Semantics- Semantic Issues in Multimedia Systems. - Deventer : Kluwer, B.V., 1998. - pp. 351 - 369. - ISBN: 0-7923-8405-9.
- Dotsika Fefie** Semantic APIs: Scaling up towards the Semantic Web [Journal] // International Journal of Information Management. - August 2010. - 4 : Vol. 30. - pp. 335-342. - doi:10.1016/j.ijinfomgt.2009.12.003.
- Dumas Joseph S. and Redish Janice** A practical guide to usability testing [Book]. - Norwood : Ablex Publishing Corporation, 1993. - ISBN: 9781841500201.
- Dzida Wolfgang** Standards for user-interfaces [Journal] // Computer Standards & Interfaces. - [s.l.] : Elsevier, 1995. - 1 : Vol. 17. - pp. 89-97.
- Eason K. D.** Information Technology And Organisational Change [Book]. - London : Taylor & Francis., 1988. - ISBN: 978-0850663914.

**Elmasri Navathe** A theory of attributed equivalence in databases with application to schema integration [Journal] // IEEE Transactions on Software Engineering. - [s.l.] : IEEE Computer Society, 1989. - 4 : Vol. 15. - pp. 449 - 463. - doi: 10.1109/32.16605.

**Eng Eirik** Qt GUI Toolkit: Porting graphics to multiple platforms using a GUI toolkit [Journal] // Linux Journal. - Houston, TX : Belltown Media, 1996. - 31. - ISSN: 1075-3583.

**Engelbart Douglas C.** X-Y Position Indicator For a Display System [Patent] : 3541541. - USA, November 17, 1970. - Categories: 345/164; 33/775; 178/18.01; 340/870.07; 340/870.44.

**Fayyad Usama, Piatetsky-Shapiro Gregory and Smyth Padhraic** From Data Mining to Knowledge Discovery in Databases [Journal] // AI Magazine. - [s.l.] : Association for the Advancement of Artificial Intelligence (www.aaai.org), 1996. - 3 : Vol. 17.

**Flores Francesc Campoy, Quint Vincent and Vatton Irene** Templates, Microformats and Structured Editing [Conference] // Proceedings of the 2006 ACM symposium on Document engineering. - Amsterdam : ACM, NY, USA, 2006. - pp. 188 - 197. - doi: 10.1145/1166160.1166211.

**Frank Andrew U. and Kuhn Werner** Specifying Open GIS with Functional Languages [Conference] // Advances in Database (4th International Symposium) / ed. Egenhofer Max J. and Herring John R.. - Portland : Springer-Verlag, 1995. - pp. 184 - 195. - doi: 10.1007/3-540-60159-7..

**Frank Martin R. and Foley James D.** Model-based user interface design by example and by interview [Conference] // Proceedings of the 6th annual ACM symposium on User interface software and technology. - Atlanta, Georgia : ACM, 1993. - pp. 129 - 137. - doi: 10.1145/168642.168655.

**Frawley William J., Piatetsky-Shapiro Gregory and Matheus Christopher J.** Knowledge Discovery in Databases: An Overview [Journal]. - [s.l.] : Association for the Advancement of Artificial Intelligence (www.aaai.org), 1992. - 3 : Vol. 13.

**Furtado Elizabeth [et al.]** An Ontology-Based Method for Universal Design of User Interfaces [Conference] // Task Models and Diagrams For User Interface Design (TAMODIA 2002). :INFOREC. - Bucharest : CiteSeerX, 2002. - CiteSeerX doi=10.1.1.13.4086.

**Furtado Elizabeth [et al.]** An Ontology-Based Method for Universal Design of User Interfaces [Conference] // Task Models and Diagrams For User Interface Design (TAMODIA 2002). :INFOREC. - Bucharest : CiteSeerX, Bucharest. - CiteSeerX doi=10.1.1.13.4086.

