INVESTIGATIONS IN ONTOLOGY BASED SOFTWARE ENGINEEING

Thesis Submitted for the Degree of

DOCTOR OF PHILOSOPHY

(Computer Science)

By

Shilpa Sharma

To

Devi Ahilya Vishwavidyalaya, Indore

Under

Faculty of Engineering Science

2012

Supervised By

Dr. (Mrs.) Maya Ingle

Professor and Senior System Analyst

School of Computer Science and Information Technology

Devi Ahilya Vishwavidyalaya, Khandwa Road, Indore (MP) – 452001

DECLARATION BY THE CANDIDATE

I declare that the thesis entitled **Investigations in Ontology Based Engineering** is my own work conducted under the supervision of **Dr. (Mrs.) Maya Ingle** at the School of Computer Science and Information Technology, Devi Ahilya Vishwavidyalaya, Indore approved by the Research Degree Committee. I have put in more than 200 days attendance with the supervisor at the center.

I further declare that to the best of my knowledge, the thesis does not contain any part of any work which has been submitted for the award of any degree either in this University or in other University/ Deemed University without prior citation.

[illegible]

Signature of Supervisor
Dr. (Mrs.) Maya Ingle

[illegible]

Signature of Candidate
Shilpa Sharma

[illegible]

Signature of Head

Head
School of Computer Science & IT
D.A.V.V. INDORE

CERTIFICATE OF THE SUPERVISOR

CERTIFICATE

This is to certify that the work entitled **Investigations in Ontology Based Engineering** is a piece of research work done by **Mrs. Shilpa Sharma** under my guidance and supervision for the degree of **Doctor of Philosophy (Ph.D.)** of Devi Ahilya Vishwavidyalaya, Indore (M.P.), India. That the Candidate has put in an attendance of more than 200 days with me.

To the best of my knowledge and belief the thesis:

(i) Embodies the work of the candidate herself;

(ii) Has duly been completed;

(iii) Fulfills the requirements of the Ordinance to the Ph.D. degree of the University; and

(iv) Up to the standard both in respect of contents and the language for being referred to the examiner.

Forwarded

[illegible]

[illegible]

Signature of Supervisor
Dr. (Mrs.) Maya Ingle

Signature of Head

Head
School of C[illegible] Science & [illegible]
D[illegible]RE

DISSERTATION APPROVAL SHEET

Thesis entitled **Investigations in Ontology based Engineering** by **Mrs. Shilpa Sharma** is approved for the award of the degree of **Doctor of Philosophy (Ph.D.)** under the Faculty of Engineering Science **(Computer Science)**.

Dr. (Mrs.) Maya Ingle	**Examiner(s)**
Supervisor	
Date:	Date:

Acknowledgement

I am on cloud nine today as, I am writing the most important part of this most important assignment of my life so far. The day I decided to go for research, I was very fickle minded because I knew about the tough roads of this journey but had no clue how to walk on. All it was as clear as mud for me. Finally, my iron-will encouraged me to face the challenges and to do something productive. But, what was that element behind my iron-will, what that ineffable energy which kept me charge was and what was that invincible power which made me complete this piece of work without failing. Well... Words are insufficient to express my emotions towards that element, energy, invincible power or that omnipresent and omnipotent almighty... I don't know how to thank GOD because it's only he who has completed this work and has chosen me as a medium. I thank GOD for creating a shield of grace around me and blessed me that I could work on this useful and meaningful subject. Thank You GOD...

As it is correctly said that GOD cannot be present everywhere so he created mother... I wish to add something more to it. GOD cannot come to us every time so he created mother and the mentor. I convey my sincere gratitude towards my supervisor **Dr. (Mrs.) Maya Ingle**, Professor and Senior System Analyst, School of Computer Science and Information Technology, DAVV who not only is my supervisor for this research work but also a strong emotional support for me throughout. She has handled me with care and made me wake up and smell the coffee while required. She is an institution of knowledge, full of enthusiasm for research and development and amazingly passionate for work. She is and will be a great source of endless motivation and inspiration for me forever... I realise this six letter word "Thanks", is like a drop in the ocean in view of her unmatchable determination of supervising my work round the clock. Despite of her extremely busy days and evenings, she never refused me to speak or meet to clear my doubts and guided me while I was chasing my tail. I am fortunate enough to have her as a supervisor indeed. While, I was working with

her eventually I came across of her Utopia of this domain. I could not resist myself to notice that she always dot all the i's and cross all the t's and expect same from us for our betterment only. Dear Madam... I am obliged for your holding arms during about to fall situations and on the tough surface of this road map.

I owe my thanks to **Late Dr. A. K. Ramani.** It was his gracious presence and overwhelming aura while I was registered for Ph. D. I am obliged for each and every support provided by him. The overall learning from him will be highly lucid and useful in my whole life.

I extend my sincere thanks for **Dr. S. Tanwani,** Professor and Head, School of Computer Science and Information Technology, DAVV, for his absolutely favourable gesture towards this research work. By providing excellent resources as most effective weapons, he made this challenge easier for me indeed.

I gratefully acknowledge **Sanghvi Institute of Management and Science,** for the support during the initial and most crucial phase of this journey. I see eye to eye that, every long journey starts with a single small step or a good beginning is the key to successful end. Let me admit that if I would not have got the positive and co-operative environment here at SIMS, I could not have reached to this state to write this work. Thank you SIMS... for providing me such an incredible support, all desired facilities and essential elbow room to inspire me. I sincerely thankful to **Medicaps Institute of Technology and Management** for creating favourable and comfortable platform for me to keep going towards my greatest achievement so far. I am extremely thankful to **Dr. Shamsher Singh**, Chief Executive Director, MITM, for his firm interest in research and science which enhanced confidence in me.

I am indebted to **Dr. (Mrs.) Kshama Paithankar,** Head, Shri Vaishnav Institute of Management, and **Dr. (Mrs.) Shilpa Bhalerao,** Head, Acropolis Institute of Technology and Research, Indore. Their knowledge and experience played a vital role in this research work. Technical discussions with them have facilitated my yen to be fulfilled. In the absence of their encouraging association, my level of enthusiasm would have gone down... way back.

And then I thank my colleague at MITM, **Ms. Abhilasha Prasad,** whose enriching advices are unforgettable and praiseworthy. I remember how regularly I used to bother her and she used to be bright and breezy all the time. I am thankful to my ex-colleague at SIMS, **Mrs. Preeti Gupta,** who, by providing her togetherness from the very beginning has really obliged me.

I am grateful to my **In Laws** for the support and encouragement they have provided me to complete my study in spite that I could not spare time for them. I don't have words to thank my adorable parents, without their kind and cordial blessings my dream could not be a came true story. Thanking you **Aai, Baba** for your unseen but most effective and productive best wishes. I am indebted to your ever supportive encouraging and empowering shower of words.

I appreciate my younger brother **Ajinkya Kulkarni** for his commendable endeavour. Despite of his own crucial time of studies, he never denied to assist me any ways. Whenever I needed him, he always before me with a boosting smile. Also, I thank my sister **Rupali Kulkarni** whose distant but worthy adhesion played a role of a booster this entire period of research work.

And, a big thanks to my dear husband **Mr. Vishwas Sharma,** whose sprightliness, exceptional patience, emotional support and upholding alone all societal relations empowered my efforts. His inspirational assertions worked as a catalyst to elicit my allegiance and potential kept me in positivism. However, there were few moments in past three years which turned me to panic but his optimistic words used to keep relaxing me.

I am delighted to thank all my friends and well wishers whose name I am missing to mention here and who directly or indirectly walked along with me throughout. Their best wishes were the most effective tool during this period. Thank you all...

Shilpa Sharma

Table of Contents

List of Figures

List of Tables

List of Appendices

CHAPTER 1 Introduction

1.1 Preamble

Research on ontology is becoming a widespread in software engineering community. Ontology is actually well known in philosophy research area since 1960s whereas in the artificial intelligence (AI) arena ontology has been focused on knowledge modelling [FGJ97, GOM01]. The term ontology is used to refer to explicit specification of a conceptualization of a domain [CJB99, DEV01]. In other words, ontology refers to a formalization of the knowledge in the domain [LINK9]. Ontology is the concept which is separately identified by domain users, and used in a self-contained way to communicate information. Combination of concept is the knowledge base or knowledge network [FMR98, JMY99, MA05]. These are some of the reasons that lead to develop ontologies for various software engineering issues. These issues include sharing common understanding of the structure of information among people or software agents. In addition, ontology can be used to enable reuse of domain knowledge and to make domain assumptions explicit for separating domain knowledge from the operational knowledge etc. [FGD92, GFC04, GH03].

In addition, ontologies have been viewed from distinct vantage points such as generality level, conceptualization structure type and nature of real world. According to generality level, ontologies are classified into high level ontologies, domain ontologies, task ontologies and application ontologies [GUA98]. In accordance with the type of conceptualization structure, ontology categorizes into terminological ontologies, information ontologies and knowledge representation ontologies [HSW+97]. As said by nature of real world concern, ontology have identified as static ontologies, dynamic ontologies, intentional ontologies and social ontologies [JMY99]. Besides, this linear way of classifying ontologies based on only sole criterion does not allow for adequate reflection of the problem's complexity. Consequently, bi dimensional classifications, taking into account two criteria such as the richness of the internal structure and the subject of the conceptualization. In this bi-dimensional proposal ontology belongs to any one of the categories such as controlled vocabularies, glossaries, thesauruses, informal and formal hierarchies, frames and ontologies with value and logical constraints [GOM01].

These numerous and varying ways of ontologies have been elucidated in order to serve knowledge based software engineering. In this manner, an ontology may take a variety of forms, but necessarily it will include a vocabulary of terms, and some specification. This includes definitions and indication of interrelated concepts which collectively impose a structure on the domain and constrain the possible interpretation of terms [UJ99].

Since ontology stimulation, substantial progress has been made in developing conceptual bases to build skill that allows reusing and sharing knowledge [GHW02, GS02]. As stated earlier, ontology has been created to share and reuse knowledge and reasoning behavior across domains and tasks. In this evolution, the most important fact has been the emergence of Ontology Based Software Engineering. It is an extension to current software engineering practices, in which the information is given a well defined meaning, better enabling resources and people to work in mutual aid. This mutual aid can be achieved by using shared knowledge thereby ontology has become key instrument in developing knowledge based software systems [GOM98, NFF+91, SSS+01].

In addition, ontology enfolds the attributes such as completeness, unambiguous, intuitive, generic and extensible. Completeness can be achieved by glancing at the different activities performed within software development. Ambiguity can be avoided by providing simple and concise definitions for each concept, as well as semi formal model of the complete ontology. Intuitiveness can be obtained by exploring the different communities participate in software development activities and by providing conceptual subset particularly adapted for each of them. Generality can be attained by upholding the ontology as small and as simple as possible and by trying to remove from it rather than add to it, as many concept as possible. The aim is to accomplish maximum expressiveness by being minimal. Finally, extensibility is realized by providing the appropriate mechanisms and anchors the points from which to add new concepts [BAR06, MF03, OVR+06].

Thus, the major contribution of ontology can be acquired in establishment of the software development methodologies such as generic software engineering, requirements engineering, reuse engineering and reliability engineering. Next, it provides best guidance to attain a life-cycle model best suited to the planned software development. Subsequently, ontology aids identification of main inputs, outputs and activities to be performed in order to develop the knowledge oriented approach. Knowledge sharing effort envisioned building knowledge-based software systems. Subsequently, the system developers need to create specialized knowledge and reasoners new to the specific task. This new system interoperates with existing systems, using to perform some of its reasoning. In this way, declarative knowledge, problem solving techniques and reasoning services would all be shared among systems. The knowledge and problem solving methods are modeled by means of ontology [DW99, RL02]

1.2 State of Art

Traditionally, software engineering termed as modelling activity. Software engineers deal with complexity through modelling, by focusing at any one time on only the relevant details and ignoring everything else [AW06, BOS95]. Later on, software engineering becomes problem-solving activity. Object-oriented methods combine the problem and solution domain modelling activities into one. The problem domain is first modelled as a set of objects and relationships. This model is then used by the system to represent the real-world concepts it manipulates. Thus, object modelling in software engineering influence all effort in information science [JAC92, WER+97]. Object models are different from other modelling techniques because these have merged the concept of variables and abstract data types into an abstract variable type known as object. Objects have identity, state, and behavior and object models are built out of systems of these objects. To make object modelling easier, there are concepts of type such as inheritance, association, and class. Object modelling focused on identity and behavior and is completely different from the traditional model's focus on information [BKK+02].

But, it is observed that models are used to search for an acceptable solution. This search is driven by experimentation and software engineers do not have infinite resources and are constrained by budget and deadlines. Given the lack of a fundamental theory, engineers often have to rely on empirical methods to evaluate the benefits of different alternatives Thereby software engineering turns into knowledge acquisition activity [GOM+98, MA05]. In modelling the application and solution domain, software engineers collect data, organize it into information, and formalize it into knowledge. Knowledge acquisition is nonlinear, as a single piece of data can invalidate complete models.

For the same reason, software engineering explores as a rationale-driven activity. When acquiring knowledge and making decisions about the system or its application domain, software engineers also need to capture the context in which decisions were made. This additional knowledge is called the rationale of the system. First, for every decision made, several alternatives may have been considered, evaluated, and argued. Consequently, rationale represents a much larger amount of information than the solution models do. Second, rationale information is often not explicit. Developers make many decisions based on their experience and their intuition, without explicitly evaluating different alternatives. In order to deal with changing systems, however, software engineers must address the challenges of capturing and accessing rationale [RL02, TA00].

In this perspective, we use ontology an optimum solution. Ontologies are a promising instrument for knowledge transfer from project to project in a certain application domain and from one development cycle of a project to the next project development cycle [SMJ02, OVR+06, VSS+05]. In the mid and long-term future, it might become an attractive software engineering paradigm which serves for closer co-operation, better compatible models, and more re-usable components in the software development field. Ontology here refers to the basic existential pool of knowledge in the world that is of interest to the discipline. In this view, the explicit treatment of knowledge and emphasizing on the category of requirement in requirement engineering practices suggests a fundamental shift in the domain oriented underpinnings of requirement engineering process. In addition, one of the main advantages of the ontology is its comprehensibility. The ontology helps to achieve some lucidness of unclear concepts related with software reuse reliability and security. Besides, the concepts were linked rigorously. Thus, there is tremendous scope in these sub domains of Ontology Based Software Engineering for researchers and practitioners.

1.3 Objectives

We have performed our research after reviewing the potential of ontology and designed the objectives on the basis of major challenges in front of current communities of software developers and practitioners. These challenges include knowledge integration and generic involuntary support. We have designed some ontology oriented models, framework and methods to chase these challenges. These models and methods are assessed on the basis of case studies and information received from case studies has been analyzed. Proposed works along with their case studies and results have been published in various national, international conferences and journals.

Our first objective is affirmed to build ontology for various information systems that enable the developers to reuse and share application domain knowledge using a common vocabulary across heterogeneous software applications [SI10a]. Ontologies involve the specification of concepts and relations that exist in the domain, definitions, properties and constraints mapped with each phase of Object Oriented Software Engineering. Thus, the phases such as *ontolysis, ontodesign* and *ontocontation* are turned out to generate Ontology Driven Information System (*ODIS*) [SI10b, SI11a].

Our next objective is avowed with the advent of knowledge intensive practices in requirement engineering. Consequently, ontology has become a definitive choice. It not only facilitates the confining of knowledge strenuous environment for requirement engineering but also enrich sharing of knowledge across various applications from different domains. Also, the ontology assists in defining information for the exchange of semantic software requirement specification data. Ontology aided requirement engineering endorses the categories of requirement to elicit, represent and analyze the diversity of factors associated with requirement engineering carried out using different requirement engineering process models. It forms various layers such as *OntoPre Requirements, OntoInput Requirements, OntoSystem Requirements and OntoOutput Requirements* depending upon requirement type and promotes the cohesiveness between the artifacts generated at every requirements engineering activity of different applications [SI11b, SI11c].

The subsequent objective is asserted to apprehend the software reuse through combining the conceptions of domain with stronger extensibility and with indexing knowledge population [SI12a]. The ingenious approach for software development with ontology validated composition in highly variable domains. The approach makes use of business domain ontologies and ontology of the domain of information system engineering. Furthermore, it relates several dimensions of software development in the course of various abstraction levels such as *Pretence, Persuade, Problem and Product.* It initiates with the *pretence view* by identification of knowledge sources useful for the application domains. Subsequently, an automatic translation of the source ontologies from a common format to the representation languages is carried out at the *problem view.* In addition, matching of the ensuing method is accepted at the *Persuade view.* Finally, the application ontologies revealed the reuse source vocabularies to a large extent in the *Product View* [SI11d]. Next, we have proposed an ontology based reuse algorithm *OntoReuseAlgo* towards process planning in software development. *OntoReuseAlgo* attempted to obtain a new process plan under new implementation requirements by modification of certain concepts and entities of the current process. *Ontological knowledge modelling* has been used to give a uniform representation of the involved information [SI11e].

Later objective is resolute to the reliability advancement using ontologies. *Ontology-Oriented Reliability* (*OnO-Reliability*) development has been proposed to enhance Object Oriented Reliability (OO-Reliability) development with the help of resources, process and product attributes. In order to achieve this goal, we have introduced the *Onto-self-ensuring recognition ordeal, Onto-multiple requests/confirmations* and *Onto-immunity management routines* [SI12c]. Making use of these, *the OnO-Reliability* development enables software architects and reliability experts to formally, explicitly, and coherently conduct reliability modelling. Besides, to improve the software reliability an OntoReliability Protocol has been proposed for developing software specifications called *OntoRelSpecifications.* It commenced reliability with abet of *description, preconditions, post conditions, standard courses, proxy courses, exceptions, inclusions, primacy, rate of uses, exceptional requirements and remarks and concerns* [SI12b]. Finally, to quantify software reliability we have proposed an *Ontological Reliability Quantification Method (ORQM)* with identification of *Project Category (PC)* based on architecture style and highly helpful to the developers to deal with software excellence.

Security has become an important quality for an ontology driven software system. While developing an ontology oriented project, various side effects occur due to the unvisualized states mainly; uncertainty, variability, ambiguity and complexity. We have suggested an Abstraction Method (*AM)* for developing the secured environment for ontology based projects developed with various perspectives such as generality, requirement, reuse and reliability engineering. It has been noticed that the influence of kinds of benefits associated with each perspective leads to aforementioned unvisualized state. Various security attributes corresponding to these perspectives are allocated to ensnare kinds of unvisualized states accordingly.

1.4 Outline of Thesis

This thesis is organized in eight Chapters to cover the research issues in the area of Ontology Based Software Engineering. These aspects mainly include enhancing generality, requirement engineering, reusability, reliability and security. A general overview of the said research field is provided along with its various aspects. A related state of art is presented subsequently. Also, the objectives of the proposed work have been mentioned in this Chapter.

We provide literature survey along with our view to Ontology Based Software Engineering research domain in Chapter 2. We present ontological engineering in the context of other disciplines and observe that it enables to enhance various issues of software engineering area. In this view, goals of software practitioners are covered to create new ways to build and improve knowledge in software engineering issues.

Chapter 3 illustrates development of ontology for various information systems to ensure the generality. In this view, we have discussed development phases of OOSDLC and ODLC in details. Various information systems along with their object-oriented development phases are covered in this Chapter. In next section, we have highlighted the OOSDLC phases and with various phases of ODLC in Information Systems. This mapping divulged *ODIS*. Base upon the mapping, we have introduced generalized models for corresponding to each phase. Lastly, we have shown the results of mapping of phases of OODSDLC with the phases of ODLC.

Chapter 4 depicts the comparison of various conventional Requirement Engineering Process models (REP) with Ontology Aided Requirement Engineering model (*OntoAidedRE*). For the same reason, we have discussed the parameters of study related to project such as Project type, Project size, Project team, Project quality, Project prioritized element and Project key element. These play a very significant role in RE for various types of projects. Next, conventional REP models with advantages and limitations in term of practices have been highlighted in next Section. Also, we have presented Ontology Aided

Requirement Engineering model (*OntoAidedRE*) covering requirement type, practices and suitability. Consequently, we have compared conventional REP models namely; Linear, Linear Iterative, Iterative and Spiral models with *OntoAidedRE.* The study reveals that none of conventional REP models acquire all project parameters. Therefore, we presented *OntoAidedRE* to show a knowledge-driven as opposed to process-driven approach to RE. It can be put into practice to overcome the problems of conventional REP models and consequently the project parameters optimally contrived by adapting *OntoAidedRE.*

In Chapter 5, first, we have discussed existing reuse subclasses followed by introduction of Object Oriented and Ontological Reuse process. Then, we have presented a reusable framework *OntoP4ViewReuse* based on ontology oriented systematic *P4View* approach for reusing. The necessity of *P4View* approach is to make available ontological knowledge that is implicitly tailored to specific application needs. *OntoP4ViewReuse* bring about to apply the ontology of varying levels such as high level, domain, task and application ontology. Consequently, we have explored a range of benefits of using *OntoP4ViewReuse.* In addition, to build a common conceptual base characterized by knowledge, Ontology Based Reuse Algorithm *(OntoReuseAlgo)* for process planning has been proposed. Also, the significant benefits of *OntoReuseAlgo* have been drawn. In addition, Ontological Reuse (OnR) has been devised from Object-Oriented Reuse (OOR) and effectiveness of OnR has been highlighted with comparative study based on software component, architecture, requirement, process, technology and experience reuse subclasses. Lastly, benefits of OnR have been delineated.

Software reliability achievement is a challenging task due to its dependency on users' perspective. In Chapter 6, we have introduced ontological approach for reliability achievement over object-oriented approach followed by a comparative analysis to outline the scope of Ontology Oriented Reliability (*OnO-Reliability*). In addition, ontological specifications have been developed using *OntoReliability* protocol and presented some case studies to practice this protocol. Subsequently, the benefits have been discussed. In the last Section, we have attempted to quantify the reliability of various project categories using project parameters and hence we have introduced Ontological Reliability Quantification Method (*ORQM*). Then, we have conducted a study of different project case as per the category with varying number and type of parameters and establish the fact that *ORQM* generates direct empirical value for software reliability. Finally, we conclude that *ORQM* is not a informal method but found to be a highly useful in absence of reliability experts and historical failure data.

Chapter 7 turns to one of the most important concept introduced in Ontology Based Projects (*OBPs*) i. e. software security. It deals with *OBPs* developed using various perspectives such as generality, requirement engineering, reusability and reliability. It has been noticed that the influence of kinds of benefits associated with each perspective leads to different unvisualized state. We have proposed secured software development environment for *OBPs* with various perspectives with the help of Abstraction Method (*AM*). This method aids different security attributes corresponding to these perspectives have been allocated to ensnare the kinds of unvisualized state accordingly. Finally, *AM* provides analytical scheme to acquire secured environment for different *OBPs*.