**Gangemi Aldo [et al.]** Sweetening Ontologies with DOLCE [Conference] // Proceedings of the 13th International Conference on Knowledge Engineering and Knowledge Management. Ontologies and the Semantic Web. - Siguenza : Springer-Verlag, 2002. - pp. 166 -181.

**Gangemi Aldo** Ontology Design Patterns for Semantic Web Content [Conference] // Proceedings of the Fourth International Semantic Web Conference. - [s.l.] : Springer-Verlag Berlin Heidelberg, 2005. - pp. 262 - 276. - doi: 10.1007/11574620\_21.

**Garcia Elena and Sicilia Miguel-Angel** User Interface Tactics in Ontology-Based Information Seeking [Journal]. - Madrid : PsyNology Journal, 2003. - 3 : Vol. 1. - pp. 242-255.

**Garrett Jesse James** Element of User Experience: User-Centered Design for the Web [Book]. - [s.l.] : Peachpit Press, 2002. - ISBN: 978-0735712027.

**Georgiev Iliya** Ontology Modelling for Semantic Web-driven Application [Conference] // Proceedings of International Conference on Computer Systems and Technologies. - 2005. - pp. II.8-1 - II.8-6.

**Goodwin Nancy C** Functionality and Usability [Journal] // Communications of the ACM. - [s.l.] : ACM, 1987. - 3 : Vol. 30. - pp. 229 - 233. - doi: 10.1145/214748.214758.

**Gould John D. and Lewis Clayton** Designing for usability: key principles and what designers think [Journal] // Communications of the ACM. - New York, NY : ACM, 1985. - 3 : Vol. 8. - pp. 300 - 311. - doi: <http://doi.acm.org/10.1145/3166.3170>.

**Gould John D.** How to design usable systems [Book Section] // Human-Computer Interaction: toward the year 2000 / book auth. Baecker Ronald M. [et al.]. - San Francisco : Morgan Kaufmann Publishers Inc., 1995. - ISBN:1-55860-246-1.

**Green Thomas R. G. and Petre Marian** Usability Analysis of Visual Programming Environments: A 'Cognitive Dimensions' Framework [Journal] // Journal of Visual Languages and Computing. - 1996. - 2 : Vol. 7. - pp. 131-174.

**Grenon Pierre and Smith Barry** SNAP and SPAN: Towards Dynamic Spatial Ontology [Journal] // Spatial Cognition and Computation. - [s.l.] : Lawrence Erlbaum Associates, Inc, 2004. - 1 : Vol. 4. - pp. 69 - 103. - doi: 0.1207/s15427633scc0401\_5.

**Gruber Thomas R.** A Translation Approach to Portable Ontology Specifications [Journal] // Knowledge Acquisition - Special issue: Current issues in knowledge modeling. - Palo Alto : [s.n.], June 1993. - 2 : Vol. 5. - pp. 199 - 220. - doi: 10.1006/knac.1993.1008.

**Gruber Thomas R.** Toward Principles for the Design of Ontologies Used for Knowledge Sharing [Journal] // International Journal Human-Computer Studies. - [s.l.] : Elsevier Ltd., November 1995. - 5-6 : Vol. 43. - pp. 907-928. - doi:10.1006/ijhc.1995.1081.

**Guarino Nicola and Giaretta Pierdaniele** Ontologies and Knowledge Bases: Towards a Terminological Clarification [Book Section] // Towards Very Large Knowledge Bases / book auth. Mars N. J. I.. - Amsterdam : IOS Pr Inc., 1995. - ISBN: 978-9051992175.

**Guarino Nicola** Formal Ontology and Information Systems [Conference] // Formal Ontology in Information Systems.. - Trento : IOS Press, Amsterdam, 1998. - Vol. N. Guarino (ed.). - pp. 3 - 15. - citeulike:4148125.