In Chapter 8, we have concluded with the contribution of our work presented in the area of Ontology Based Software Engineering in this thesis.

CHAPTER 2

Literature Review

2.1 Introduction

Ontological engineering has garnered increasing attention over the last few years, as researchers have recognized ontologies. Ontologies are not just for natural language processing, metaphysics, common sense knowledge and enterprise modelling etc. [DEV02, FCM+03]. However, it caters software engineers in modelling various applications of the world and hence can make use of ontologies to obtain knowledge based applications [CJB99, FGD92, FMR98]. In addition, a recent survey of the field suggests developers of practical software systems may especially benefit from ontology use [FH97, MA05, NHM00]. This survey earmarked several application classes of software engineering that benefit from using ontologies. These classes include generalized information system modelling, software requirement engineering, software reuse, software reliability, software security and abstraction [ABH+99, BOE96, DEV02, FMR98].

Ontologies are explicit representations of domain concepts and provide basic structure or armature around which knowledge based system can be constructed [ST99]. Ontology is a system of concepts and relations, in which all concepts are defined and interpreted in a declarative way [DEV02, GFC04]. System defines the vocabulary of a problem domain and a set of constraints that can be combined to model a domain. In a distributed environment, agents use ontologies to establish communication at the knowledge level using specific languages and protocols [BTD04]. Ontologies are explicit representations of agents' commitments to a model of the relevant world and hence enable knowledge sharing and reuse [JMY99, NFF+91, LINK9].

Ontological engineering encompasses a set of activities conducted during conceptualization, design, implementation and deployment of ontologies. Ontological engineering covers topics including knowledge representation formalisms, development methodology, knowledge sharing and reuse, knowledge management, business process modelling, systematization of domain knowledge, information retrieval, standardization, and evaluation [MIZ98, NFF+91, HOO98]. It also provides design rationale of a knowledge base, helps to define the essential concepts of the world of interest, allows for a more disciplined design of a knowledge based system, and enables knowledge accumulation [DW99, GOM01, SSS+01].

Several special issues of journals and magazines dedicated to the field of ontologies have described current trends in the field of ontologies. These include creating large-scale ontologies, defining expressive languages for representing ontological knowledge, and implementing systems that support ontology-based applications [CJB98, CJB99, LGS+99, LEN95, SRK+97, VRM+99]. Unfortunately, a vast majority of these articles have not covered the relations between ontological engineering and other software engineering areas. As a result, specialists from other disciplines struggle to understand the benefits of ontologies, and to map the terminology of ontological engineering to their own fields.

The organization of this chapter is done as follows: Section 2.2 elaborates ontological engineering, its themes and aspects of two general disciplines that help to develop ontologies at specification and conceptualization stage such as modelling and metamodelling. Section 2.3 emphasizes on goals of software practitioners. Moreover, it attempts to clarify the skills useful for practicing ontology based software engineering. Various disciplines of software engineering which cater by ontology are discussed in Section 2.4. Finally, we conclude in Section 2.5 with the summary.

2.2 Ontological Engineering

Ontological engineering field has been subject to considerable study and research during the last decade. It is observed that, ontological engineering refers to the set of activities that concern the ontology design principles such as clarity, minimal encoding bias, extendibility, and coherence [UG96, DEV02, GFC04]. According to principle of clarity, ontology should communicate effectively the intended meanings of defined terms. Also, definitions should be objective and defined by necessary and sufficient conditions. Whereas, minimal encoding bias principle suggests that the conceptualization should be done at knowledge level without depending on a particular symbol level encoding. Then, in line with principle of extendibility, ontology designer should be able to define new terms for special uses based on existing vocabulary. Lastly, principle of coherence states that ontology should sanction inferences that are consistent with the existing definitions [GRU92]. Now, we describe various themes of ontological engineering in this section.

2.2.1 Themes

Ontological literature on ontologies and ontological engineering usually covers the concepts shown in Figure 2.1. While, we put ontological engineering in the context of other disciplines, many similarities and analogies arise. These similarities allow practitioners to make connections between ontological engineering and other disciplines, to bridge comprehension gaps, and to see known concepts and practices in another light.

Naive Physics and Commonsense Knowledge

Ontology about naive physics is defined as the ontology for liquids developed by Hayes [HAY85]. Naive physics can be hard to formalize such as the problem with knowledge about liquids is that they have no definite shape and can merge split mix in mysterious ways. Formalizations of knowledge about physical objects can be found in [CLA81, S87, CRC95, BBW96b]. Ontologies of microscopic and macroscopic views on the electrical domain are combined by Liu [LIU92]. Conversely, aim of Cyc project is to build up a large knowledge base with commonsense knowledge [LG90, LINK9]. In addition, to help structuring knowledge in the knowledge base, ontology of common-sense top-level concepts have been developed [HOB85, DAV90].

Engineering and Technical Applications

Ontologies have been developed for engineering and technical applications. Ontology for the Sisyphus elevator design problem (VT) is described in [ST99]. In KACTUS project, ontologies for diagnosis of electrical networks and for the exchange of knowledge about ship design and oil platforms have been written [LAB+96]. YMIR ontology is a domain independent, sharable ontology for the formal representation of engineering design knowledge, based on systems theory [ALB93]. PHYSSYS ontology is less biased to a mathematical representation [BAT97]. Knowledge formalized in PHYSSYS has been used to develop number of applications such as a model revision assistant, OLMECO library of model fragments for simulation and for ecological product disassembly analysis. EngMath is an ontology used for mathematical modelling in engineering applications [GRU92]. It has been reused many times such as in PHYSSYS and in CML. CML is an ontology about time, continuity and object properties to enable the sharing of models based on compositional modelling [FFB+94, FF91 FOR84]. In addition, CML ontology has been used to develop ontology for thermodynamic systems and ontology for VT.

Penman Upper Model is a general model about natural language that can be used for the generation and processing of different languages such as Italian, German and English [BMR94]. Other ontologies formalize the semantics of the part–whole relation in natural language [GUA04]. Also, natural language about movement in the French language has been formalized [SAB93]. Ontologies are also used for the development of systems for extraction of knowledge [VRS99].

Corporate and Enterprise Modelling

TOVE ontology formalizes knowledge about production/ communication processes, activities, causality, resources, quality and cost in business enterprises [FCF93, GS02]. Ontologies have also been developed for the implementation of knowledge bases for formalization and conservation of the knowledge of experts in enterprises such as KONE ontology that deals with conservation of corporate knowledge about crankshaft design [GRU92].

Medical Diagnosis and Knowledge Acquisition

Knowledge in the medical domain about diagnosis, therapy planning and patient monitoring has been formalized in the GAMES-II project [FS94, HSW+97]. As ontology formally specifies meta-level domain knowledge, it can be an excellent specification for tools that acquire knowledge from domain experts such as PROT 'EG'E-II project [FGD92].

2.2.2 Specification and Conceptualization

We have observed that, desirable qualities for ontologies such as being decomposable, extensible, maintainable, modular and interfacable tied to the information analyzed, universally understood, and translatable characteristics [FGJ97, GUA04]. Also, these are desirable for interoperable software components or classes of objects in object-oriented design [KG02, GFV96, SMJ02, WAR09]. Practitioners from other fields may use different terminology but its meanings are often similar. Hence, Figure 2.2 shows aspects of two general disciplines that can help develop ontologies at the specification and conceptualization stage namely; Modelling and Metamodelling [AW06, GL02]. In practice, knowledge of these disciplines helps to organize the knowledge acquisition process. Moreover, it specifies the ontology's primary objective, purpose and scope. Lastly, it builds initial vocabulary and organizes taxonomy in an informal or semiformal way and possibly using an intermediate representation [SBF98, MS00, RL02, VSS+05].

Modelling

Ontologies are specific, high-level models of knowledge underlying all things, concepts, and phenomena [GS02, GHW02]. While with other models, ontologies not represent the entire world of interest. Rather, ontology designers select aspects of reality relevant to task [CJB98, VRM+99]. For example in domain of books, ontology designer selects one set of book attributes when developing ontology of a library, and different set when developing ontology of bookbinding. All models follow principles and constraints termed as concept relations and axioms [SMJ02]. Although, there exist different ways to represent ontologies such as ontological engineers most frequently use hierarchical modelling at conceptualization level [DR99, LEN95]. These ontologies represent concept hierarchies and taxonomies in layers and use pictorial representation to visually enhance representation [FGD92]. Layers in ontology representation range from domain-independent to task and domain-specific. As a result, ontologies contain knowledge of appropriate hierarchical and or layered models of relevant world [GFV96, GF95].

Metamodelling

Conceptualizing and specifying ontologies have a strong metamodelling essence [GS02, GHW02]. A metamodel of a modelling technique, improves the rigor of different but similar models [FS97]. Ontologies accomplish same for knowledge models. Without ontologies, knowledge bases representing knowledge of the same domain are generally incompatible while using similar knowledge models [CAC01, HOO98, NFF+91]. Metamodelling is preferred because it allows practitioners to preserve the usefulness of any specific model. Ontology simply provides the skeleton for the corresponding models of the domain knowledge [GFV96, GF95]. Generally, ontology is a metamodel describes a way to build models. Its components such as the concepts it defines and the relations between them are always (re)used as building blocks when modelling the parts of domain knowledge [MF03, TL00, SMB07]. When developing a practical software system, it helps the mechanism that with built-in knowledge of the models to be deployed and makes the development mechanism intelligent [KL02].

Many potentially useful parallels exist between ontological engineering and software engineering disciplines such as software architectures and software patterns [GF95, SS99]. It is observed that, many practitioners understand similarities between phases of the ontology development and software development processes [EW05, WAR09]. Thus, there exist potential scope of research and software practitioners can benefit from knowing more about such useful parallels. To accomplish this, software practitioners must seize a range of goals with the help of ontological engineering.

2.3 Goals for Software Practitioners

For software practitioners, ontology based development is extremely important to attain software knowledge acquisition, exchange and reuse [BAR06, DEV02, GUA98, CJB99]. However, ontological engineering enable achieving a range of goals such as to precisely define terms and highly structured definitions of domain concepts, not text-based information. Next, it provides consensus knowledge of a community of people and high expressiveness. Then, coherence and interoperability of resulting knowledge bases is made available by it. In addition, it endows with stability and scalability of ontologies. Lastly, it organizes a foundation for solving a variety of problems and constructing multiple applications.

Although, ontologies are content-related than representation-related and achieving these goals calls for formalization and co-existence of artistic creativity and systematically applied knowledge from other disciplines [DW99, FCM+03]. Ontology can be developed collaboratively by many distributed individuals and organizations with differing expertise, goals, and interactions. Various communities of experts and practitioners examine problems from different perception and are concerned with different dimensions of the content's semantics and representation [MA05, NM04, SSS+01]. These individuals need to properly understand each other and meaningfully communicate their views of domain knowledge to form meaningful higher-level knowledge [OVR+06, TA00].

Once application developers are ready to use the ontology, developers should be able to convert it into a desired form, such as object base or knowledge base, using representation methods [GRU92]. Hence, ontological engineering must rely on several content formats, frameworks, and development strategies that reduce semantic ambiguity and allow for sharing and reusing knowledge and practices from other disciplines. In addition, ontological engineering involves developing higher-level knowledge-based products that express the consensus knowledge of a community of agents [SMB07, MC06, CHK+07]. Certain software engineering disciplines and issues rarely discussed by ontology researchers that can help software engineers and practitioners. These include knowledge intensive information system modelling, software requirement engineering, software reuse, software reliability and software security and abstraction, as presented in Figure 2.3.

2.4 Software Engineering Disciplines

Software developers consider ontologies as a trend involving methodology and technology as AI community develops ontologies that use special-purpose mechanisms [SMJ02, WAR09]. However, ontology is always about entities and relationships. In addition, methodology from traditional software engineering such as using ER model, top-down decomposition strategy and structured system analysis are used to represent it. For example, Methontology framework for developing ontologies proposes a close relative of traditional waterfall model of software development for an ontology development lifecycle [FGJ97, LGS+99]. Moreover, entire ontology developed using Methontology framework is stored in a relational database and can encode its ontology in its data dictionary [MIZ98, FH97]. All design criteria for ontologies, such as clarity, extensibility, coherence, and minimal encoding bias also represent design criteria for software systems modules [FCM+03, GF95]. Ontology researchers and developers can explore a large variety of iterative and incremental traditional software development methodologies for new ideas in ontology based software engineering. We consider some of them and discuss as follows:

2.4.1 Ontology Driven Information System

Ontology development process nearly coincide with those of object-oriented software development [LGS+99, MIZ98, VRM+99]. In both cases, it is important to assemble domain vocabulary in the beginning, often starting from the domain's generic nouns, verbs, and adjectives [FS97, ST99, GFV96, GF95]. Object-oriented analysis stresses different aspects than ontological analysis and yet analogous [MIZ98, JAC92, SMJ02]. The result of object-oriented analysis is a draft of domain ontology relevant to the application. Besides, as object-oriented designers define classes, objects, hierarchies, interface functions and system behavior, ontological engineers use intermediate representations such as semantic networks, graphs, and tables to design hierarchies and other concept relationships [CAC01, FEA+02]. Both types of specialists use templates to specify product details [FS97, LGS+99]. Classes can be merged or refined with ontologies. Class libraries and previous design specifications often provide reuse in object-oriented design with the help of previously encoded and available ontologies [FMR98, JCJ+07, WAR09].

In addition, important differences exist between Ontology Development Life Cycle (ODLC) and Object Oriented Software Development Life Cycle (OOSDLC) from practitioner's perspective. ODLC signifies knowledge-level stance in describing system, while OOSDLC largely refers to the means of design and implementation [CJB99, HOO98]. For example, in semantic-based information retrieval system, ontologies specify the meaning of concepts to be searched, while object-oriented design represents domain models [KL02, MS00]. Object-oriented design languages such as UML offers explicit design methodology and notation for all design artifacts, but ontological and metamodelling principles are only implicit in those languages [MIZ98, BKK+02]. In other words, ontology is abstracted at knowledge level from corresponding class diagrams, object diagrams, and use-case diagrams, represented in any object-oriented notation such as UML [SBF98, WER+97, SK03]. The role of ontology is to convey and explicitly specify domain concepts, terms, definitions, relations, constraints, and other semantic contents that object-oriented analysis and design should rely on and support [EMB04, GUA04, MC06, SS06, SK04].

It is observed that one area of ontology based software engineering requires additional efforts involves developing a generally accepted notation for representing ontologies. Software engineers have used several different notations in object-oriented design over the past decade, but all have converged to UML notation, which provides a metamodel of object-oriented design. It defines graphical notation for representing classes, objects, and relationships, covering all practical aspects of object oriented design [JAC92, JCJ+07]. But, ontological engineering strive standard notation that is accepted, understood, and used in practice [VRM+99 FS97, WAR09].

2.4.2 Software Requirement Engineering

The basics of ODLC involve designing and specifying overall system structure and underlying organization [GUA98, GUA04, HOO98]. Ontologies are architectural armatures for building knowledge bases, models, and software structural designs [SBF98, ST99]. This assertion helps requirement engineers to build requirement specifications of any application. Structural design style in the field of requirement engineering, characterizes a family of systems related by shared structural and semantic properties [SHA95, LV02, LV04]. A style typically defines a vocabulary of design elements, design rules for compositions of elements and semantic interpretation of design element compositions [VAN03]. Many successful designs can share a style. Styles contain condensed skeletons of the architectural knowledge gained by experienced software designers, and provide a means to reuse that knowledge [BOE96, BOS95, LPR93].

In addition, ontologies structure knowledge in form of layers to separate use-specific knowledge from more reusable knowledge [MIZ98, VRM+99]. Other structural design styles help to define requirement engineering solutions include pipeline and data abstraction [SHA95, BL03, EW05]. In addition, layered style is suitable for applications involving distinct classes of services that can be arranged hierarchically. It is explored that, researchers have proposed layers for building requirements by considering basic system level services and utilities appropriate to many applications [ST99, SG05, SPL06, ZZY+07]. But, it lacks in specific application task that depends on requirement type to make knowledge intensive requirement engineering process.

2.4.3 Software Reuse

Early software reuse practices focused on code and made ad hoc. However, reuse changed as the industry matured. Reuse became planned and systematic [ABH+99, DEV02, GUA98]. Currently, any product of the software life cycle can potentially be reused. According to some researchers the active areas of reuse research includes reuse libraries, domain engineering methods and tools, reuse design, design patterns, domain specific software architecture, component, generators, measurement and experimentation [MA05, HAM04, NM01, RL02]. However, ideas emerging from this period lacks in reuse design principles, commonality and variability analysis, and various approaches to knowledge generators [FK05].

A significant objective of ontology is to build reusable knowledge components and knowledge-based services that can be invoked over networks [SZY98, REI97, MVI95, SS99]. Consequently, software engineering field is attempting to develop repositories of reusable, pretested, interoperable, and independently upgradable software components that enable plug-and play design. These objectives necessitate designing systems from application elements constructed independently by developers using different languages, tools, and computing platforms [SMB07, WZX06]. Ontologies can precisely define the semantics of components as well as the types of relations and communication between software components [HAM04, MERM03, LINK6]. Consequently, ontologies are used to enable a basis for designing and developing interoperable software components in practice.

2.4.4 Software Reliability

Software reliability is probability of failure free operation of a computer program in a specified environment for a specified time [FH97, LEN95]. Software reliability has been discussed, in a number of studies on software reliability evaluation focusing on post-software development that includes reliability modelling, reliability estimation and tools development [LGS99, MIZ98, SHA95]. Reliability evaluation taking place prior to software development is attracting a growing attention among software architects and reliability experts. This issue tackles by introducing an ontology-based approach. This approach is characterized with integration of software reliability engineering and software architecture design [GHJ+95, LEN95, SZY98]. In particular, this approach suggests software architecture design as the first phase of evaluating reliability in the development of software systems [VRM+99]. However, reliability can be accomplished at requirements analysis phase.

2.4.5 Software Security

Security analysis and design involves the identification of security attributes and the design of solutions that address these attributes in an efficient and effective manner [MGM03]. Effective software security control has been emphasized mainly to ontology based projects due to its expediency, flexibility and comprehensibility. Consequently, it needs methodology of improving the current posture of project security while developing these projects with various perspectives [SI11a, SI11b, SI11d]. It attempts to provide a range of benefits related to ontology based projects. It is observed that these benefits may incur different unvisualized states and for ensnaring these states, ontology based software projects thereby indulge with the instinct for security attributes. Conversely, the involvement of these factors may be horded in such a way that may render to acquire the security perspective of ontology based software projects.

2.5 Summary

In this chapter, we have highlighted the importance of ontologies in software development process while presenting a literature survey component of the thesis. We present ontological engineering in the context of other disciplines and observe that it enables both ontological engineers and other specialists to view respective fields from different perspectives. In this view, goals of software practitioners are covered to consider awareness of such similarities In addition, these enable to create new ways to build and improve knowledge. Chapter proceeds with introduction of various software engineering disciplines information system building, requirement engineering, reuse, reliability and security. We observe that, ontologies are needed in all software engineering disciplines to explore entities, attributes and relationships in the relevant world. Moreover, all disciplines require knowledge that constitutes data structures, methods, or algorithms.

CHAPTER 3

Ontology Driven Information Systems

3.1 Introduction

The rapid development of new application domains has introduced important changes to information dissemination and application processes [KL02]. It has been observed that, contemporary Information Systems are increasingly distributed and heterogeneous. Consequently, the next generation of Information Systems ought to solve the semantic heterogeneity to formulate the information available. Ontology plays an essential role in the construction of Information System since it allows the establishment of correspondences and interrelations among the different domains [GUA98, HOO98]. In order to develop a long term oriented and extremely generalized software, Ontology Engineering (OE) has been practiced in recent years.

Ontology Engineering (OE) presents the paradigm of choice for growing number of Information Systems, over Object Oriented Software Engineering (OOSE) approach. OE approach stresses different aspects than OOSE. Ontology development and Object Oriented software development have their own, concurrent, intertwined life cycles which have something in common but also differ in their goals, responsibilities, time horizons etc. [WAR09]. The role of ontologies is to capture domain knowledge in a generic way and to provide a commonly agreed upon understanding of a domain. The common vocabulary of an ontology, defining the meaning of terms and their relations, is usually organized in taxonomy and contains modelling primitives [GFC03]. In addition, Ontology is well known as description of declaration. While building a new ontology, analysis phase serves for initiating overall ontology development process. The ontology domain is conceptualized, the glossary is (further) filled, enhanced, extended and cross-references are determined in design phase. During the construction or implementation phase, ontology is translated into a concrete ontology with the help of programming language [MS00, GFV96].

In recent scenario, use of procedure oriented approach is discouraged in software development organizations for developing the Information Systems as Object Oriented Technology has grappled its importance. During the software development, it has been observed that the current candidate system is being intended. Accordingly, resultant software is restricted to a particular application. Nevertheless, these concepts works effectively on domain oriented software development projects. However, OE approach will enable the software practitioners to integrate the concerned information in a seamless and a flexible manner [SSS+01, SBF98]. Thus, global networking and continuous development of new application domains involve changes in information dissemination and application processes.

3.2 Background

In recent years, it has been recognized that the use of ontologies are advantageous for software engineering. Ontology representations are little known outside AI research laboratories. In contrast, commercial interest has results in ideas from object oriented programming community maturing into industry standards and powerful tools for object oriented analysis design and implementation. And, this maturing standards and tools can be used for ontology modelling. Ontology has been known as formally specified models of bodies of knowledge, defining concepts used to describe a domain and interrelationship that holds between them [FGJ97, SRK+97]. The object oriented paradigm in software engineering, influencing all effort in information science and is one of the main objectives of the software engineering discipline. It has been observed that, Object Oriented Modelling (OOM) is different from other modelling techniques because it has merged the concept of variables and abstract data types. This abstract variable type termed as an Object. Objects have identity, state, and behavior and object models built out of systems of these objects. To make it easier, there exist concepts of type such as inheritance, association, and class. OOM focuses on identity and behavior and therefore, is completely different from the relational model's focus on information [JAC92]. However, Ontology is well known as description of declaration and abstract way to define the domain information. It involved with vocabulary and constrains the use of data widely in software development, which requires a significant degree of structure. Also, ontology, allow expressing the similarity of concept in OOM [SMJ02]. Thus, ontologies have been come up as an important tool for coping with very great, compound and various sources of information [GF95, GUA04]. Now, we describe information systems, Object Oriented SDLC and Ontology Development Life Cycle in this section.

3.2.1 Information Systems

An information system is any combination of information technology and people's activities that support operations, management and decision making. In a very broad sense, the term information system is frequently used to refer to the interaction between people, processes, data and technology. These are mainly classified into Transaction Processing System (TPS), Management Information System (MIS), Office Automation System (OAS), Decision Support System (DSS) and Expert System (ES) [LINK1].