**Guarino Nicola** Formal ontology, conceptual analysis and knowledge representation [Journal] // International Journal of Human-Computer Studies. - [s.l.] : Academic Press, 1995. - 5 - 6 : Vol. 43. - pp. 625 - 640. - doi:10.1006/ijhc.1995.1066.

**Guarino Nicola** Semantic Matching: Formal Ontological Distinctions for Information Organization, Extraction, and Integration [Conference] // Summer School on Information Extraction. - Frascati, : Springer, 1997.

**Hitzler Pascal, Krötzsch Markus and Rodolph Sebastian** Foundations of Semantic Web Technologies [Book]. - Dayton, Ohio : CRC Press T & F Group, 2009. - ISBN: 978-1-4200-9050-5.

**Horridge Matthew [et al.]** A Practical Guide To Building OWL Ontologies Using The Protégé-OWL Plugin and CO-ODE Tools [Report] / The University Of Manchester. - Manchester : The University Of Manchester, 2004. - pp. 0 - 117.

**Hudak Paul** Conception, evolution, and application of functional programming languages [Journal] // ACM Computing Surveys. - [s.l.] : ACM, September 1989. - 3 : Vol. 21. - doi: 10.1145/72551.72554.

**Jackson Michael** The design and use of conventional programming languages [Book Section] // Human Interaction With Computers / book auth. Smith H. T.. - [s.l.] : Academic Press, 1980. - Vol. H T Smith & T R Green eds. - ISBN: 978-0126528503.

**Johnson Jeff [et al.]** The Xerox Star: A Retrospective [Journal] // Computer. - [s.l.] : IEEE Computer Society , 1989. - 9 : Vol. 22. - pp. 11 - 26. - doi: 10.1109/2.35211 .

**Johnson Jeff** Selectors: Going Beyond User-Interface Widgets [Report] : Technical Report / Human-Computer Interaction Department ; Hewlett-Packard Laboratories. - Palo Alto, CA 94304 : ACM, 1992. - pp. 273-279. - ACM 0-89791 -513-5 /92/0005-0273.

**Judson David H.** Web browser with dynamic display of information objects during linking [Patent] : 5572643. - USA, November 5, 1996.

**Kalinichenko Leonid [et al.]** Ontological Modeling [Conference] // Proceedings of the 5th Russian Conference on Digital Libraries RCDL2003. - St.-Petersburg : [s.n.], 2003.

**Kieffer Suzanne, Coyette Adrien and Vanderdonckt Jean** User interface design by sketching: a complexity analysis of widget representations [Conference] // Proceedings of the 2nd ACM SIGCHI symposium on Engineering interactive computing systems. - Berlin : ACM, 2010. - pp. 57 - 66. - 10.1145/1822018.1822029.

**Klieber Werner [et al.]** Using Ontologies For Software Documentation [Conference] // Proc Malaysian Joint Conference on Artificial Intelligence MJCAI2009. - Kuala Lumpur : [s.n.], 2009.

**Kristiansen Renate and Trætteberg Hallvard** Model-based user interface design in the context of workflow models [Conference] // Proceedings of the 6th international conference on Task models and

diagrams for user interface design. - Toulouse : Springer-Verlag, 2007. - pp. 227 - 239. - ISBN: 978-3-540-77221-7.

**Kuhn Werner** An Image-Schematic Account of Spatial Categories [Conference] // Lecture Notes in Computer Science: Spatial Information Theory (COSIT'07). - Melbourne : Springer-Verlag, 2007. - pp. 152 - 168. - doi: 10.1007/978-3-540-74788-8\_10.

**Landay James A. and Myers Brad A.** Interactive sketching for the early stages of user interface design [Conference] // Proceedings of the SIGCHI conference on Human factors in computing systems. - Denver, Colorado : ACM Press/Addison-Wesley Publishing Co., 1995. - pp. 43 - 50. - doi: 10.1145/223904.223910.