Transaction Processing System (TPS) collects, stores, modifies, and retrieves the transactions of an organization. A transaction is an event that generates or modifies data that is eventually stored in an information system. Examples of such system include Airline Reservation System Payroll Processing System, Transport Ticket Reservation System, and Purchase Order Entry System etc. However, MIS provides information essential to manage organizations efficiently and effectively. MIS encompasses three primary components such as technology, people (individuals, groups, or organizations), and data/information. Sales Order Entry, Hotel Reservations and Payroll System etc. include the examples of MIS. Next, DSS has been defined as a computer-based information system that supports business or organizational decision-making activities. A properly designed DSS is an interactive software-based system intended to help decision makers, compile useful information from a combination of raw data, documents, and personal knowledge, or business models to identify and solve problems and make decisions. Generally, DSSs are interactive, flexible, and adaptable information systems, developed to support the solution of non-structured management problems for improved decision making. For example, medical decision making often involves making a diagnosis and selecting an appropriate treatment.

Now, OAS refers to the varied computer software used to digitally create, collect, store, manipulate, and relay office information needed for accomplishing basic tasks. Office automation helps in optimizing or automating existing office procedures. Raw data storage, electronic transfer, and the management of electronic business information comprise the basic activities of an office automation system. Lastly, ES helps to guide users to find solutions to problems and is useful in diagnosing, monitoring, selecting, designing, predicting and training.

3.2.2 Object Oriented SDLC (OOSDLC)

In the course of object-oriented software development, various models such as Requirement, Analysis, Design and Implementation have been constructed at the different stages as shown in Figure 3.1. This modelling practice is helpful because of seamless transitions between the models and simple traceability maintenance [JCJ+07]. Requirement Model is confined at the functional requirements. It is comprised of a Use Case Model and Object Model. Use Case Model constitutes actors and use cases, supported by an intuitive domain and interface descriptions. Actor indicates interaction with the system and use case specifies a flow that a specific actor invokes in the system. Then, Object Model provides a conceptual, easy to understand picture of the system.

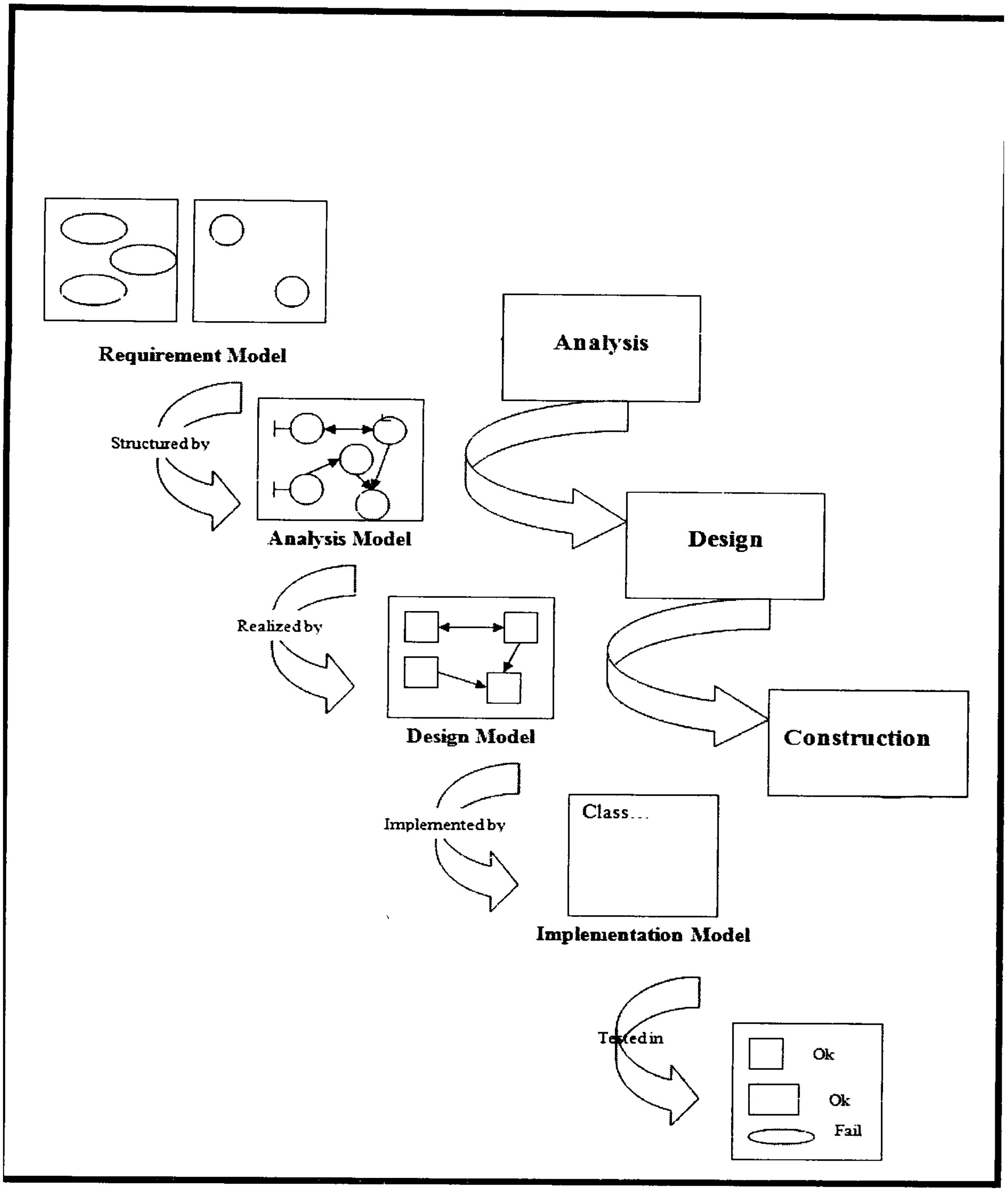


Figure 3.1: Object Oriented Software Development Life Cycle

Next, Analysis Model aims to present a robust and changeable object structure. During this phase Requirement Model is structured into Analysis Model. Three types of objects namely Entity objects, Interface objects and Control objects are used. The use case functionalities which are directly dependent on the system's environments are handled in Interface objects whereas tasks dealing with the storage and handling of information are placed in the Entity objects. Finally, functionalities specific to one or few use cases and not placed in any of the other objects are placed in Control objects. Consequently, Design Model is intended to adopt and refine the object structure to current implementation environment. This model has been regarded as a formalization of Analysis Model. It defines the structure and hierarchy, interfaces, rules for commitment and block is used to represent design object. One block implements one analysis object. Lastly, Implementation Model helps to implement the system. It is evident that the base for system implementation is Design Model [JAC92]. Now, we present examples of MIS, TPS and ES that have been developed using object oriented software engineering concepts.

Hospital Management System illustrates the example of MIS. The system has several types of functions represented through use cases. These use cases associated with different possible actors as shown in Figure 3.2. Figure 3.3 shows Object Model demonstrating its role in formulation of use case descriptions of the use cases. Figure 3.4 illustrates the Analysis Model that has eight interface objects namely Staff or Nurse Panel, Doctor Panel, Receptionist Panel, Information Panel, Patient Panel, Record System Panel, Expense Panel, and Income Panel to communicate with system. Entity objects such as Expenditure and Income inherit Finance Manager and five control objects specifically Appointment Fixer, Receipt generator, Login, Salary Manager, Bill issue to unite the other objects to get overall system functionality. Figure 3.5 shows Design Model using the blocks as an abstraction of the actual implementation. Each block corresponds to every object of Analysis Model showing their functionalities. Figure 3.6 shows the Implementation model (Class diagram) that implements each block of Design Model into corresponding classes.