**Latif Atif [et al.]** Turning keywords into URIs: simplified user interfaces for exploring linked data [Conference] // Proceedings of the 2nd International Conference on Interaction Sciences: Information Technology, Culture and Human. - Seoul : ACM, 2009. - pp. 76 - 81. - doi: 10.1145/1655925.1655939.

**Leijen Daan** wxHaskell: a portable and concise GUI library for haskell [Conference] // Proceedings of the 2004 ACM SIGPLAN workshop on Haskell. - Snowbird, Utah : ACM, 2004. - pp. 57 - 68. - doi: 10.1145/1017472.1017483.

**Lichter Horst, Schneider-Hufschmidt Matthias and Zullighoven Heinz** Prototyping in industrial software projects-bridging the gap between theory and practice [Journal] // IEEE Transactions on Software Engineering. - [s.l.] : IEEE Computer Society, November 1994. - 11 : Vol. 20. - pp. 825 - 832. - doi: 10.1109/32.368126 .

**Liu Ben, Chen Hejie and He Wei** Deriving user interface from ontologies: a model-based approach [Conference] // Proceedings of the 17th IEEE International Conference on Tools with Artificial Intelligence. - Hong Kong : IEEE Computer Society, 2005. - pp. 259 - 264. - doi: 10.1109/ICTAI.2005.55 .

**Loebe Frank and Herre Heinrich** Formal Semantics and Ontologies: Towards an Ontological Account of Formal Semantics [Conference] // Proceedings for International Conference on Formal Ontology in Information Systems (FOIS). - Saarbrücken : IOS Press, 2008. - pp. 49 - 62. - doi:10.3233/978-1-58603-923-3-49.

**Lyon Douglas A.** Semantic Annotation for Java [Journal] // Journal of Object Technology. - Zurich : [s.n.], 2010. - 3 : Vol. 9. - pp. 19 - 29.

**Martin Philippe** Knowledge representation in RDF/XML, KIF, Frame-CG and Formalized-English [Conference] // Conceptual Structures: Integration and Interfaces, 10th International Conference on Conceptual Structures, ICCS. - [s.l.] : Springer, 2002. - Vol. 2393. - pp. 77 - 91. - ISBN: 3-540-43901-3.

**Mayhew Deborah J.** Principles and guidelines in software user interface design [Book]. - Englewood Cliffs, N.J. : Prentice Hall, 1992.

**Mayhew Deborah J.** The usability engineering lifecycle [Conference] // CHI '99 extended abstracts on Human factors in computing systems. - Pittsburgh, Pennsylvania : ACM, 1999. - pp. 147 - 148. - doi: <http://doi.acm.org/10.1145/632716.632805>.

**Moggridge Bill** Designing Interactions [Book]. - New York : The MIT Press, 2007. - ISBN: 978-0262134743.

**Montero Susana, Díaz Paloma and Aedo Ignacio** Formalization of web design patterns using ontologies [Conference] // Proceedings of the 1st international Atlantic web intelligence conference on Advances in web intelligence. - Madrid : Springer-Verlag, 2003. - pp. 179 - 188. - ISBN: 3-540-40124-5.

**Myers Brad, Hudson Scott E. and Pausch Randy** Past, present, and future of user interface software tools [Journal] // ACM Transactions on Computer-Human Interaction (TOCHI) - Special issue on human-computer interaction in the new millennium. - [s.l.] : ACM, 2000. - 1 : Vol. 7. - pp. 3 - 28. - doi: [10.1145/344949.344959](http://doi.acm.org/10.1145/344949.344959).

**Najar Salma [et al.]** Semantic representation of context models: a framework for analyzing and understanding [Conference] // Proceedings of the 1st Workshop on Context, Information and Ontologies. - Heraklion : ACM New York, NY, USA ©2009, 2009. - pp. 6:1- 6:10. - doi: [10.1145/1552262.1552268](http://doi.acm.org/10.1145/1552262.1552268).