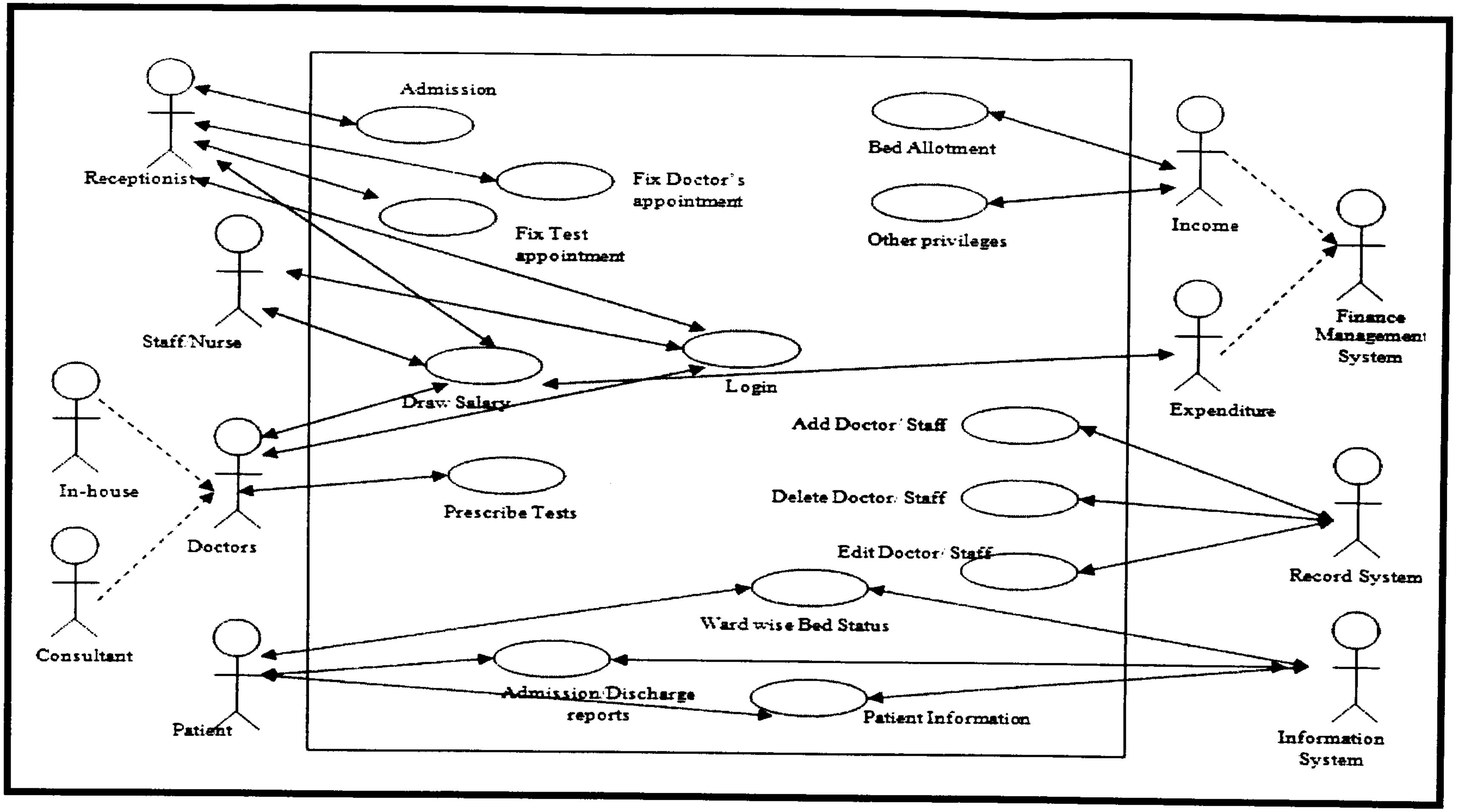


Figure 3.2: Use Case Model of Hospital Management System

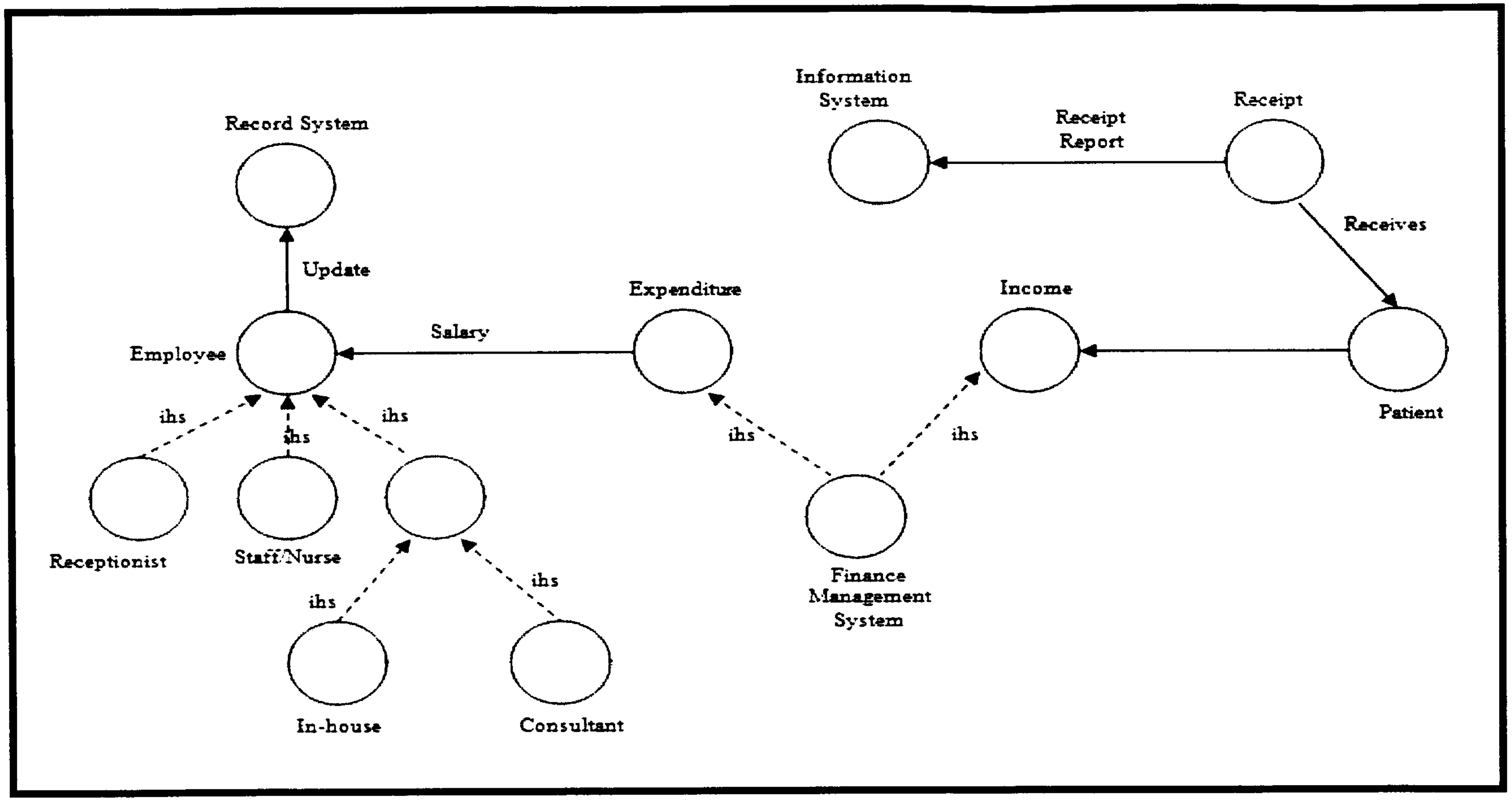


Figure 3.3: Object Model of Hospital Management System

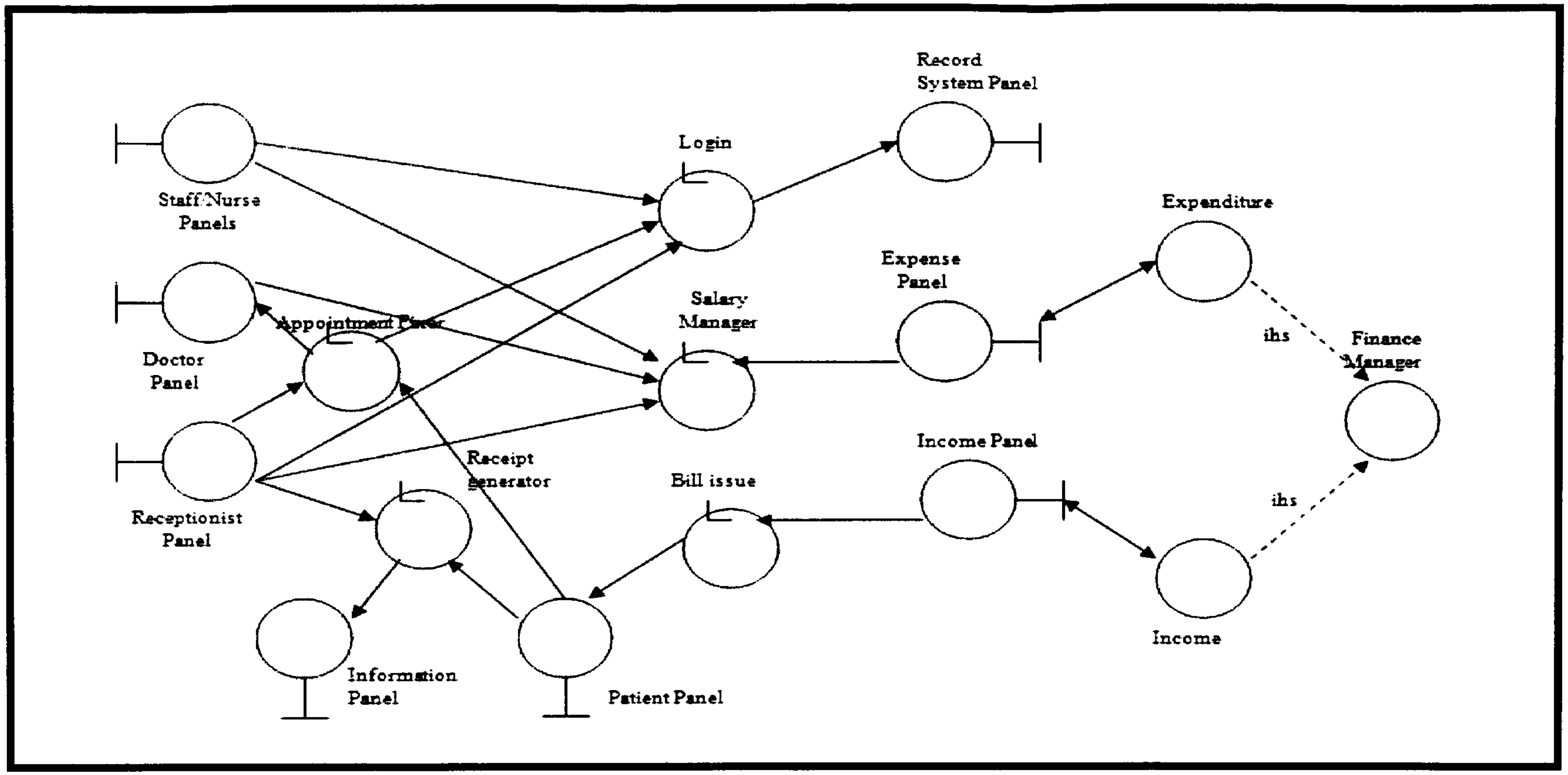


Figure 3.4: Analysis Model of Hospital Management System

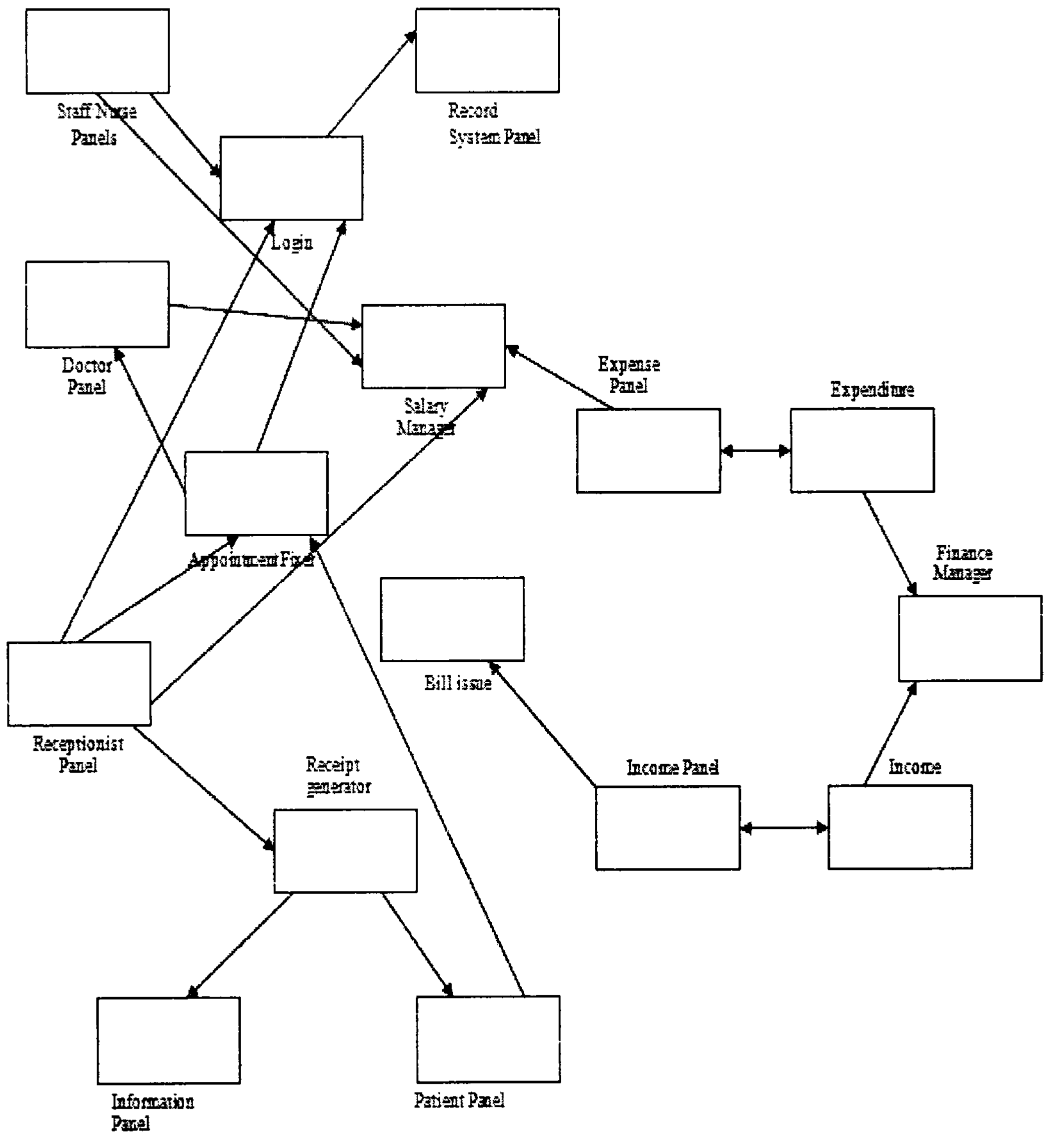


Figure 3.5: Design Model of Hospital Management System

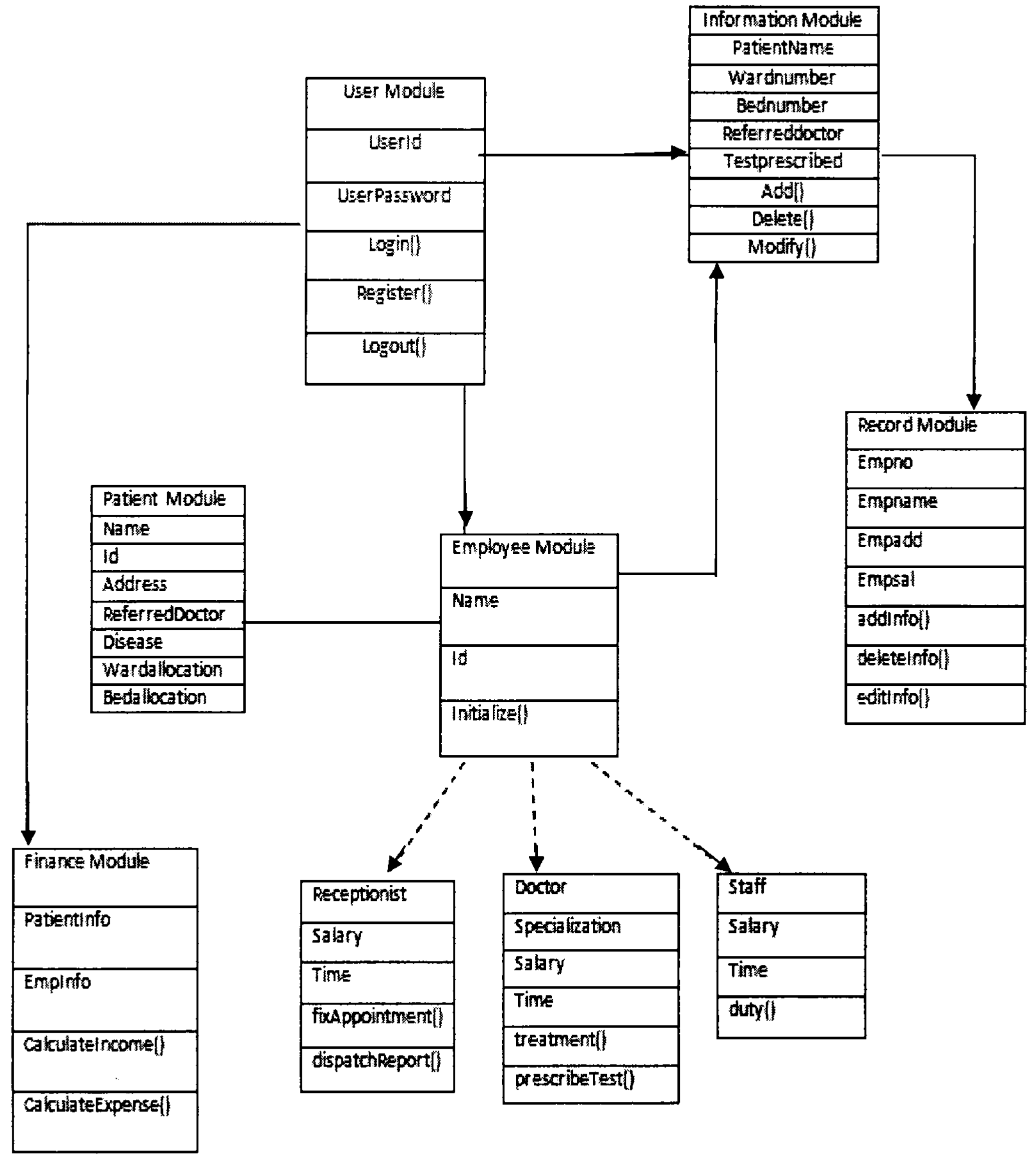


Figure 3.6: Implementation Model of Hospital Management System

Next, Railway Reservation System exemplifies TPS. It has several different types of functions represented through use cases like Request for availability which includes Date, Required train and Reservation type, Request form, Fill form, Submit form, Accept form, Issue ticket, Make a reservation, Make cancellation and Update data base associated with different possible actors. These actors are Traveler, Booking Clerk and Reservation Data base working in a system as shown in Figure 3.7. Figure 3.8 shows Object Model showing its role in formulation of use case descriptions of the use cases. In addition, Analysis Model has four interface objects such as Traveler Panel, Booking Panel, Reservation database Panel and Reservation Panel to provide communication of user with the system. An entity object called Train Information is provided since it sustains in a system till the traveler is registered. Six control objects such as Availability Checker, Ticket issuer, Database updater, Ticket canceller, Amount checker and amount checker has been included to get overall system functionality as shown in Figure 3.9. Lastly, Figure 3.10 illustrates Design Model using the blocks whereas Figure 3.11 presents Implementation Model for the railway reservation system.

Now, Recycling Machine System demonstrates the ES. Use Case Model of this system has been comprised of use cases such as Returning Item, Generate Daily Report and Change Item associated with Customer and Operator actors as shown in Figure 3.12. Whereas, Figure 3.13 shows Object Model that includes objects such as Deposit item classified as Can, Bottle or Crate. Then, Figure 3.14 depicts Analysis Model that has four interface objects such as Alarm device, Customer Panel, Operator Panel and Receipt Printer to provide statement of user with the system. Subsequently, entity objects such as Receipt basis and Deposit item has been involved since these remain constant till system works. In addition, three control Objects such as Alarmist, Deposit Item Receiver and Report Generator has been included to unite system functionality. Finally, Figure 3.15 shows the Design Model using blocks and Figure 3.16 confirms the Implementation Model for Recycling Machine System.