**Namgoong Hyun. [et al.]** An Adaptive User Interface in Smart Environment [Conference] // IEEE Tenth International Symposium on Consumer Electronics ISCE '06. - St. Petersburg : IEEE Explore, 2006. - pp. 1 - 6. - doi: [10.1109/ISCE.2006.1689450](http://doi.ieeecomputersociety.org/10.1109/ISCE.2006.1689450) .

**Navigli Roberto and Velardi Paola** Learning Domain Ontologies from Document Warehouses and Dedicated Web Sites [Journal] // Computational Linguistics. - [s.l.] : MIT Press, June 2004. - 2 : Vol. 30. - pp. 151 - 179. - doi: [10.1162/089120104323093276](http://doi.acm.org/10.1162/089120104323093276).

**Neto Renato F. Bulcão, Kudo Taciana Novo and Pimentel Maria da Graça** Using a software process for ontology-based context-aware computing: a case study [Conference] // Proceedings of the 12th Brazilian Symposium on Multimedia and the web. - Natal, Rio Grande do Norte : ACM, NY, USA, 2006. - pp. 61 - 70. - doi: [10.1145/1186595.1186604](http://doi.acm.org/10.1145/1186595.1186604).

**Newell Alan** The Knowledge Level [Journal] // Artificial Intelligence. - 1982. - 1 : Vol. 18. - pp. 87 - 127.

**Nielsen Jakob** User interface directions for the Web [Article] // Magazine Communications of the ACM. - New York : ACM, January 1999. - 1 : Vol. 42. - pp. 65 - 72. - doi: [10.1145/291469.291470](http://doi.acm.org/10.1145/291469.291470).

**Nirenburg Sergei and Raskin Victor** Ontological semantics, formal ontology, and ambiguity [Conference] // Proceedings of the international conference on Formal Ontology in Information Systems - Volume 2001. - Ogunquit, Maine : ACM, NY, USA, 2001. - pp. 151 - 161. - doi: [10.1145/505168.505183](http://doi.acm.org/10.1145/505168.505183).



**Norman Donald A.** The Design of Everyday Things [Book]. - New York : Basic Books, 1988. - ISBN: 9780465067107.

Ontology-based User Interface Development: User Experience Elements Pattern [Journal] // Journal of Universal Computer Science. - April 1, 2011. - 7 : Vol. 17. - pp. 1078 - 1088. - doi: 10.3217/jucs-017-07-1078.

**Osterwalder Alexander and Pigneur Yves** An e-Business Model Ontology for Modeling e-Business, e-Reality: Constructing the e-Economy [Conference] // 15th Bled Electronic Commerce Conference. - Bled : EconWPA, 2002.

**Parkin Simon E., Moorsel Aad van and Coles Robert** An information security ontology incorporating human-behavioural implications [Conference] // Proceedings of the 2nd international conference on Security of information and networks. - Famagusta : ACM, NY, USA, 2009. - pp. 46 - 55. - doi: 10.1145/1626195.1626209.

**Patil Lalit, Dutta Debasish and Sriram Ram** Ontology-Based Exchange of Product Data Semantics [Journal] // IEEE Transactions on Automation Science and Engineering. - July 2005. - 3 : Vol. 2. - pp. 213 - 225. - doi: 10.1109/TASE.2005.849087.

**Paulheim Heiko** Ontologies for User Interface Integration [Conference] // Proceedings of International Semantic Web Conference. - Chantilly, VA : Springer, 2009. - pp. 973 - 981. - doi: 10.1007/978-3-642-04930-9\_63.

**Petrasch Roland** Model based user interface development with HCI patterns: variatio delectat [Conference] // Proceedings of the 1st International Workshop on Pattern-Driven Engineering of Interactive Computing Systems. - Berlin : ACM, 2010. - pp. 10 - 11. - doi: 10.1145/1824749.1824752.

**Pirlein Thomas and Studer Rudi** An environment for reusing ontologies within a knowledge engineering approach [Journal] // International Journal of Human-Computer Studies. - [s.l.] : Academic Press, November 1995. - 5 - 6 : Vol. 43. - doi:10.1006/ijhc.1995.1083.

**Poli Roberto** Descriptive, Formal and Formalized Ontologies [Book Section] // Husserl's Logical Investigations Reconsidered / book auth. Fisette D.. - [s.l.] : Kluwer Academic Publishers, 2003. - Vol. 48.

**Puerta Angel R.** Issues in Automatic Generation of User Interfaces in Model-Based Systems [Conference] // Proceedings of the Second International Workshop on Computer-Aided Design of User Interface. - Belgium : Universitaires de Namur, Namur, 1996. - pp. 323 - 326. - ISBN: 2-87037-232-9.

**Raymond Eric Steven and Landley Rob W.** The Art of Unix Usability [Online] / prod. Raymond Eric S.. - April 18, 2004. - 0.1. - <http://catb.org/~esr/writings/taouu/html/index.html>.

**Reeda P. [et al.]** User interface guidelines and standards: progress, issues, and prospects [Journal] // Interacting with Computers. - [s.l.]: Elsevier, 1999. - 2: Vol. 12. - pp. 119 – 142. - doi: [http://dx.doi.org/10.1016/S0953-5438\(99\)00008-9](http://dx.doi.org/10.1016/S0953-5438(99)00008-9).

**Rubin Jeffrey** Handbook of Usability Testing: How to Plan, Design, and Conduct Effective Tests [Book]. - New York : John Wiley & Sons, Inc., 1994. - ISBN: 978-0471594031.

**Schlungbaum Egbert** Model-based User Interface Software Tools Current state of declarative models [Report] : Technical Report / Graphics , Visulisation and Usability Center ; Georgia Institute of Technology. - Atlanta, Georgia : Georgia Institute of Technology, 1996.

**Shackel Brian** Usability— context, framework, definition, design and evaluation [Book Section] // Human factors for informatics usability. - [s.l.] : Cambridge University Press, 1991. - ISBN: 0-521-36570-8.

**Shahzad Syed Khuram and Granitzer Michael** Ontological Framework Driven GUI Development [Conference] // Proceedings of 10th International Conference on Knowledge Management and Knowledge Technologies. - Graz : [s.n.], 2010. - pp. 198 - 206.

**Shahzad Syed Khuram, Granitzer Michael and Helic Denis** Ontological Model Driven GUI Development: User Interface Ontology Approach [Conference] // 6th International Conference on Computer Sciences and Convergence Information Technology (ICCIT). - Seogwipo : [s.n.], 2011.

**Shahzad Syed Khuram, Granitzer Michael and Klaus Tochtermann** Designing User Interfaces through Ontological User Model: Functional Programming Approach [Conference] // Proceedings of the Fourth International Conference on Computer Sciences and Convergence Information Technology ICCIT 2009. - Seoul : IEEE Explore, 2009. - pp. 99 - 104. - doi:10.1109/ICCIT.2009.330.

**Shankar Natarajan** Automated Deduction for Verification [Journal] // ACM Computing Surveys. - [s.l.]: ACM New York, NY, USA , October 2009. - Vol. 41, No. 4. - pp. 20:1-20:56. - <http://doi.acm.org/10.1145/1592434.1592437>.

**Shanks Graeme, Tansley Elizabeth and Weber Ron** Using ontology to validate conceptual models [Article] // Communications of the ACM - Service-oriented computing CACM. - [s.l.]: ACM, NY, USA., October 2003. - 10 : Vol. 46. - pp. 85 - 89. - doi: 10.1145/944217.944244.

**Smith Barry** An Essay in Formal Ontology [Journal] // Grazer Philosophische Studien. - Graz : [s.n.], 1978. - pp. 39–62.