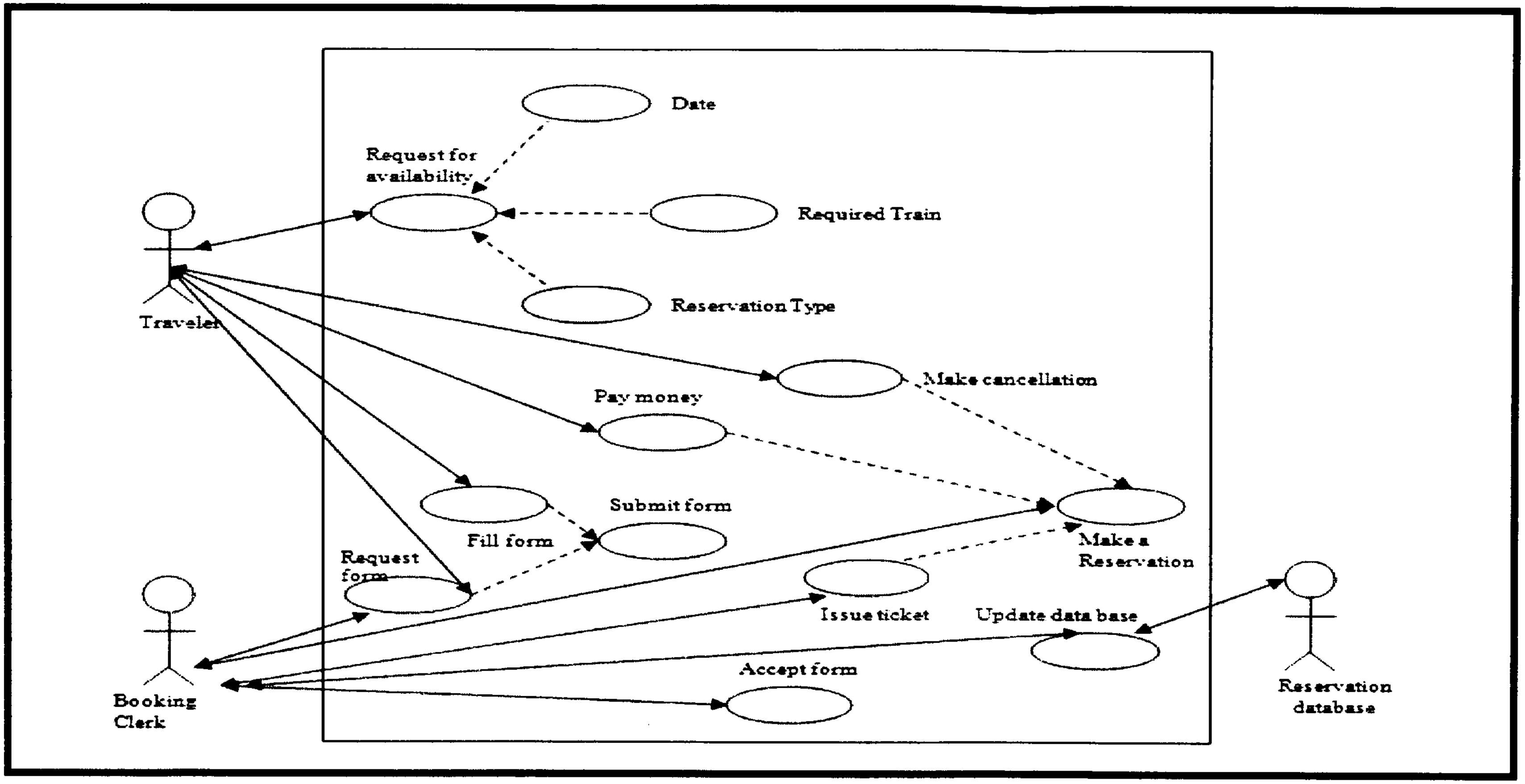


Figure 3.7: Use Case Model of Railway Reservation System

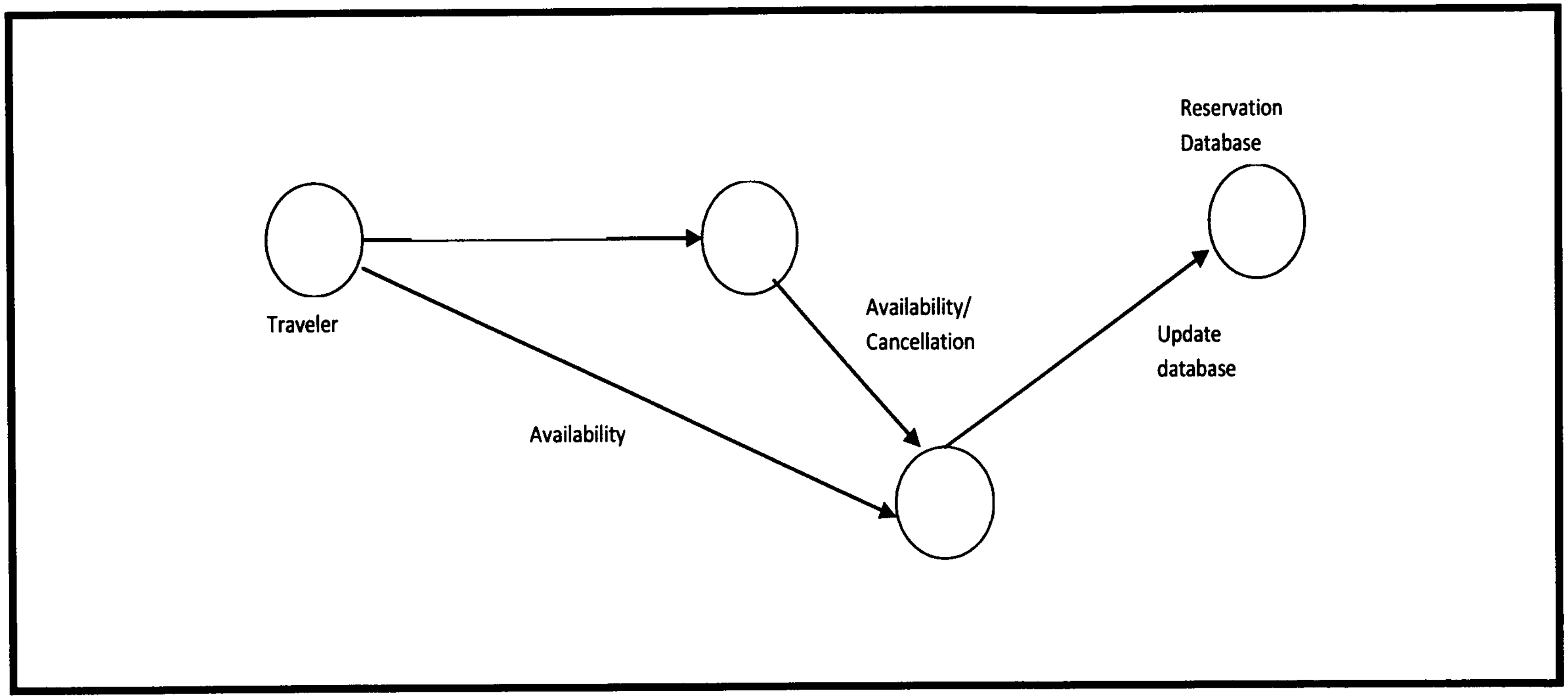


Figure 3.8: Object Model of Railway Reservation System

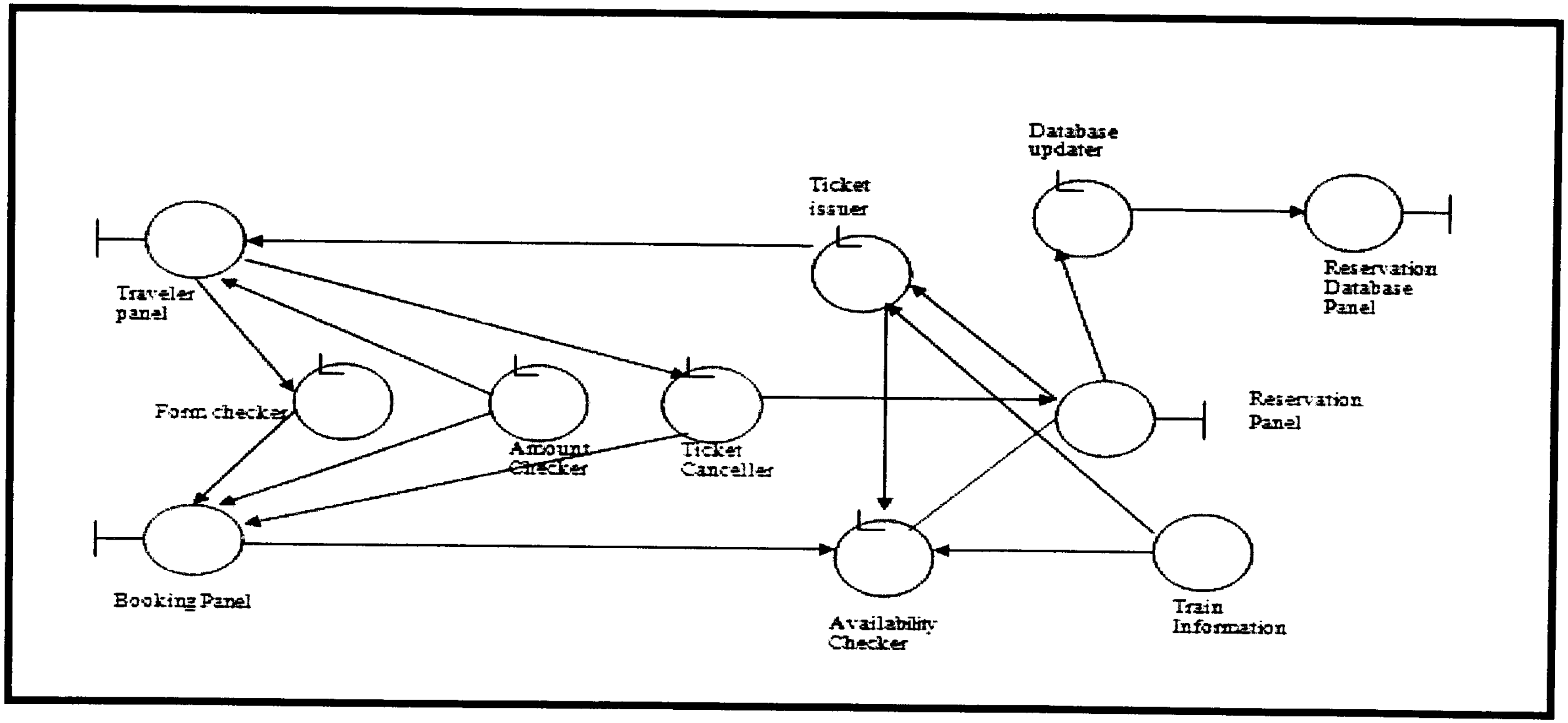


Figure 3.9: Analysis Model of Railway Reservation System

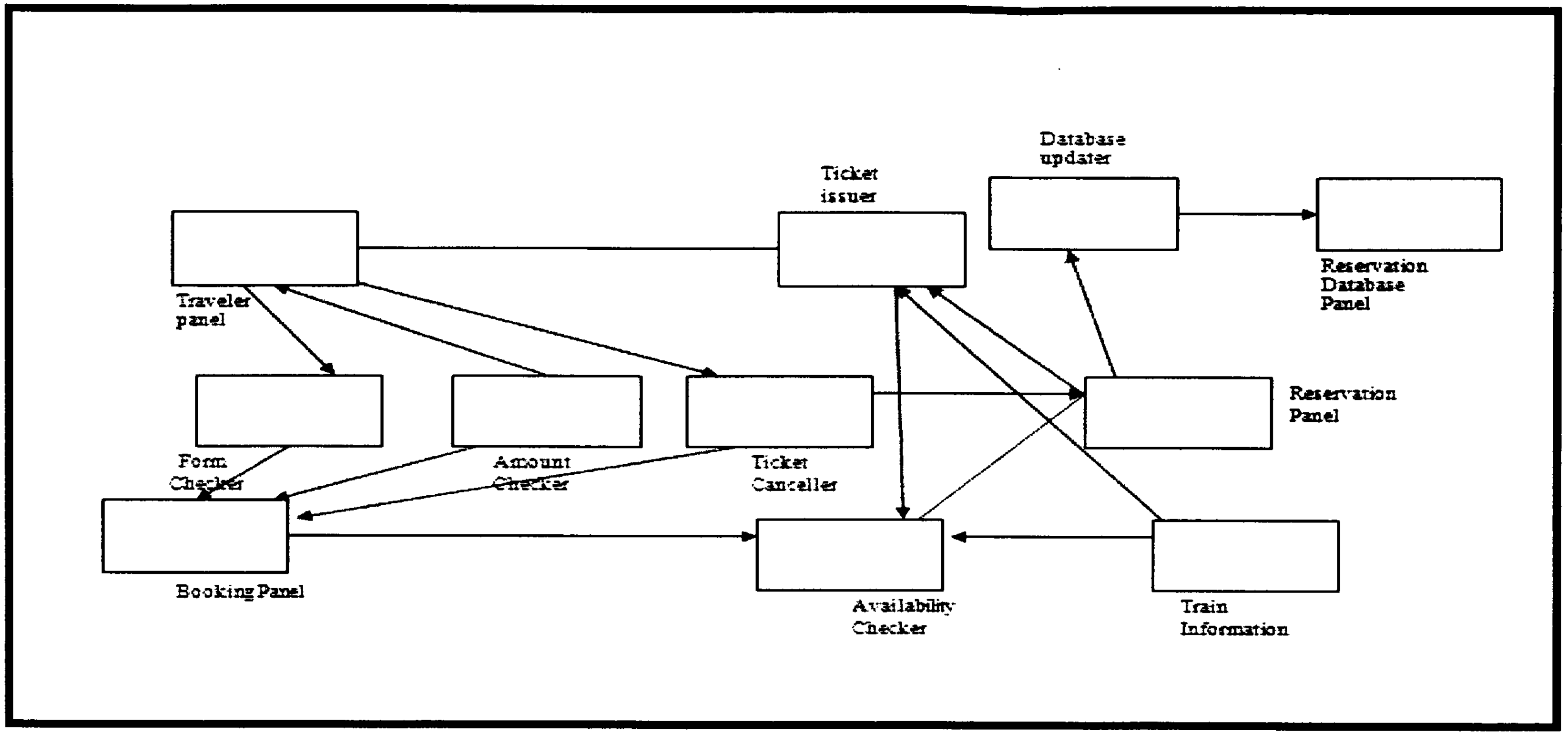


Figure 3.10: Design Model of Railway Reservation System

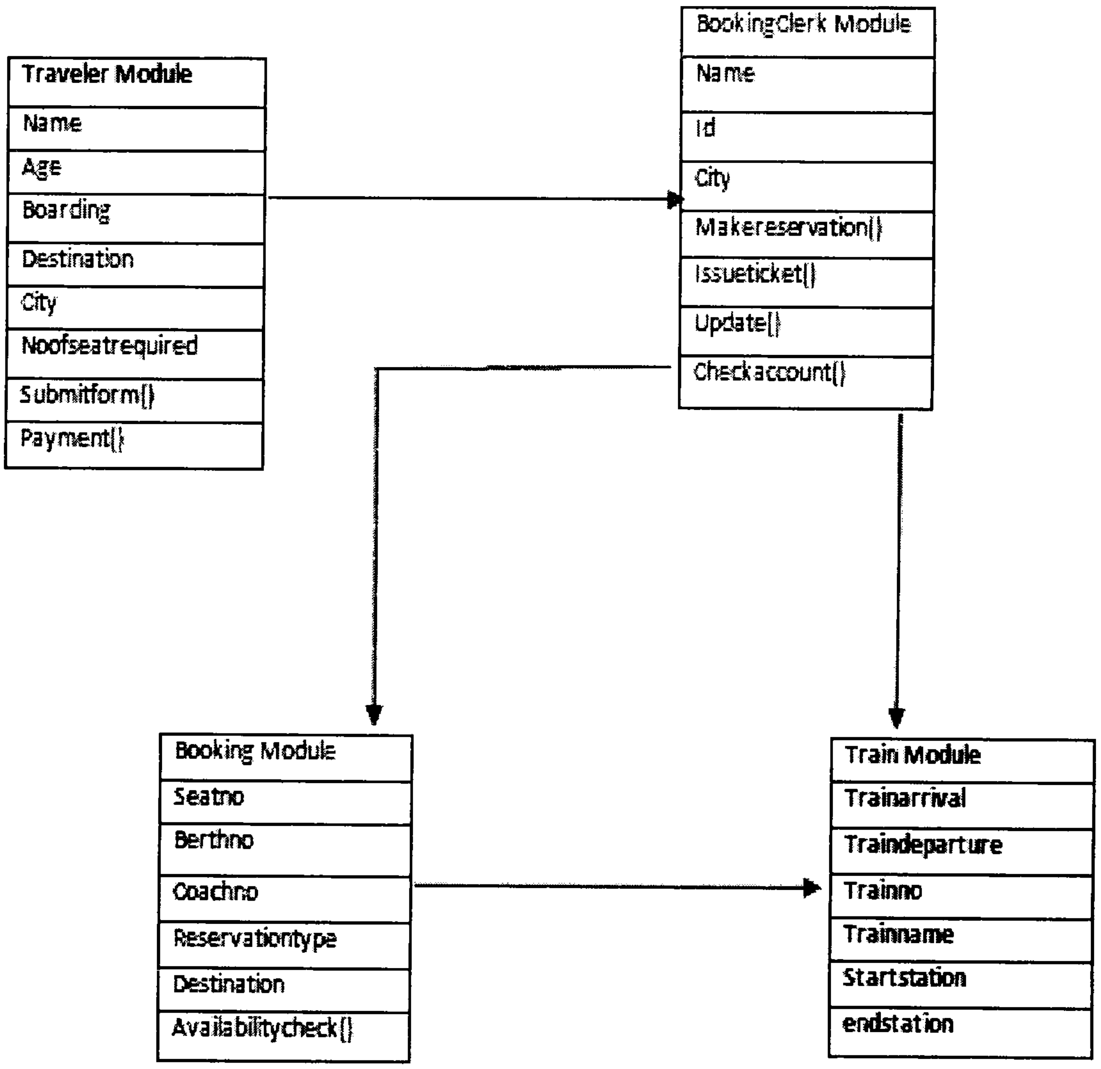


Figure 3.11: Implementation Model of Railway Reservation System

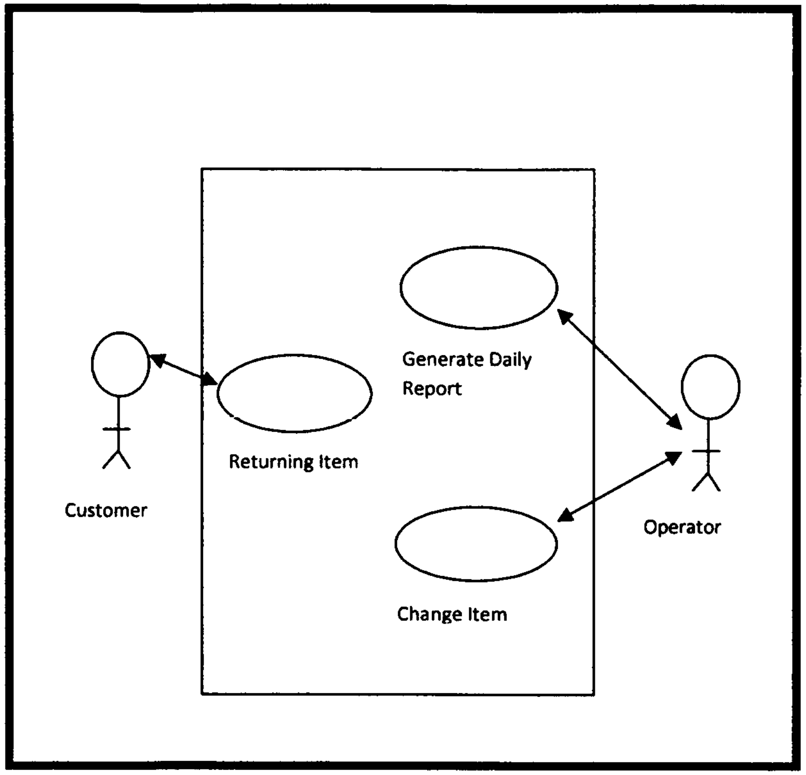


Figure 3.12: Use Case Model of Recycling Machine System

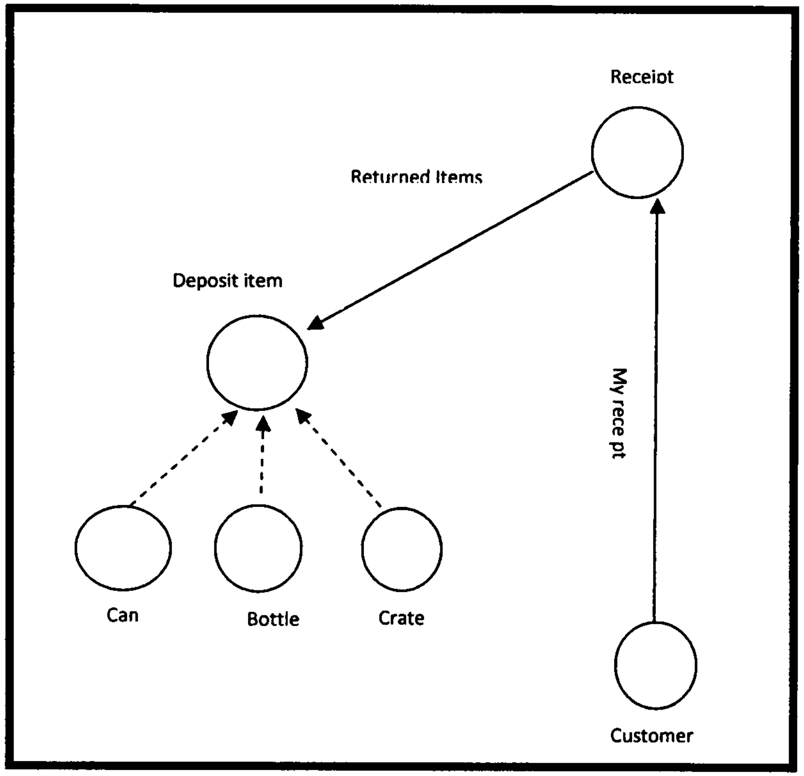


Figure 3.13: Object Model of Recycling Machine System

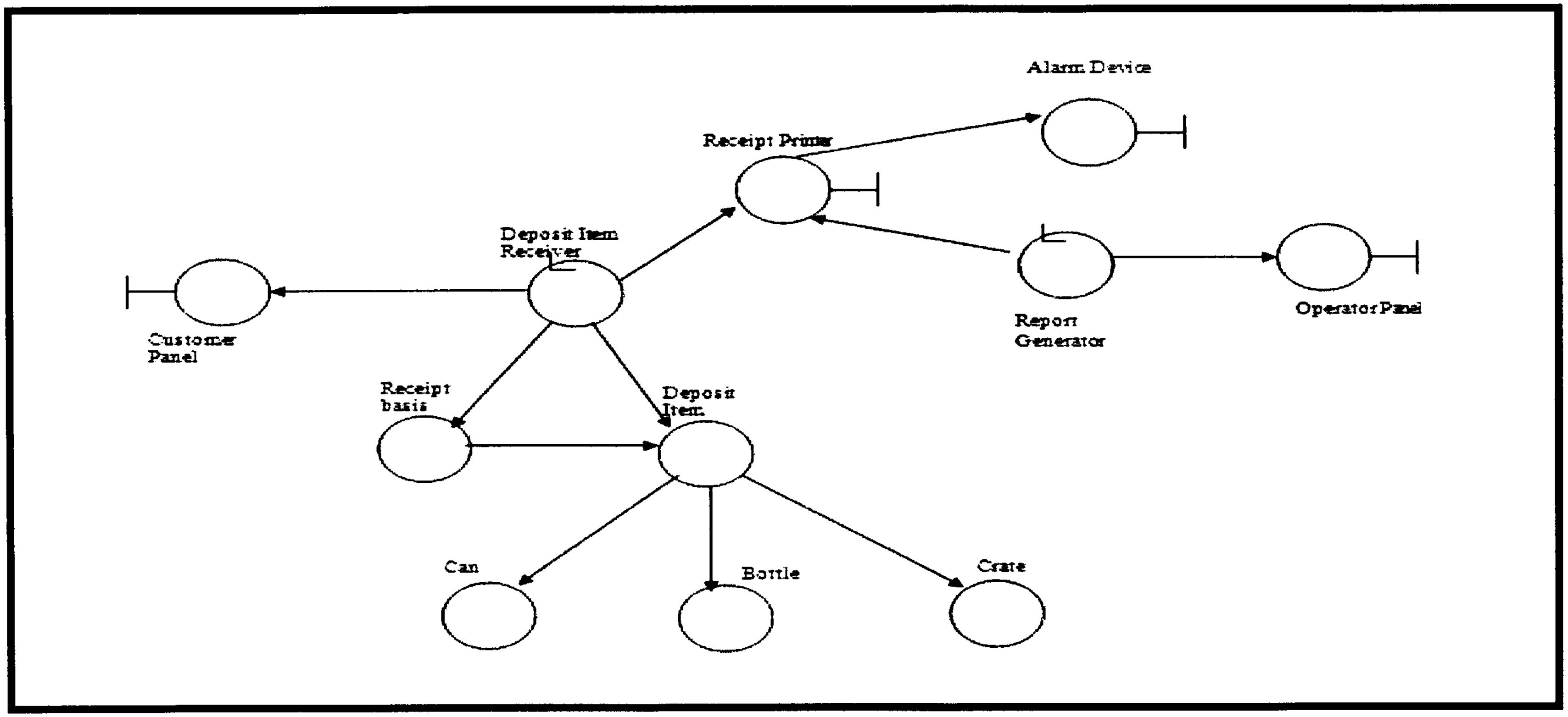


Figure 3.14: Analysis Model of Recycling Machine System

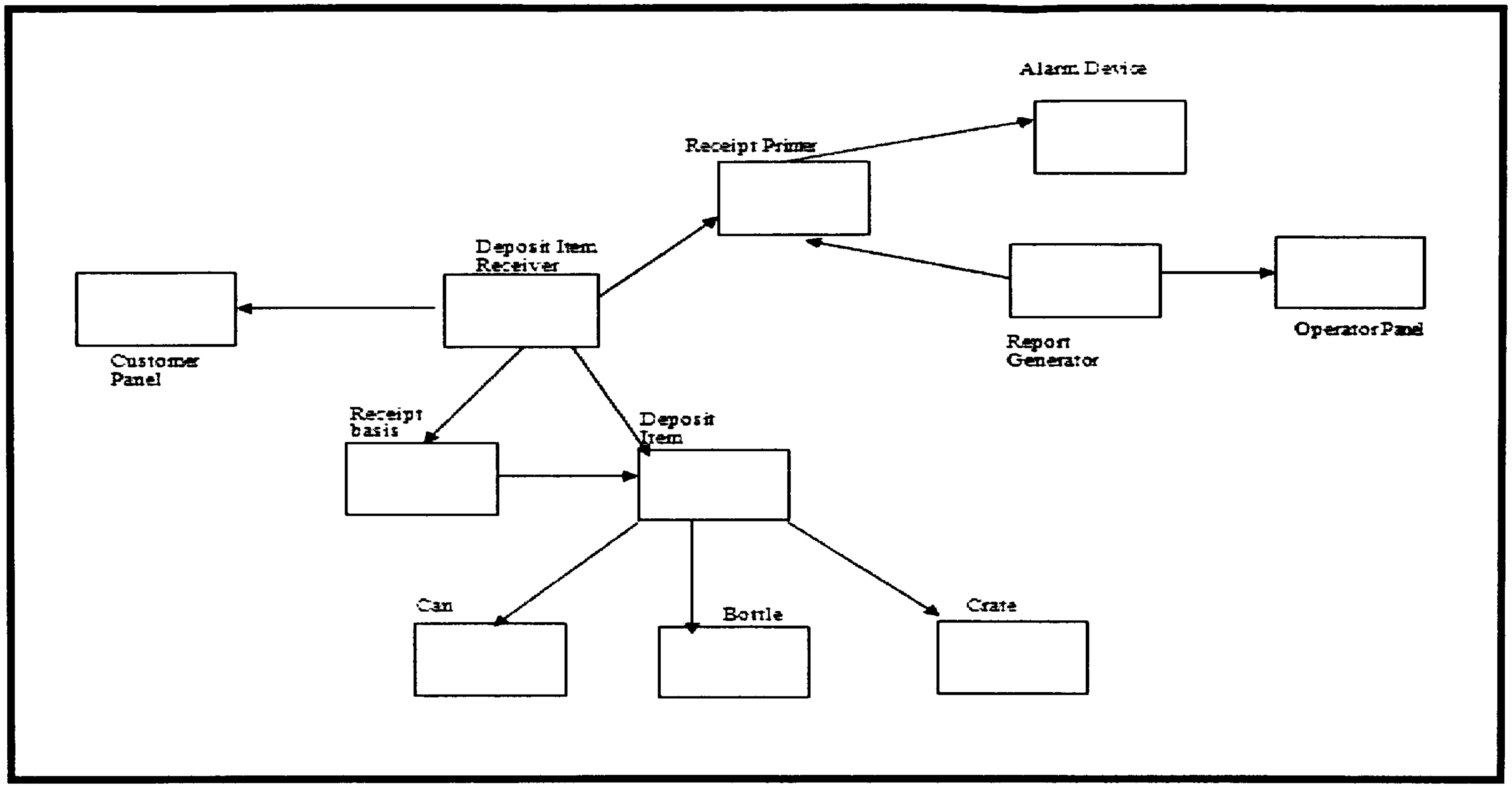


Figure 3.15: Design Model of Recycling Machine System

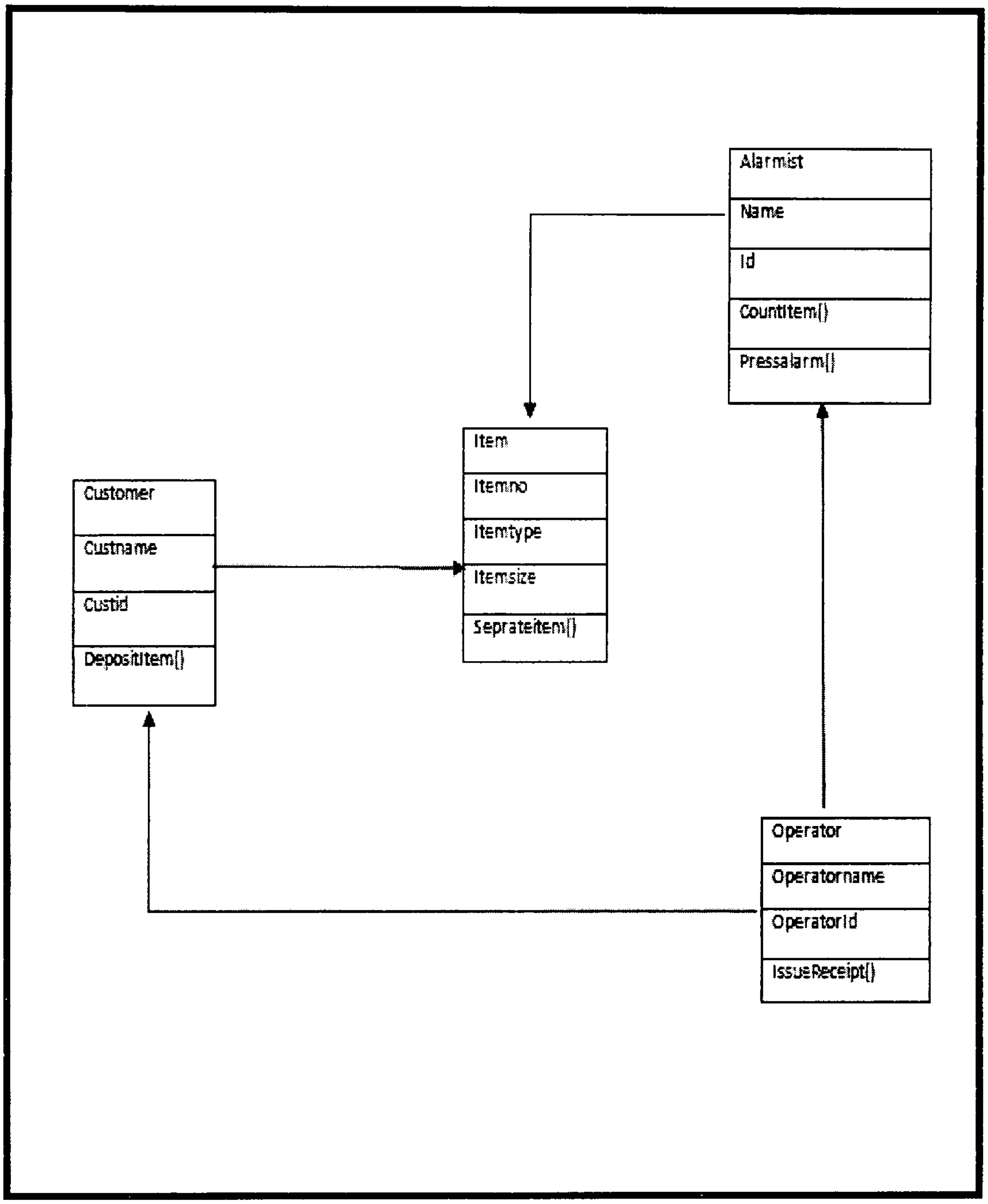


Figure 3.16: Implementation Model of Recycling Machine System

3.2.3 Ontology Development Life Cycle (ODLC)

Ontology Development Life Cycle (ODLC) refers to the activities that have to be performed when building ontologies. These have been classified into three categories such as Ontology Management, Ontology Development and Ontology Support Activities as shown in Figure 3.17. Ontology Management activities include scheduling, control and quality assurance. Scheduling activity identifies the tasks to be performed, arrangement, time and resources needed for completion. However, control activity identifies scheduled tasks to be performed in anticipated slot. Finally, quality assurance activity assures that the quality of each and every product or output is satisfactory.

Ontology Development activities has been grouped into pre-development, development and post-development activities. During pre-development, an environment study identifies the problem to be solved. In development, specification, conceptualization, formalization and implementation activities have been constituted. Firstly, specification activity states intended uses and end users of ontology. Then, conceptualization activity structures domain knowledge and formalization activity transforms conceptual model into formai or semi computable model. However, implementation activity builds computable model in an ontology language. Lastly, in post-development, maintenance activity updates and corrects ontology if needed and evaluation activity manages ontology changes.

Ontology Support activities composed of knowledge acquisition, evaluation, integration, alignment, documentation and configuration management. The goal of knowledge acquisition activity is to acquire knowledge from experts in a given domain. Evaluation activity makes technical judgment of associated environments. An integration activity is required, when building new ontology by reusing other ontologies already available. Whereas, merging activity unify concepts, terminology, definitions, constraints etc., from source ontologies. Documentation activity provides details of completed stages and configuration management activity records all the versions of documentation [UG96, FGJ97].

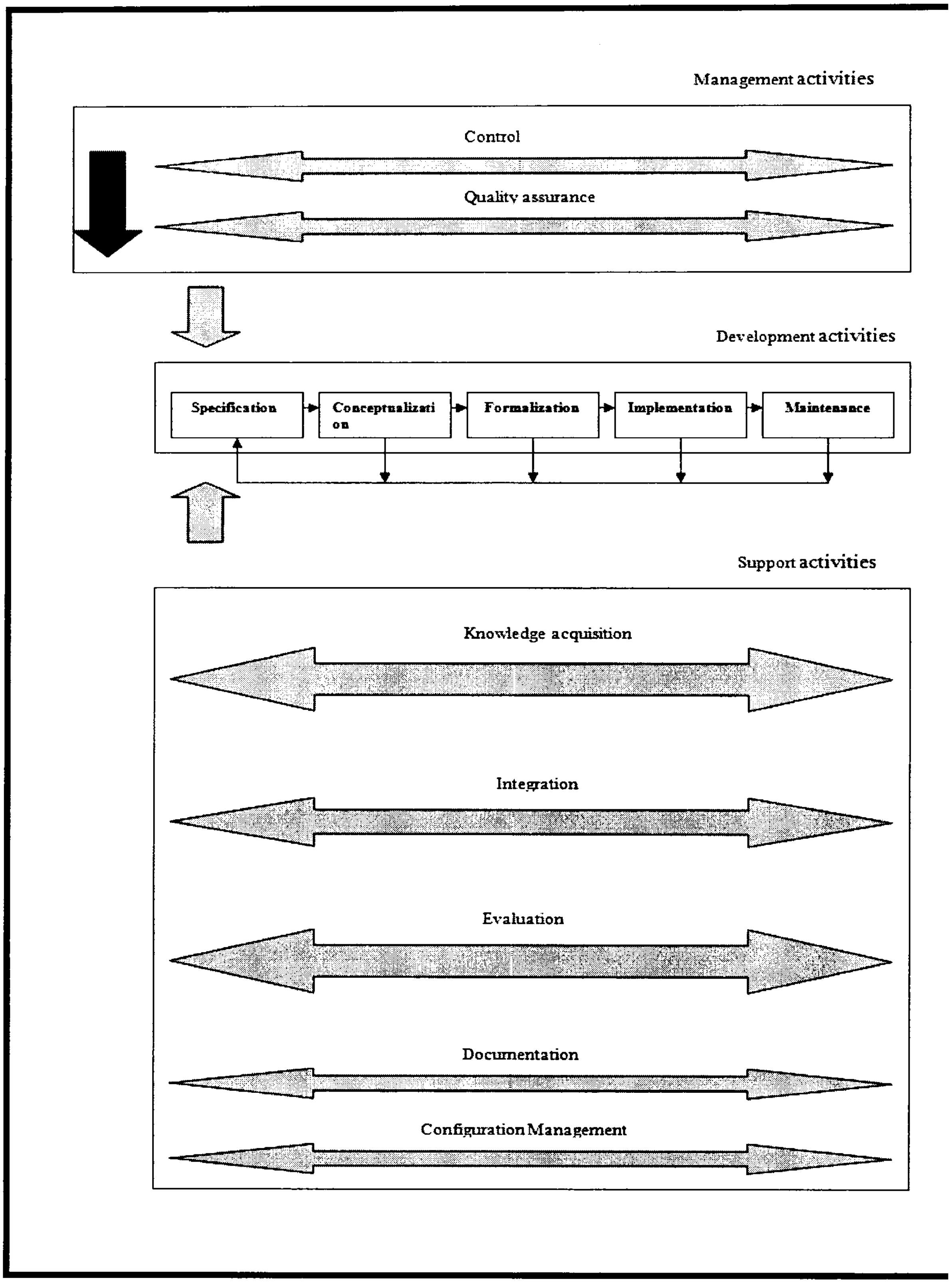


Figure 3.17: Ontology Development Life Cycle

3.3 Mapping OOSDLC and ODLC in Information Systems

It has been studied that software quality and productivity can be improved by the use OOSDLC. It can automate several tasks of the software development process and making easier to control it. But, OOSDLC enables developers to build domain specific systems only. To build a generalized system, explicit conceptualization of the domain is essential. Ontologies involve the specification of concepts and relations that exist in the domain, definitions, properties and constraints. Consequently, we have attempted phase wise mapping of OOLC and OOSDLC with the help of common elements. Mapping of OOSDLC and OOLC curtails the constraints of domain specific systems during the development that resulted into a generalized development environment.

3.3.1 Ontology Driven Information System (*ODIS*)

Ontology Driven Information System can be developed using phased development life cycle. It comprises of *Ontolysis, Ontodesign* and *Ontocontation* as shown in Figure 3.18. Firstly, *Ontolysis* concerns with the purpose identification and requirements specification to clearly identify domain conceptualization. A model using a graphical language, with a dictionary of terms is used to facilitate the communication with domain experts. Subsequently *Ontodesign* consists of ontology formalization, integration of existing ontologies and ontology evaluation. Ontology formalization aims to explicitly represent the conceptualization captured in a formal language. Then, to seize established conceptualizations, it is essential to integrate the current ontology with existing ones. On the other hand, ontology evaluation checks for the accomplishment of requirements specification. Lastly, the ontology development has been documented, including purposes and adopted design criteria in *Ontocontation* Thus, one key to promote the advantages of ontologies is in a generality perspective across OOSDLC.