**Smith Barry and Welty Christopher** Ontology: Towards a new synthesis [Conference] // Proceedings of the international conference on Formal Ontology in Information Systems (FOIS2001). - Ogunquit, Maine : ACM Press, 2001. - pp. 3-9. - doi: 10.1145/505168.505201.

**Smith Barry** Beyond Concepts: Ontology as Reality Representation [Conference] // International Conference on Formal Ontology and Information Systems. - Turin : IOS Press, 2004. - pp. 73 - 84. - doi: 10.1.1.58.5118.

**Smith Barry** Formal ontology, common sense and cognitive science [Journal] // International Journal of Human-Computer Studies. - [s.l.] : Academic Press, 1995. - 5 - 6 : Vol. 43. - pp. 641 - 667. - doi:10.1006/ijhc.1995.1067.

**Smith Barry** Logic and Formal Ontology [Book Section] // Husserl's Phenomenology: A Textbook (Current Continental Research) / book auth. Mohanty J. N. / ed. McKenna William R.. - Lanham : University Press of America, 1989. - ISBN-13: 978-0819175311.

**Smith Barry** Mereotopology: A theory of parts and boundaries [Journal] // Data & Knowledge Engineering. - [s.l.] : Elsevier Science, 1996. - 3 : Vol. 20. - pp. 287-303. - doi:10.1016/S0169-023X(96)00015-8.

**Smith Barry** Ontology (Science) [Online] // Nature Proceedings. - Nature Proceedings, July 15, 2008. - <http://hdl.handle.net/10101/npre.2008.2027.2>. - doi:10.3233/978-1-58603-923-3-21.

**Smith Sidney L. and Mosier Jane N.** Guidelines for designing User Interface Software [Report] = ESD-TR-86-278 : Technical / The MITRE Corporation Bedford ; United States Air Force, Hanscom Air Force Base. - Massachusetts : Userlab Inc., 1998.

**Sommerville Ian [et al.]** Cooperative Systems Design [Journal] // The Computer Journal. - [s.l.] : Oxford University Press, 1994. - 5 : Vol. 37. - doi: 10.1093/comjnl/37.5.357.

**Spyns Peter, Meersman Robert and Jarrar Mustafa** Data modelling versus Ontology engineering [Journal] // Newsletter ACM SIGMOD Record. - [s.l.] : ACM New York, NY, USA, December 2002. - 4 : Vol. 31. - pp. 12-17. - doi: 10.1145/637411.637413.

**Stamper Ronald and Liu Kecheng** Organisational dynamics, social norms and information systems [Conference] // Proceedings of Twenty-Seventh Hawaii International Conference on System Sciences. - Hawaii : IEEE, 1994. - pp. 645 - 654.

**Sukaviriya Piyawadee Noi and Foley James D.** Supporting adaptive interfaces in a knowledge-based user interface environment [Conference] // 1st international conference on Intelligent user interfaces. - Orlando, Florida, United States : ACM New York, NY, USA ©1993, 1993. - doi: 10.1145/169891.169922.

**Sun Microsystems, Inc.** Java Look and Feel Design Guidelines [Online] // Sun Developers Network. - Sun Microsystems, Inc., February 2011. - <http://java.sun.com/products/jlf/ed2/book/index.html>.

**Szekely Pedro** Retrospective and Challenges for Model-Based Interface [Conference] // Proceedings of the Third International Eurographics Workshop. - Namur : Springer-Verlag, 1996. - pp. 1 - 27. - ISBN : 3-211-82900-8.

**Thomas Cathy and Bevan Nigel** Usability context analysis: a practical guide [Report] / National Physical Laboratory. - Teddington, Middlesex : HMSO National Physical Laboratory, 1996.

**Uschold Michael** Ontology-Driven Information Systems: Past, Present and Future [Conference] // Proceedings of Fifth International Conference of Formal Ontology in Information Systems. - Karlsruhe : IOS Press, 2008. - pp. 3 - 18.

**Van Dam Andries** Post-WIMP user interfaces [Article] // Communications of the ACM. - [s.l.] : ACM, 1997. - 2 : Vol. 40. - pp. 63 - 67. - doi: 10.1145/253671.253708.

**Van den Bergh Jan and Coninx Karin** notations, Model-based design of context-sensitive interactive applications: a discussion of [Conference] // Proceedings of the 3rd annual conference on Task models and diagrams. - Prague : ACM, 2004. - pp. 43 - 50. - doi: 10.1145/1045446.1045456.

**Vanderhulst Geert, Luyten Kris and Coninx Karin** Put the User in Control: Ontology-driven Meta-level Interaction for Pervasive Environments [Conference] // First International Workshop on Ontologies in Interactive Systems, 2008. ONTORACT '08.. - Liverpool : IEEE Computer Society, 2008. - pp. 51 - 56. - doi: 10.1109/ONTORACT.2008.15 .

**Varzi Achille C.** Basic Problems of Mereotopology [Conference] // Published in N. Guarino (ed.), Formal Ontology in Information Systems, . - Amsterdam : IOS Press, 1998. - pp. 29 - 38. - doi: 10.1.1.6.9025.

**Vet Paul E. van der and Mars Nicolaas J.I.** Bottom-up construction of ontologies [Journal] // IEEE TRANSACTIONS ON KNOWLEDGE AND DATA ENGINEERING. - Enschede : IEEE Computer Society, July/August 1998. - 4 : Vol. 10. - doi: 10.1109/69.706054.

**Viswanathan Satya and Peters Johan Christiaan** development, Automating UI guidelines verification by leveraging pattern based UI and model based [Conference] // Proceedings of the 28th of the international conference extended abstracts on Human factors in computing systems. - Atlanta, Georgia : ACM, 2010. - pp. 4733 - 4742. - doi: 10.1145/1753846.1754222.

**Wache H. [et al.]** Ontology-Based Integration of Information — A Survey of Existing Approaches [Conference] // Proceedings of IJCAI-01 Workshop: Ontologies and Information Sharing.. - Seattle, WA : CEUR-Workshop Proceedings, 2001. - pp. 108-117.

**Walrath Kathy [et al.]** The JFC Swing Tutorial: A Guide to Constructing GUIs, Second Edition [Book]. - Redwood City, CA : Addison Wesley Longman Publishing Co., Inc., 2004. - ISBN: 0201914670.

**Wand Y. and Weber R.** An ontological model of an information system [Journal] // IEEE Transactions on Software Engineering. - [s.l.] : IEEE Computer Society, 1990. - 11 : Vol. 16. - pp. 1282 - 1292. - doi: 10.1109/32.60316.

**Woo Mason [et al.]** OpenGL Programming Guide: The Official Guide to Learning OpenGL, Version 1.2 [Book]. - Boston, MA : Addison-Wesley Longman Publishing Co., Inc., 1999. - ISBN: 0201604582.

**Woods David D. and Johannesen, Leila and Potter, Scott S.** Human interaction with intelligent systems: an overview and bibliography [Journal] // ACM SIGART Bull. - [s.l.] : ACM, 1991. - 5 : Vol. 2. - pp. 39 - 50. - doi: 10.1145/122570.122571.

**Zaihrayeu Ilya [et al.]** From web directories to ontologies: natural language processing challenges [Conference] // Proceedings of the 6th international The semantic web and 2nd Asian conference on Asian semantic web conference. - Busan : Springer-Verlag, 2007. - pp. 623 - 636. - ISBN: 978-3-540-76297-3.

**Zimmermann Antoine [et al.]** Formalizing Ontology Alignment and its Operations with Category Theory [Conference] // Proceedings of the 2006 conference on Formal Ontology in Information Systems: Proceedings of the Fourth International Conference (FOIS 2006). - Amsterdam : ACM, NY, USA, 2006. - pp. 277 - 288. - ISBN: 1-58603-685-8.