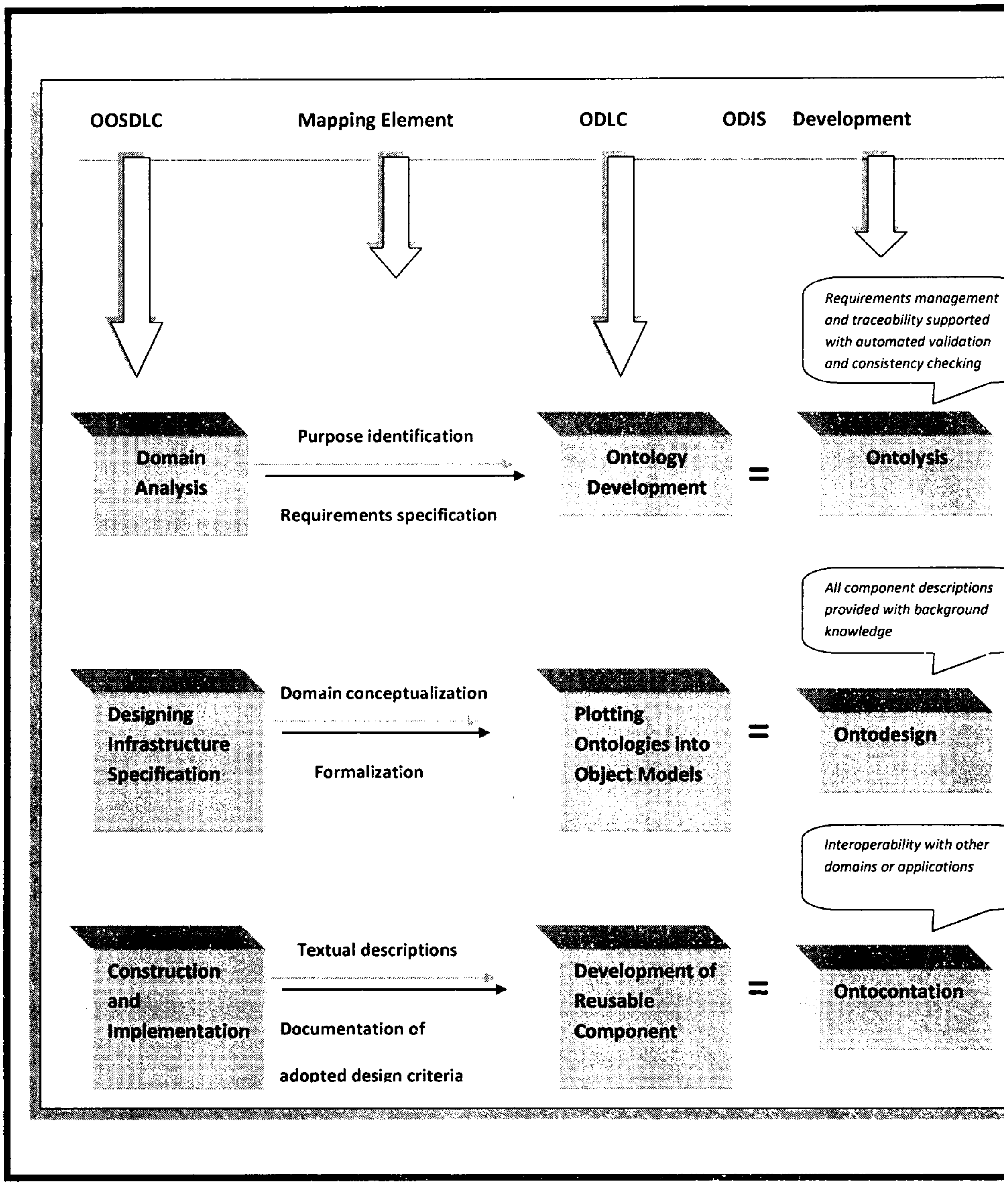

OOSDLC
Mapping Element
ODLC
ODIS Development
Requirements management and traceability supported with automated validation and consistency checking
Domain Analysis
Purpose identification
Requirements specification
Ontology Development
=
Ontolysis
All component descriptions provided with background knowledge
Designing Infrastructure Specification
Domain conceptualization
Formalization
Plotting Ontologies into Object Models
=
Ontodesign
Interoperability with other domains or applications
Construction and Implementation
Textual descriptions
Documentation of adopted design criteria
Development of Reusable Component
=
Ontocontation

Figure 3.18: Ontology Driven Information System (*ODIS*) Development

3.3.2 *ODIS* Development

In *ODIS* development life cycle, *Ontolysis* implies an investigation of existing ontologies through the approach of developing a Generalized Use Case Model for a system as shown in Figure 3.19. There can be three actors possible namely *User, Operator* and *Administrator. User* points to information regarding the end users and *Operator* represents the information about the workers. However, *Administrator* signifies the information concerning overall controller for a system. Use cases represent generalized functionalities associated with a system such as *System access, Concerned information checking* and *System transaction specification. System access specification* keeps the information regarding all the security rights on the hierarchical level. Next, *Concerned information checking specification* provides all the related supervisory information and *System transaction specification* confers all the possible transactions that can be carried out for a system. In addition, transfer of codified knowledge for the application domain has been established with Generalized Object Model. Figure 3.20 depicts Generalized Object Model that constitutes objects, logical attributes, static instance associations (explicitly the static references between these objects) and possible operations to manipulate the objects.

Next, *Ontodesign* builds on ontological definitions and commitments for the application domain with Generalized Design Model. Three types of objects namely Interface objects, Entity objects and Control objects has been encapsulated. Also, concept of block as design objects defines the structure and hierarchy, interfaces, rules for commitment as shown in Figure 3.21.

Lastly, *Ontocontation* involves feedback of the application results and experiences to the ontology developers/maintainers and have to be incorporated there in order to keep the ontology a proactive component. It is engrossed with defining the classes in the ontology then arranging the classes in a taxonomic hierarchy, followed by defining slots and filling in the values for slots for instances.

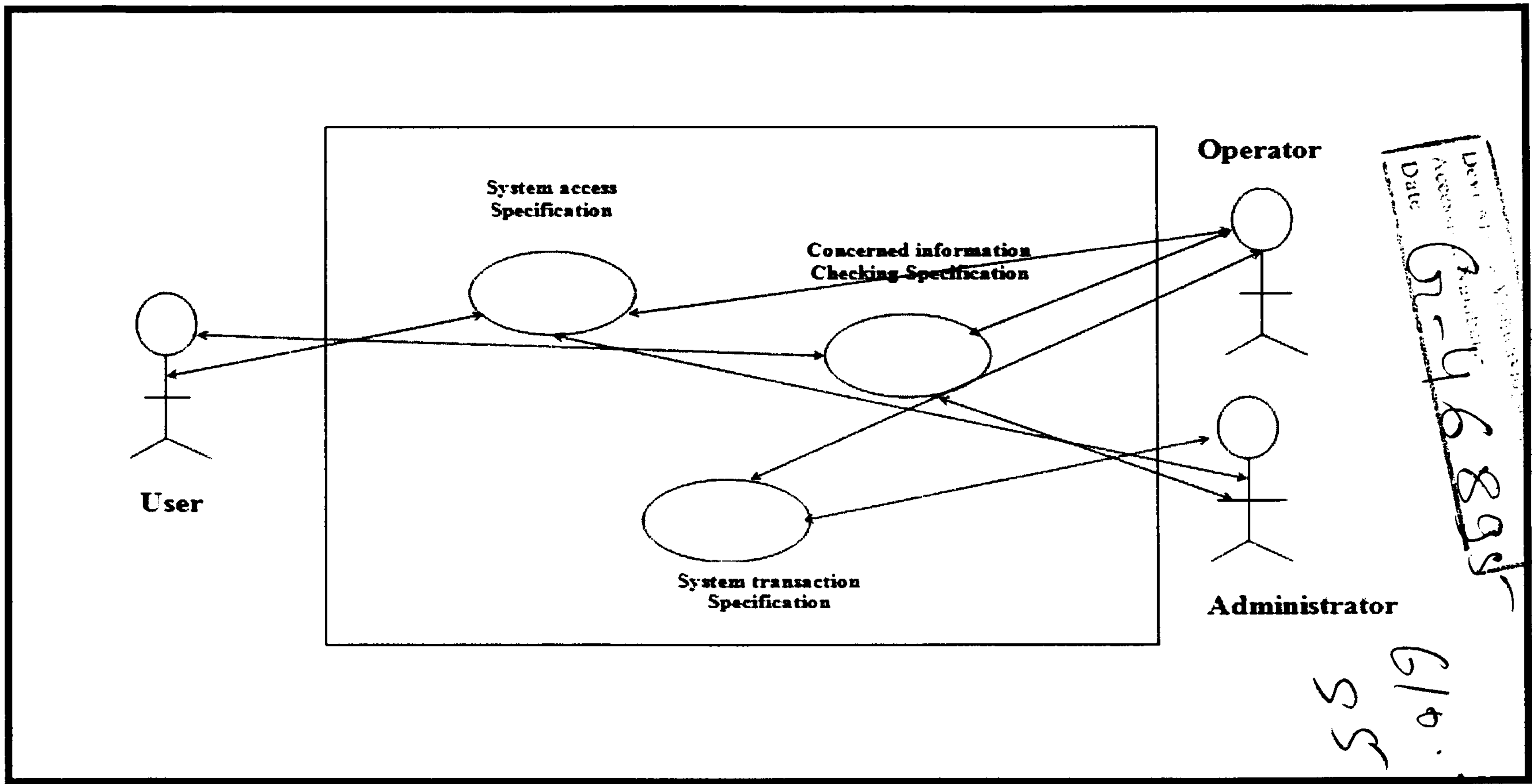


Figure 3.19: Generalized Use case Model of *ODIS*

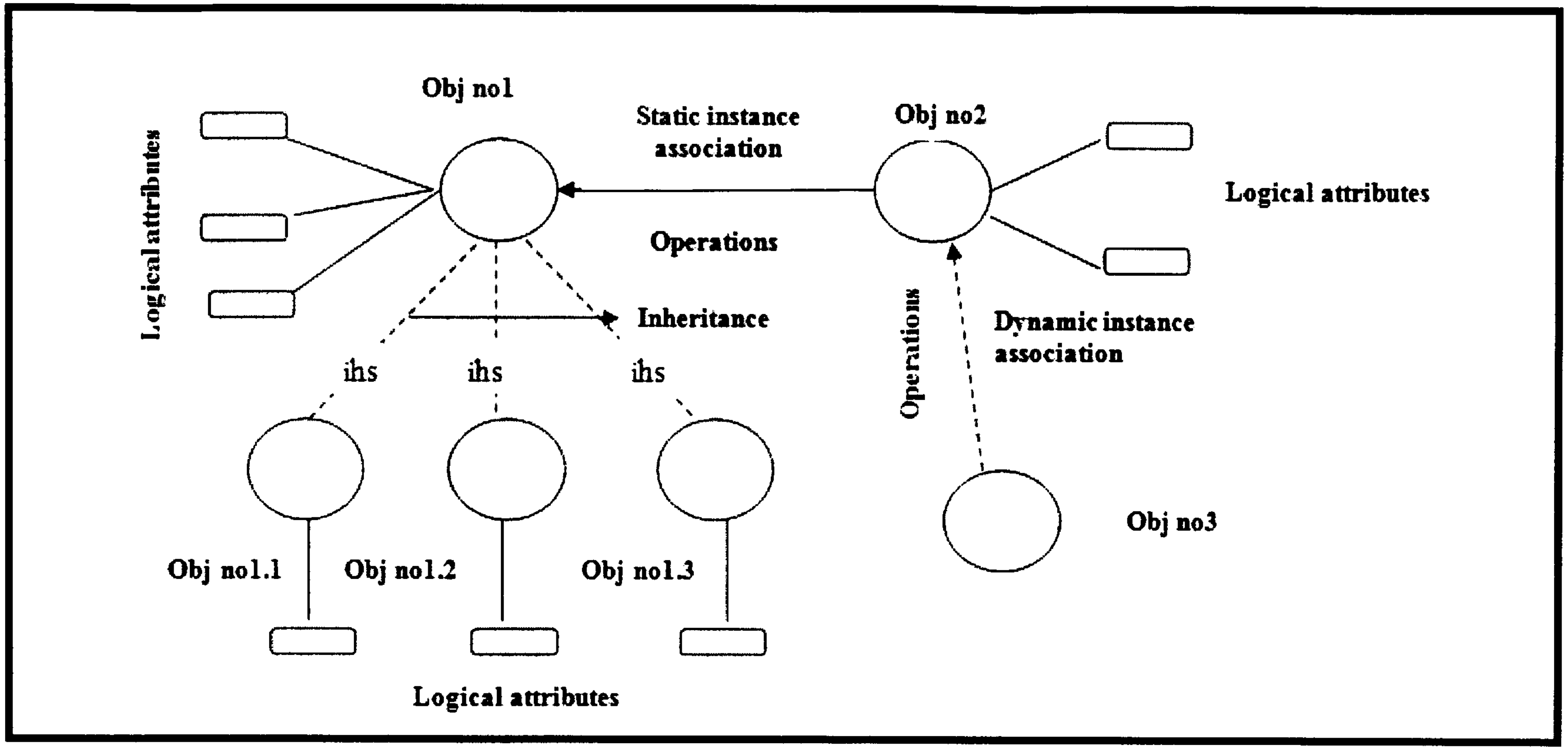


Figure 3.20: Generalized Object Model of *ODIS*

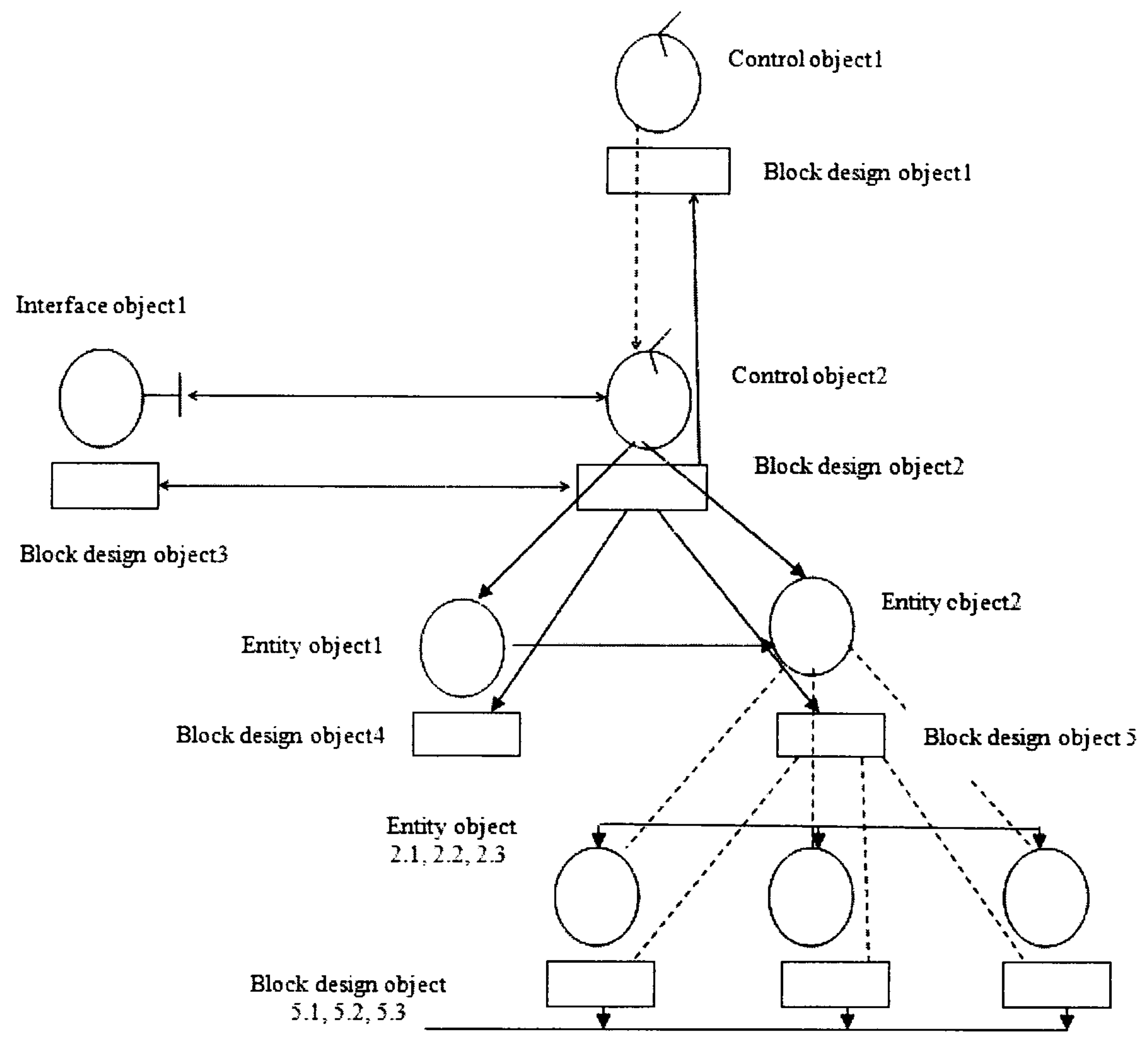


Figure 3.21: Generalized Design Model of *ODIS*

3.4 Results

It has been observed that an Information System can be build using ontological approach. Since, ontology profitably drives all aspects of an Information System thereby it can be Ontology Driven Information System (*ODIS*). Also, it enables the developer to reuse and share application domain knowledge using a common vocabulary across heterogeneous software applications. Let us take Hospital Management System using *ODIS* development life cycle. While developing Hospital Management System using OOSDLC, use cases and actors developed can be derived from Generalized Use Case Model such as *System access specification* use case signifies Login, Admission and Add, Delete or Edit Doctor or Staff. Next, *Concerned information checking specification* includes Patient Information, Ward wise Bed Status, Bed Allotment and Other Privileges. Lastly, *System transaction specification* comprised of Fix Doctor or Test Appointment, Prescribe Tests, Admission or Discharge Reports and Draw Salary.

On the other hand, Staff or Nurse, Doctors and patients constitutes *User* actor whereas Receptionist, Information keeper and Daily Record keeper indicates in *Operator* actor. In addition, Finance manager works as an *Administrator* actor. Now, objects built in Object Model of OOSDLC can be obtained from Generalized Object Model. Employee, Patient, Expenditure, Income Receipt, Daily Record keeper, Information keeper and Finance Manager confers *Objects*. However, Update indicates the *Operation* and Salary points to *Static instance association*. Moreover, Receives specifies *Dynamic instance association*. In the same way, Generalized Design Model and *Ontocontation* instigate Implementation Model of OOSDLC respectively.

Thus, we can develop aforesaid kinds of generalized models for each and every information system.

3.5 Summary

In this chapter, we have attempted to develop ontology for different Information Systems to ensure the generality. In this view, we have presented the Object Oriented concepts with related literature survey. We have mainly focus on development phases of OOSDLC and ODLC in details. In addition, we have studied various types of Information Systems that have been implemented in industry. We have attempted to develop an illustration of TPS, MIS and ES using OOSDLC and followed by ODLC. In next section, we have highlighted the mapping of OOSDLC phases and with various phases of ODLC in Information Systems. This mapping divulged *ODIS*. Our investigations reveled that all the phases of OOSDLC can be mapped with phases of ODLC to form corresponding generalized models. It is interesting to note that by using these generalized models lessen the system development efforts at each phase. It is useful in terms of knowledge transfer from project to project in a certain application domain and from one development cycle of a project to the next. At last, we have highlighted the results of mapping of phases of OODSDLC with the phases of ODLC.

CHAPTER 4

Ontology Aided RE and Process Models

4.1 Introduction

Requirements Engineering (RE) phase of a software project is vital to its successful completion. Consequently, the ability to identify problems and suggestions for improvements in RE process opens up significant potential for increasing the success of software projects. RE process is often depicted with the help of Linear Sequential, Linear Iterative, Iterative and Spiral Requirements Engineering Process (REP) models. These conventional REP models have been successful for the confining of knowledge strenuous environment. However, to deal with increasing competencies of software project, the software systems demand knowledge intensive RE process. Also, RE process has been required to promote the cohesiveness among the information gathered and to provide a coherent view between the stakeholders. Therefore, we have encapsulated the ontology for invention of generalized requirement set that may cater various applications from different domains.

By adopting ontology, requirements knowledge has been represented as ontology concepts and therefore become more definite, complete, consistent and convenient to share and reuse [ZZY+07, JIN00, KS06]. Based on the above decree, it appears potential to consider and evaluate elementary shift in the way RE has been practiced. To accomplish this shift, a possible orientation towards knowledge-driven RE by identifying shortcomings of current process-driven RE approaches has been considered. Furthermore, we have emphasized on categorization of requirements depending on relevant changes of knowledge to bring about knowledge-driven RE. Since, it has been generally accepted that domain knowledge play a very important role in RE after long-term of requirements practice. Besides, guided by the domain knowledge and requirement type, domain users can present the requirements more effectively, and requirements analysts can understand the requirements more accurately and construct correct requirements model. Accordingly, we suggest Ontology Aided Requirements Engineering *(OntoAidedRE)* model in this chapter that reconcile and unify domain-specific concepts, approaches and knowledge. *OntoAidedRE* provides an unambiguous and precise terminology that can jointly explicable and functional across various realms. It formally expresses primitive requirements through top-down refinement of generalized requirements, so that it becomes easier to detect and handle incompleteness and inconsistency of any domain requirements.

4.2 Background

With the advances in software development, the role of RE is to establish engineering principles amenable to analysis, design and implementation. Traditional requirements engineering is an iterative process and continues iteratively until the project is complete. This process involves the activities such as defining the terms in the domain of discourse, stating, clarifying and agreeing on assumptions and constraints, discussing and negotiating the needs and the objectives for a software development [VAN01, VAN03]. There exist conventional REP models to define these activities. In addition, these models are conflicting in nature, ranging from linear and incremental, recurring and iterative in structure [HKW+, LPR93].

Ontology refers to the basic existential pool of knowledge in the world of interest to the discipline [NM00, GRU92, EW05]. In order to elicit system requirements correctly and unambiguously, researchers in RE community have been studying and developing a number of ontology based approaches. Many of them adopt ontology to describe static knowledge for all domains [JMF08]. All these approaches suggest that the ontology is necessary to express domain knowledge for the sake of knowledge based elicitation and reuse. But, variety of the knowledge existing in application domain makes it difficult for generalized and reusable requirement elicitation. To resolve this problem ontology should not act merely as a static knowledge base. But, ontology must be used as a substantial assurance in aiding the elicitation and elaboration of requirements. By introducing dynamic knowledge such as requirement type, requirement elicitation would become an inducible process and more knowledge be reused. In this view, explicit treatment of knowledge for emphasizing on the category of requirement in RE practices suggests a fundamental shift in the domain oriented underpinnings of RE process [BL03, SG05, SM98].

4.2.1 Requirement Engineering (RE)

Requirement Engineering (RE) is the branch of software engineering concerned with the real world goals for functions and constraints on the software systems [VAN08b]. Requirements are often specified, validated and documented across different domains, disciplines and dislocation of the respective stakeholders and authors. Also, requirements reuse and designing of solutions obtained in a hysterical way is due to the lack of knowledge intensive environment. Besides, members of the same domain may use different terminology, and often there exists no common perceptive of the terms or concepts used and consequently problems have been appeared [KOG+08]. Therefore, many requirement engineers have been facing numerous challenges when developing software requirement specifications for highly complex, long-lead projects and services of various domains [VL00, LV02].

Currently, requirements engineers must understand the different types and levels of requirements for high-quality RE. It requires an interdisciplinary approach that considers the needs of multiple stakeholder groups. It also requires expertise in various RE activities including requirements elicitation, requirements analysis and negotiation, requirements documentation and validation, requirements management and configuration management [MMJ+04].

Requirements Elicitation

The goal of requirement elicitation is to gather raw requirements. It involves technical staff working with customers to find out about the application domain, the services that the system should provide and the system's operational constraints.

Requirements Analysis and Negotiation

Requirements analysis and negotiation is an activity that aims to discover problems and conflicts with the requirements and reach agreement on changes to satisfy all system stakeholders or people that are affected by the proposed system. The final purpose is to reach a common understanding of the requirements between all project participants.

Requirements Documentation and Validation

During this activity, the defined requirements have been written down in a software requirements specification document and validated against criteria of correctness, completeness, consistency, verifiability, un-ambiguity and traceability etc.

Requirements Management

Requirements management consists of managing changes of requirements specifically keeping requirements consistent. It is achieved by ensuring identification of interdependencies between requirements, other requirements and artifacts.

Configuration Management

Configuration management is a process of identifying and defining components in a system. It controls the release and change throughout the lifecycle, recording and reporting the status of components and change requests. In addition, it deals with verification of completeness and correctness of systems components. Requirements management can be seen as integrated part of change and configuration management.

4.2.2 Parameters of Study

The project attributes such as project type, project size, project team, project effort, project quality, project prioritized element and project key element play a very significant role during project development. Managing these project attributes will add a new dimension to requirement engineering process thereby contributing to project success [SPL06]. Consequently, we define these project attributes as follows:

Project Type

It defines the statement of work that must be completed during the development and includes building the product prototype [SPL06]. Project type illustrates the classification of projects according to acquired distinctiveness such as operations support, management support and others. Operation support systems include transaction processing, process control and office automation systems. However, management support system constitute management information and decision support system whereas others including expert systems as described in Chapter 3.

Project Size

Project size determines the scalability of methodology. There exist three most important factors such as estimated effort, experience level of project manager and complexity. These are used to categorize project size into small, medium and large. Project size in terms of effort hours is defined as 1 to 250 effort hours fall in small project size. Subsequently, project having 251 to 5000 effort hours signifies medium project size and projects over 5000 effort hours implies large project size. Then, experience level of the project manager suggests that an experienced manager manages larger projects with at least up to a higher effort threshold. Consequently, project size becomes medium or small. On the other hand, an inexperienced project manager manages a 2000 hours project in a way of large project size. Moreover, complexity of a project indicates large project size for 1000 hours project that is extremely critical to the business whereas 5000 hour project becomes small as two or more similar projects managed before by the project manager [LINK2].

Project Team

Project team is a group of individuals with appropriate and complementary professional, technical or specialist skills that aims to provide technical expertise in support of project objectives. It also contributes to understand the use project management standards specified in a project and maintain the project documentation in line with the project quality plan. It can consist of human resources within one functional organization, or it can consist of members from many different functional organizations such as project analyst, Change Control Board member, client project manager, project designer, project manager or team leader [LINK3].

Project Quality

Project quality is the totality of features and characteristics of a product or service that bear on its ability to satisfy stated or implied requirements. There exist several indicators used as a range of measuring the project quality, or the perception of quality [STA08]. Table 4.1 shows Engagement Measures (EM) that specifies internal customer involvement in key project activities. Next, Planned vs. Actual Cumulative Review Count (P vs. A CRC) illustrates ratio of expected to actual project activity count. Lastly, Assessment Measures (AM) points to customer satisfaction surveys and stakeholder expectations evaluation [STA08]. For example, in online examination system there exist great involvement of faculties i.e. customer in all related activities such as setup examination, subject, code and criteria etc. Therefore, the value of EM becomes high. Since, ratio of expected to actual project activity count calculated as 0.5, P vs. A CRC value turns into low. However, enormous use of online examination system provides high value of AM. Then, according to Table 4.1 quality status of online examination system comes in range of 65 to74%.

Table 4.1: Project Quality Indicators

EM	Pvs.A CRC	AM	Quality Status
High	High	High	~100%
High	High	Med	75-99%
High	High	Low	65-74%
High	Med	High	75-99%
High	Med	Med	51-64%
High	Med	Low	41-49%
High	Low	High	65-74%
High	Low	Med	41-49%
High	Low	Low	31-39%
Med	High	High	75-99%
Med	High	Med	51-64%
Med	High	Low	41-49%
Med	Med	High	51-64%
Med	Med	Med	~50%
Med	Med	Low	~40%
Med	Low	High	41-49%
Med	Low	Med	~40%
Med	Low	Low	26-30%
Low	High	High	65-74%
Low	High	Med	41-49%
Low	High	Low	31-39%
Low	Med	High	41-49%
Low	Med	Med	~40%
Low	Med	Low	26-30%
Low	Low	High	31-39%
Low	Low	Med	26-30%
Low	Low	Low	0-25%

Project Prioritized Element

Project prioritized element measures the project characteristics on three aspects viz. cost effectiveness, time effectiveness and project functionality. It begins with the project cost element comprised of acquisition cost and the operation cost. Then, it describes the project time element which includes overall time required to complete the project. Finally, project functionality is characterized by usability, serviceability and compatibility as well as the modes of operation. Different information systems emphasize on different priority elements according to the types such as TPS, OAS, PCS, DSS and ES. For example, while automating hotel management system, cost and functionality grows to be priority elements than time element. However, for power utility system time and functionality elements dominate than cost.

Project Key Element

Project key element is a well-defined knowledge artifact for every project that expresses the value of product and its competitive benefit in terms of generality. It provides the guidelines to project team for knowledge driven RE process. Generally, conventional REP models used to develop requirement specification set for domain specific systems such as TPS, OAS, PCS, DSS and ES. Eventually, these models undersupplied for knowledge driven RE process.

4.2.3 Conventional REP Models

Requirement engineering process is often depicted with a linear, incremental, cyclical and iterative model. Within these models, common RE activities such as elicitation, analysis and negotiation to documentation and validation have been combined under different labels, which follow varied organization formats [HKW+04]. We have defined these activities for different models as follows:

Linear REP Model

Linear REP model constitutes five activities namely; concept, problem analysis, feasibility and choice of options, analysis and modelling and requirement documentation as shown in Figure 4.1. Concept triggers RE process and defined for an improvement or enhancement of the product. In many cases, it triggers the organizational level RE process. Since, the concept moves into a specific project it triggers RE process at project level or development process level. During problem analysis phase, an understanding of the nature of problem is developed. It helps requirements engineers to identify set of alternative solutions by generating an appropriate representation of the problem. However, feasibility and choice of options phase is concerned with evaluating the costs and benefits of alternative solutions and negotiations. Subsequently, detailed analysis and modelling phase deals with a more detailed analysis of the requirements. Once this process is finished, requirements specification document can be completed. In addition, validation process takes place at the end of each phase in this model. Being a simple model, it is mostly used for small projects with some less amount of complexity. But, it is observed that it is not appropriate for large projects to get requirements.

Linear Iterative REP Model

Linear iterative REP model starts with generation of initial requirements statements at requirement elicitation phase as illustrated in Figure 4.2. These statements based on user needs, domain information and existing system information analysis. Requirements analysis phase finds problems in the initial requirements statements generated in requirements elicitation phase with the help of completeness checking. In completeness checking the incomplete requirements are pinpointed. The agreement phase is the process of discussing the issues and problems pointed out in the requirements analysis phase and finding some agreement with which all of the stakeholders can live. Eventually, solutions are identified and issues are resolved to the satisfaction of the parties involved. In many cases, it is possible that information available for agreement is sufficient or

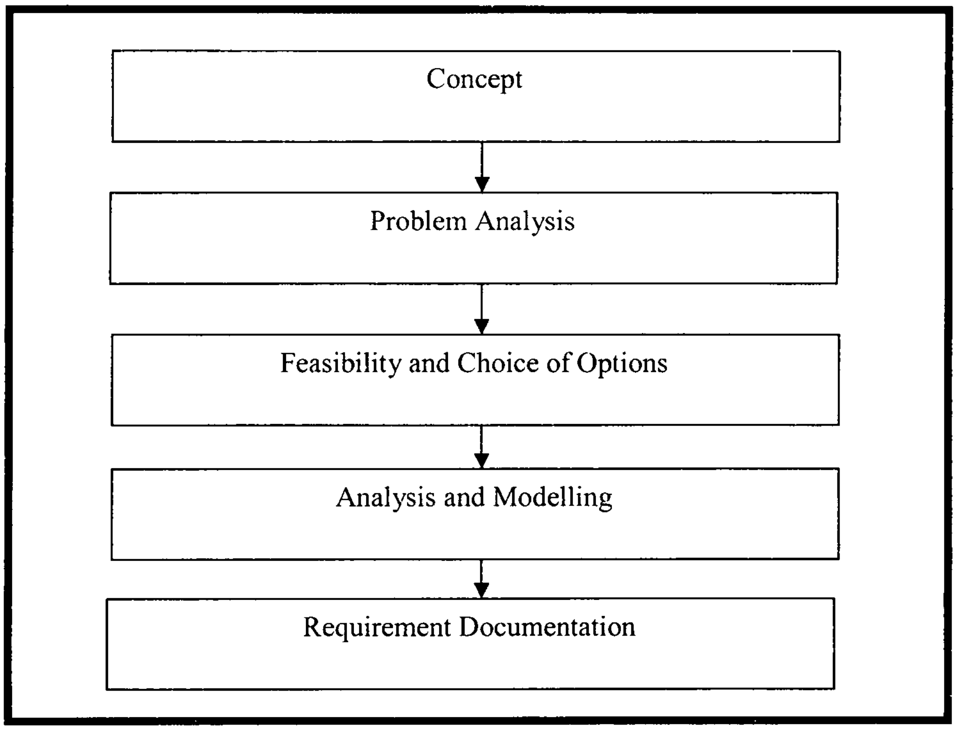


Figure 4.1: Linear REP Model

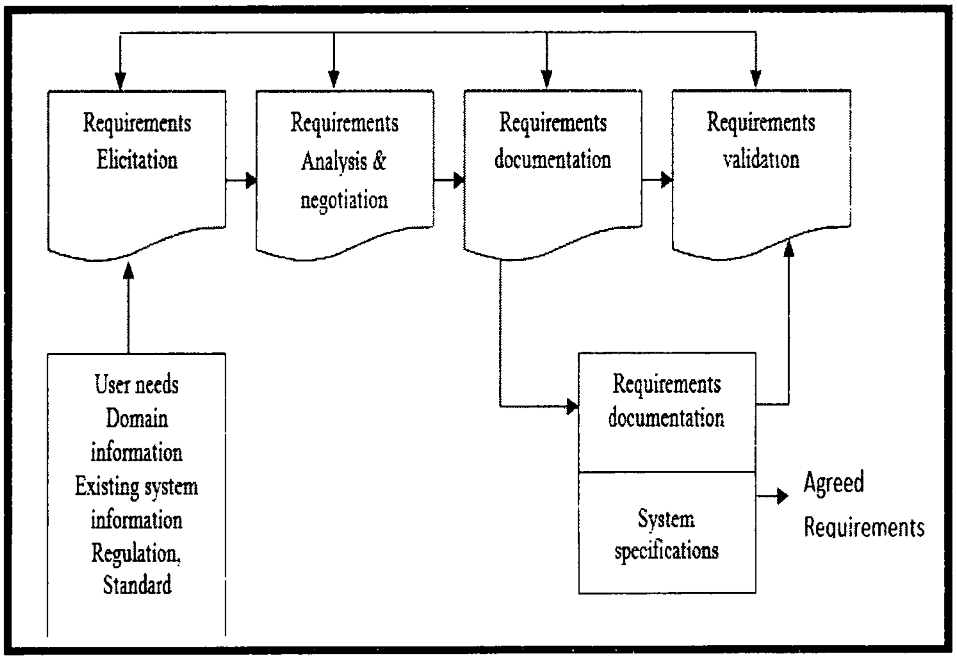

Requirements Elicitation
Requirements Analysis & negotiation
Requirements documentation
Requirements validation
User needs Domain information Existing system information Regulation, Standard
Requirements documentation
System specifications
Agreed Requirements

Figure 4.2: Linear Iterative REP Model

some new requirements emerge. In such cases, the unresolved issues or new requirements are forwarded again to next iteration. This iteration continues until the stakeholders are agreed and the final system specification is achieved. At the end of this phase, final requirements statements generated and forwarded to the validation phase for validation and discussion. During validation phase validation checklists are discussed, and agreed actions are performed. Consequently, this model is useful for the system where the specifications should be pin point accurate and validated multiple numbers of times through the potential stakeholders.

Iterative REP Model

Iterative REP Model is used to perform RE in multiple iterations and hence is better for software that is launched versions by versions. It is comprised of three phases namely; elicitation, specification and validations as shown in Figure 4.3. In this model, elicitation is considered as an ongoing process. It provides the knowledge to other processes such as specification and validation. The purpose of requirements elicitation is to gain of knowledge relevant to the problem that can be used to produce formal specifications. Requirements specification is the central process that controls both the elicitation and validation processes. During specification it becomes apparent that more information about the problem is required. This triggers the process of elicitation which in turn supplies the needed information. Conversely, some change in the problem domain (e.g. change in some assumption, made about the domain) triggers a change in the specifications. Accordingly, elicitation can take place during the specification process. Similar interactions appear between specification and validation such as completion of some part of the specifications can cause the need for validation.

Requirements validation is defined as the process which certifies that the requirements are consistent with customers' and users' intents. It is ever present in all stages of RE. The need for validation is triggered by the acquisition of new knowledge about the problem domain (elicitation), or by formulation of requirements (specification).

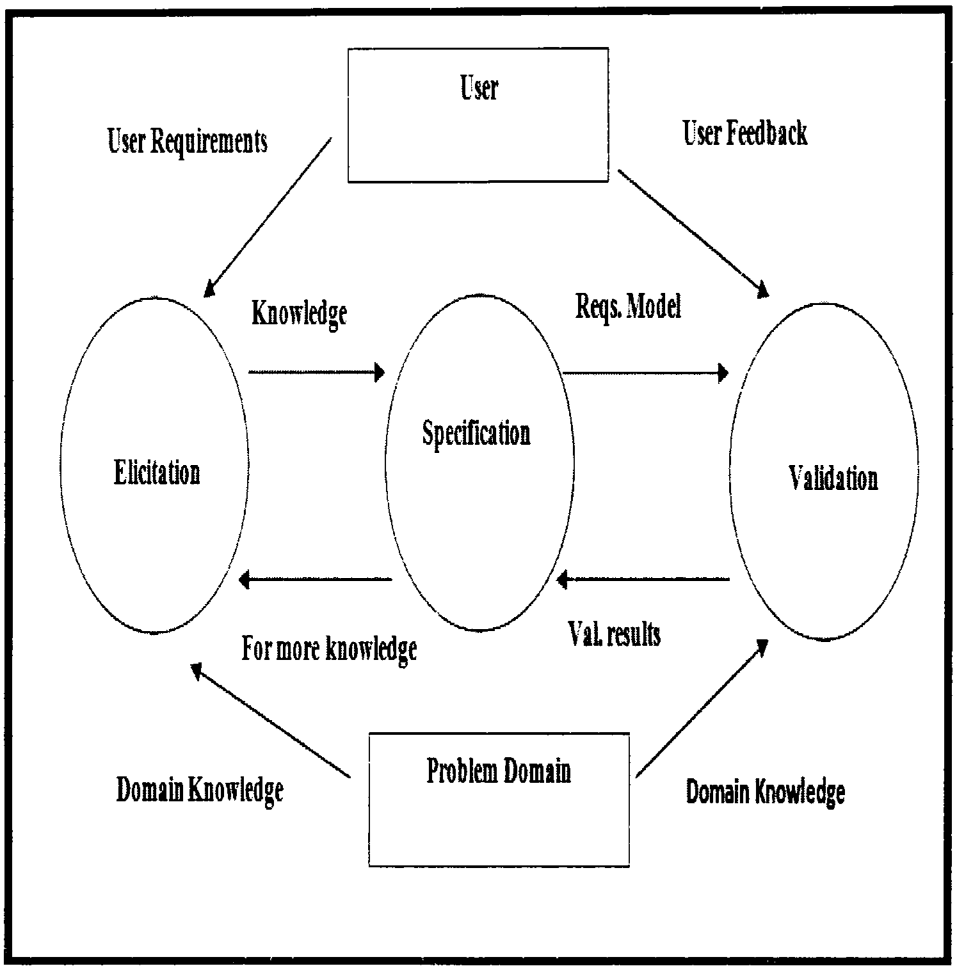


Figure 4.3: Iterative REP Model

Spiral REP Model

In spiral REP model, all different RE activities repeats until a decision is adopted about acceptability of requirement documents as illustrated in Figure 4.4. It is performed in spirals. One spiral represents the complete version of the requirements on the basis of which the system has to be developed. Each spiral is divided into four quadrants such as specification extraction, discussing and analyzing requirements, requirements documentations and requirements validations. It starts with requirement extraction phase that collects information in requirement development process. In this phase requirements are identified by consulting with customers, developers and users. While, in discussing and analyzing requirements phase requirements are studied in regard of necessity, compatibility, completeness and possibility. During this phase requirements are analyzed and modeled and possible interference of requirements are omitted by prioritizing discussions and risks of the issue are identified. Output of this level is a complete compatible and prioritized set of requirements. The final goal of RE is to document requirements to be met and purpose of requirements documentation is making relation between requirements understood by users and developers. Hence, requirements document describes application extent as well as under development system.

Requirements document can be considered as a base for controlling changes and evaluating future products. Validation in requirement engineering is done for controlling the quality. Requirements validation means confirming that requirements are complete and well- written and supply needs of customer. This phase may continue repeating other requirements development phases because of identified deficiencies, gap between requirements, additional information and other issues. Implemented software product is validated in software life cycle test phase on the basis of its requirements. The main characteristic of this model is to handle the unwanted consequences such as speciation delay and requirements change etc. which can badly affect the cost schedule and quality of the project.

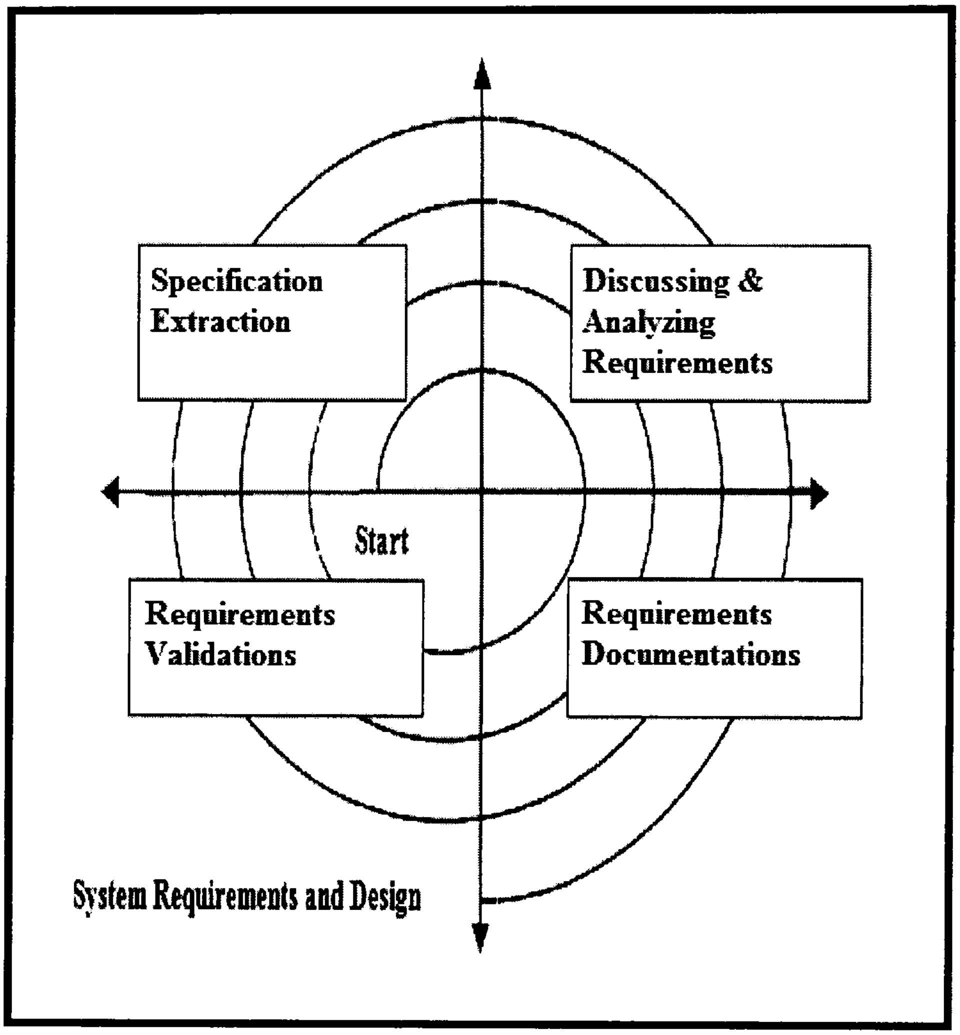

Specification Extraction
Discussing & Analyzing Requirements
Start
Requirements Validations
Requirements Documentations
System Requirements and Design

Figure 4.4: Spiral REP Model

4.3 Ontology Aided REP Model (*OntoAidedRE*)

We have proposed *OntoAidedRE* to transform conventional process driven requirement engineering. It follows an encrusted approach and intended to be a generic paradigm that enables knowledge driven requirements engineering. Ontology is used to strengthen the generality of concepts that depend on types of requirements for different application domains such as TPS, OAS, PCS, DSS and ES. The product of applying *OntoAidedRE* promotes cohesiveness depending on relevant changes of knowledge. Consequently, a layered structure has been designed taking into account the inter relationships between different domain requirement types as shown in Figure 4.5. We have devised four layers namely; *OntoPre Requirements, OntoInput Requirements, OntoSystem Requirements* and *OntoOutput Requirements*. Moreover, *OntoSystem Requirements* comprised of three sub layers such as *OntoSystem Operational Requirements, OntoSystem Control Requirements* and *OntoSystem Parameter Requirements.* The comprehensive depiction of these layers is as follows:

OntoPre Requirements

While developing requirements, it has been observed that every system has statements of fact and assumptions. It defines the expectations of the system in terms of tasks, objectives, constraints, and measures of effectiveness and suitability. Hence, this layer is primarily responsible to facilitate user verification or identification that describes task objective accomplishment by the system. Consequently, *OntoPre Requirements* for the candidate system defines system definition method, system activation method and system defined constraints.

For example, while developing requirements set of TPS, OAS, PCS or DSS, the first objective is to get into it for further use. Accordingly, we have included Login-Password or security check services and client registration facility into *OntoPre Requirements*. Through these services or facilities, clients get int0o the system and understand its definition in terms of various functions performed by it. Besides, it signifies role of system clients and system constraints imposed on client type such as system administrator has different role than the system user.

OntoInput Requirements

OntoInput Requirements have been designed to provide in the system with all initial data required to initiate the system process. It identifies necessary attributes, characteristics, or excellence of a system to have value and utility to a user. Accordingly, this layer triggers the initial qualifying terms for the candidate system to acquire first insights into its usability. Further, it evolves to conceptualize the problem domain and system-defined or access details such as stacking of initial data values.

It has been observed that during any OAS development, requirements for office details, customer records, employee records etc. provides initial inputs or the system required specifications. Similarly, in PCS requirements for process source details and in DSS existing standards defined for decision making constitutes *OntoInput Requirements.*

OntoSystem Requirements

OntoSystem Requirements define a role of a software system or its component. A role is described as a set of actions, behavior, and features. It explains the type of system job by identifying the necessary action or activity that must be accomplished. Accordingly, these requirements capture ideas, perspectives and relationships at various levels of detail such as operational, parameter and control level. Consequently, the layer is divided into three sub layers to create a chain of commands consisting of various services or concerns associated with system and the environment.

OntoSystem Operational Requirements

OntoSystem Operational Requirements depicts operational life cycle specifically operational distribution or deployment. There exist requirements for calculations, technical details, data manipulation and processing and other specific functionality that a system has to accomplish. Hence, this sub layer covers all the system access procedures such as system modification and updating competencies which includes creation, addition, deletion, alteration etc.

For example in TPS requirements to add, delete or update any transaction whereas in OAS requirements to add, delete or modify the customer, employee or office asset details strives for *OntoSystem Operational Requirements*. In addition, *OntoSystem Operational Requirements* for DSS includes requirements to add or update the present status and compare it with existing standards for decision making at that instance.

OntoSystem Control Requirements

OntoSystem Control Requirements have been designed to ensure that system procedures must perform with integrity. It has been observed that control procedures deals with the integrity of internal process information and the accuracy provided to system output. Establishing effective control procedures early in software system helps to create an ethnicity of fair software system management. As a result, this sub layer is responsible for system control provision. Also, these control procedures have been catalyzed by pensiveness of knowledge engineering environment.

It has been observed that, while developing PCS such as Power Utility System, requirements for the power control must be engrossed to drive the system successfully. Therefore, *OntoSystem Control Requirements* have been devised to define total power units and to shut down the supply in case of sudden failure. Similarly, while developing TPS such as Online Examination System, requirements for setting up criteria for examination are included in *OntoSystem Control Requirements.*

OntoSystem Parameter Requirements

Presently, requirements have been interactively developed across all identified functions based on system life cycle factors. Also, these have been characterized in terms of the degree of certainty, degree of criticality to system success, and their relationship to other requirements. Thus, *OntoSystem Parameter Requirements* have been designed to facilitate the critical system parameters to accomplish the task. Generally, it is measured in terms of coverage and suitability or inclination. Hence, the entire system parameterizing procedures are ascertained at this sub layer. It allows confining the system decision information as intelligent records such as conventional databases and data dictionaries.

For example, in Online Examination System, *OntoSystem Parameter Requirements* have been designed to facilitate requirements for setting up students for examination. In addition, it aids in setting up paper such as question selection as per subject, to inform in case of scarcity of question in data dictionaries and to confirm about paper has set. Then, during Hotel Management System, OntoSystem *Parameter Requirements* includes requirements for membership provision such as to validate the customers for membership in case meets the defined criteria.

OntoOutput Requirements

Finally, requirements for system eventual presentation such as to view the system output in the form of reports, transaction receipt, bills or invoices etc. have been indicated at this layer. Also, *OntoOutput Requirements* provides addition of final information and updating it to any kind of communication portal such as mail services or on mobile phones.

For example, *OntoOutput Requirements* comprised of requirements for viewing results in Online Examination System, reports of bill generation in Hotel management System, consumer invoice in Power Utility System and view status of customer in credit Ranking System.

4.4 REP Models vs. *OntoAidedRE*

While reviewing the conventional REP models, it has been observed that each has certain lacunas over the former hence there exists no ideal REP model. Linear REP model is a basic model and can be used for simple and small projects only. This model provides a foundation for other models. But, there exist many problems such as freezing of requirements, no user feedback, no validation of requirements and no iterations of RE. However, Linear Iterative REP model solves some of the problems of Linear REP model such as freezing of requirements and requirements invalidation. But, there also exist some problems associated with Linear Iterative REP model such as it has no provision for reverse engineering. Conversely, Iterative and Spiral REP models suggest the user and domain information feedback in case of new iteration in the product. The new iteration is called a version but still no methodology has been set to manage the project [HKW+04].

Nevertheless, the problems with aforementioned models can be minimized by using inter relationships between different domains. Also, the other main element for requirements elicitation in RE is the use of requirement type. *OntoAidedRE* model bestowed engineers with such broad-spectrum requirement specifications. It transforms conventional RE with complementary semantics in a unifying ontological engineering process. Thus, *OntoAidedRE* model gathered the information using a uniform representation scheme that promotes cohesiveness between the requirement specifications set generated from different applications and creates a shared understanding from multiple dimensions. Also, to enable participation from diverse stakeholders, this layered approach is supported with ontological engineering process that provides rich modelling constructs with easily understandable semantics. RE for various information systems from various domains such as TPS, PCS, OAS and DSS have been practiced in this section to verify competence of *OntoAidedRE* over conventional REP models.

4.4.1 Linear REP Model vs. *OntoAidedRE*

As discussed earlier, Linear REP model consists of five activities namely; concept, problem analysis, feasibility and choice of options, analysis and modelling and requirement documentation. Being a simple model, it is mostly used for small projects with some less amount of complexity. Accordingly, we suggest Online Examination System as an example of TPS for practicing RE using Linear REP model. RE process starts with concept phase that recommend creative, critical teaching and learning to help students to cope with the information age. Then, problem analysis phase offers a dynamic elucidation that saves time to prepare examination papers, evaluate the examination automatically and paperless. Subsequently, detailed analysis enables development of requirement specifications such as requirement to set up exam, set up subject and subject code, set up students, manages teachers and view results. Thus, requirement document completes with aforesaid requirements and during validation phase requirement change transpire such as set up exam criteria. But, Linear REP model lacks to seize this requirement change. To overcome this problem, we use *OntoAidedRE* model for requirement specifications development as shown in Table 4.2.

Requirement Engineering (RE) using *OntoAidedRE* starts with developing *OntoPre Requirements*. It indicates requirement for Login to authenticate the user role such as administrator, teacher and student. Next, requirements to setup exam, set up subject and subject code constitutes *OntoInput Requirements*. Then, *OntoSystem Operational Requirements* signify requirements to register, edit and delete teacher, student or questions. However, *OntoSystem Parameter Requirements* cover requirements to setup student for examination and setup paper. Now, *OntoSystem Control Requirements* overcome the requirement change problem of Linear REP model that is to consider requirements to set up exam criteria. Lastly, requirement to view results connote by *OntoOutput Requirements*.

4.4.2 Linear Iterative REP Model vs. *OntoAidedRE*

As presented previously, Linear Iterative REP model comprised of five phases namely; requirements elicitation, requirements analysis, requirements agreement, requirements documentation and requirements validation. This model is useful for the system for which specifications should be pin point accurate and be validated multiple numbers of times through the potential stakeholders. Consequently, we consider Power Utility System as an example of PCS to establish requirements set with the help of Linear Iterative REP model. It begins with requirements elicitation phase to generate initial requirements such as requirements to monitor the work processes. Next, requirements analysis phase finds the problem in the requirements generated at elicitation phase. These

Table 4.2: Online Examination System using *OntoAidedRE*

S. No.	Layer Name	Requirement Specifications	Description
1	OntoPre-requirements	Login	To authenticate the examination administrator, teacher or student.
2	OntoInput Requirements	Setup Exam	To register name of examination
		Setup Subject	To register name of a subject
		Setup Exam Code	To link name of a subject and name of an exam
3 a	OntoSystem Operational Requirements	Register/ Edit/ Delete Teacher, Student and Question	To register, edit and delete the teacher and student information/ profile and questions
3b	OntoSystem Control Requirements	Setup Criteria Examination	To setup the criteria for examination papers, the number of questions to set and its duration
3 c	OntoSystem Parameter Requirements	Setup student for examination	To assign student for examination
		Setup paper	To select which question to set for a particular subject
			To inform that the question bank does not contain any question for a particular subject
			To inform that the paper is already set.
4	OntoOutput Requirements	View results	To view result in print/ report form

problems cover power source details and power supply mode. Subsequently, during agreement phase identifies requirement to control the work processes. Now, this new requirement forwarded to next iteration that is from elicitation phase to agreement phase. Finally, requirements statements generated and forwarded to the validation phase for validation and discussion. It is observed that, the model endures with requirements engineering again and again. To uncover the aforesaid problem, *OntoAidedRE* model is used to provide the acceptable requirements set as illustrated in Table 4.3.

OntoAidedRE model begins with establishment of *OntoPre Requirements* such as Login for system activation. Next, *OntoInput Requirements* covers requirements for keeping details of power source. Subsequently, *OntoSystem Operational Requirements* indicates requirements for power distribution operations such as add main grid function and high and low tension line functions. It also includes requirements for power supply mode operations such as seasonal and regular power supply, periodic power cut and failures. Now, *OntoSystem Control Requirements* covers the requirements to power control that is being identified in agreement phase of Linear Iterative REP model. Lastly, *OntoOutput Requirements* defines requirements for consumer billing such as billing as per category viz. high tension, low tension and direct reading.

4.4.3 Iterative REP Model vs. *OntoAidedRE*

As stated before, Iterative REP model comprised of three phases namely; elicitation, specification and validations. Since, this model performs RE in multiple iterations hence is suitable for softwares launched versions by versions. Accordingly, we consider Hotel Management System as an example of OAS for practicing RE using Iterative REP model. It starts with elicitation phase to acquire knowledge relevant to the hotel management to produce formal specifications and further for validation process. But, this model has no methodology set to manage the project. To eradicate this problem, *OntoAidedRE* model is practiced to establish requirement set as shown in Table 4.4.

Table 4.3: Power Utility System using *OntoAidedRE*

S. No.	Layer Name	Requirement Specifications	Description
1	OntoPre-requisites	Login	System activation
2	OntoInput Requirements	Power Source Details	To enter types of power sources such as hydro-power, hydel power, and nuclear power
3 a	OntoSystem Operational Requirements	Power distribution	Add main grid, high tension line, low tension line functions
		Power supply mode	Add procedure as per the mode such as periodic power cut, failures, seasonal and regular power supply
3b	OntoSystem Control Requirements	Power Control	To define total power requirement, number of generation units To Shut down the supply as per the schedule or failure
3 c	OntoSystem Parameter Requirements	Not applicable	######
4	OntoOutput Requirements	Consumer billing	Billing as per the category such as high tension, low tension or direct reading

Table 4.4: Hotel Management System using *OntoAidedRE*

S. No.	Layer Name	Requirement Specifications	Description
1	OntoPre-requisites	Login	To verify the system administrator and user identity by providing the user id and password
2	OntoInput Requirements	Customer details	To enter the required customer details in the system
		Employee details	To register employee with all appropriate details
		Room details	To open a new room type and related activities
3 a	OntoSystem Operational Requirements	Booking	To allow the customer for accommodation
		Cancellation	To allow the customer for non availing the accommodation
		Modification	To allow the modification to room assigned
		Check in/ Check out	To admit and relieve of the customer from hotel
3b	OntoSystem Control Requirements	Not applicable	######
3c	OntoSystem Parameter Requirements	Membership details	To validate the customer for membership, if meets the defined criteria
		Package details	
4	OntoOutput Requirements	Bill generation	To view the final statement of customer in print/ report form before check out

OntoAidedRE model provides a systematic methodology for practicing RE. It begins with establishment of *OntoPre Requirements* to verify user identity. Next, *OntoInput Requirements* covers requirements for keeping details of customers, employees and rooms. Then, *OntoSystem Operational Requirements* specify requirements for booking, cancellation, modification, check in and checkout activities. Now, *OntoSystem Parameter Requirements* present the requirements for membership provision and package details. Lastly, *OntoOutput Requirements* defines requirements for bill generation to view the final statement of customer before checkout.

4.4.4 Spiral REP Model vs. *OntoAidedRE*

As discussed earlier, in Spiral REP model, one spiral represents the complete version of requirements. Each spiral is divided into four quadrants called specification extraction, discussing and analyzing requirements, requirements documentations and requirements validations. The main characteristic of this model is to handle speciation delay and requirements change Thus, we consider Credit Ranking System as an example of DSS to establish requirements set using Spiral REP model. It starts with requirement extraction phase to identify requirements by consulting with customers, developers and users. It specifies credit ranking policies with the help of viewing the credibility of a customer such as the financial status. While, discussing and analyzing requirements phase helps to get aware with the specified standards as per the class of a person such as service or business class. Next, requirements document considers as a base for controlling changes and evaluating future policies. Now, Validation is done for controlling the quality. This phase continues repeating other requirements development phases because of identified deficiencies and gap between requirements. To reveal this complexity, *OntoAidedRE* model is used to provide the adequate requirements set as illustrated in Table 4.5.

Table 4.5: Credit Ranking System using *OntoAidedRE*

S. No.	Layer Name	Requirement Specifications	Description
1	OntoPre-requisites	Login	To verify person's identity such as business class or service class
2	OntoInput Requirements	Registration	To record the personal details
		Standard definition	To define and enter the standard limits a per the class
3 a	OntoSystem Operational Requirements	Present working status	For service class, present salary, perks and previous experiences are defined. For business class, products, target market, previous balance sheets
3b	OntoSystem Control Requirements	Not Applicable	######
3 c	OntoSystem Parameter Requirements	Present financial status	For service class, add income tax returns, loans, assets and liabilities, sources of income For business class, add balance sheets, loans repayment track records, share value and patterns, assets and liabilities
4	OntoOutput Requirements	View status	The system will generate the reports as per the processed data

OntoAidedRE model helps to avoid repetition of RE activities and handles the cost and schedule of the system. It initiates with establishment of *OntoPre Requirements* to verify the person's identity such as business or service class. Next, *OntoInput Requirements* specify requirements for user registration for keeping personal details and standards to enter the standard limits as per the class. Subsequently, *OntoSystem Operational Requirements* indicate requirements to add or modify the present working status of business or service class persons. Now, *OntoSystem Parameter Requirements* cover the requirements to provide present financial status according to rules defined for service and business class. Lastly, *OntoOutput Requirements* defines requirements to view status that is generated by system as per the processed data.

4.5 Comparative Study

A comparative study of different conventional REP models and *OntoAidedRE* on the basis of various parameters is illustrated in Table 4.6.

It is observed that conventional REP model such as Linear REP model is restricted to RE of TPS and MIS project types as it follows sequential life cycle. Next, Linear iterative REP model is confined to PCS project type as it is helpful for the system in which the specifications should be pin point accurate and validated multiple numbers of times through the potential stakeholders. Subsequently, Iterative REP model performs RE in multiple iterations and hence it is better for OAS project type. Then, Spiral REP model repeats RE activities until a decision is accepted about requirement document and hence works well for DSS project type. On the other hand, *OntoAidedRE* functions with all project types as it enables knowledge driven RE depending upon inter-relationship between different domain requirement types.

It is found that Linear REP model and Linear Iterative REP model limits the requirements development to small and medium project size as it works for projects having less complexity and projects facilitated with multiple validations

Table 4.6: Comparison of Requirement Engineering Process Models

Parameter	**Linear REP Model**	**Linear Iterative REP Model**	**Iterative REP Model**	**Spiral REP Model**	**OntoAidedRE**
Project Type	TPS and MIS	PCS	OAS	DSS and ES	All
Project Size	Small and Medium	Medium	Medium and Large	Medium and Large	Small, Medium and Large
Project Team [in number]	1 to10	26 to 100	11 to 25	11 to 25	Only 1
Project Quality	Upto 65%-74%	Upto 65%-74%	Upto 51%-64%	Upto 75%	Upto 100%
Project Prioritized Element	Cost and Time	Cost and Functionality	Functionality	Functionality	Time, Cost and functionality
Project Key Element	Not Available	Not Available	Not Available	Not Available	Available

respectively. Then, Iterative REP model and Spiral REP model helps in RE activities of medium and large project size because these models suggest the requirement development versions by versions. Conversely, *OntoAidedRE* coordinate all project sizes as it provides cohesiveness depending upon changes in knowledge.

It is noted that, every conventional REP model requires project team between one to twenty five persons as these REP models work on various RE activities such as requirement elicitation, analysis, documentation etc. whereas *OntoAidedRE* works on layered structure based on requirement types such as *OntoPre, OntoInput, OntoSystem* and *OntoOutput* requirements.

As illustrated in Table 4.6, conventional REP models provide upto 75% project quality as these REP models have no efficient methodology to manage the project. On the other hand, *OntoAidedRE* provides systematic methodology for practicing RE and thus improves the project quality to maximum extent.

It is observed that, presence of project prioritized element such as cost, time and functionality varies across conventional REP models. Linear REP model includes cost and time but not the functionality as no user interaction or feedback is provided in its RE activities. For Linear Iterative REP model, cost and functionality matters as compared to time as it include multiple validation activity. On the other hand Iterative and Spiral REP model involve only the functionality element as these models typically works on iterations and versions Conversely, *OntoAidedRE* includes all the prioritized elements namely; cost, time and functionality because it includes requirement types to confirm the project functionality and thus no cost and time elements are sacrificed.

Project key element includes well defined knowledge artifact to establish the open-ended progression of RE process. Since all conventional REP models work on domain driven RE hence not involve this project key element. On the other hand, *OntoAidedRE* provides knowledge driven RE and used to strengthen the generality of concepts depending upon requirement types for different application domains.

4.6 Summary

Requirement Engineering is promising process and especially draws on with the aim of amenable to analysis, communication, and subsequent implementation. In this chapter, we have discussed the parameters of study related to project such as Project type, Project size, Project team, Project quality, Project prioritized element and Project key element. These play a very significant role in RE for various types of projects. The conventional REP models with advantages and limitations in term of practices have been highlighted in next Section. Also, we have presented Ontology Aided Requirement Engineering model (*OntoAidedRE*) covering requirement type, practices and suitability. Consequently, we have compared conventional REP models namely; Linear, Linear Iterative, Iterative and Spiral models with *OntoAidedRE*. The study reveals that none of conventional REP models acquire all project parameters. This, in turn, often severely affects the successful completion of projects. We have presented *OntoAidedRE* to show a knowledge-driven as opposed to process-driven approach to RE. It can be put into practice to overcome the problems of conventional REP models and consequently the project parameters optimally contrived by adapting *OntoAidedRE*.

CHAPTER 5

Approaches for Ontology Based Reusability

5.1 Introduction

Currently, the absolute prospective of software reuse in highly variable domains cause rigid to unleash the software applications. Also, many business concepts constrained by tight regulations are difficult to protract. Building an application from scratch is a resource intensive process for information system, passing by domain-specific variability. When dealing with legacy systems, the cause emerges especially due to the time elapsed between the requirements specification for each developed module and the present. Usually, each application has its own configuration, stores its own data and is guided by its own business rules [CM07]. As a result, during software development, the reuse prospect is generally expected condition [RAM05, FK05].

In addition, it is extensively reckoned that the development and utilization of reusable software artifacts is necessary for improving software development efficiency and software prominence. Most software development methodologies recognize the utility of reuse, and some even provide processes and contrivances to directly support it. Effective software reuse requires collections of designed-for-reuse software components. In addition, mechanisms to retrieve reuse candidates to adapt and create new ones using the information provided by similar components [MER+03]. Moreover, it is needed to bind these elements using a software process that truly accede to software reuse. In this context, ontologies can play an important role. Ontologies have become an important mechanism for building software, since these can be used to overcome barriers created by disparate vocabularies, representations and tools. Ontology may take a variety of forms, but necessarily it includes a vocabulary of terms, and some specification of related meaning [MF03, OVR+06]. This includes definitions of concepts that are inter-related which collectively impose a structure on domain and constrain the possible interpretations of terms. Attempts have been made to reconcile the terms with feature modelling, domain modelling, etc. However, there is strong need of combining the conceptions of domain with stronger extensibility and with indexing knowledge population. Accordingly, we develop ontology based approaches for reusability. The role of ontologies is to capture domain knowledge in a generic way and to provide a commonly agreed upon understanding of a domain.

In view of this, we introduce *Ontop4ViewReuse* framework with ontology validated composition. It caters in highly variable domains due to emergence of several dimensions of software development in the course of various abstraction levels. It is based on ontology oriented systematic *P4View* approach for reusing. Next, *OntoReuseAlgo* for knowledge integration and reuse towards process planning in software development is commenced. It is based on *Ontological Knowledge Modelling* to provide reusable and shareable engineering applications. Lastly, we develop Ontological reuse (OnR) from Object-Oriented Reuse (OOR). It potentially applies all the phases of OOR such as development of reusable artefacts, representation and classification of artefacts into repositories, and utilization of the artefacts from repositories. Also, a range of classes of reuse have been identified for comparison of OOR and OnR.

5.2 Background

There exists software reuse around for years and involves variety of concepts. Early software reuse practices focused on code and implemented in adhoc or opportunistic manner. Also, the active areas of reuse research in the past twenty years include domain engineering methods, reuse design, design patterns, domain specific software architecture and component. All these areas weli catered by Object-Oriented Reuse [DEV02, SS03, HL01, GH95]. Consequently, reuse has become planned and systematic. But, ontology is adapted to enhance Object-

Oriented Reuse. Ontology is a formal explicit description of concepts in a domain of discourse and oriented towards a systematic method for reusing. In this view, the reuse approach is introduced early in the life-cycle of software development. Accordingly, any product of the software life cycle can potentially be reused. It is a formal and well-documented process which is no more domain-specific and can be recreated. This approach follows a well-planned, lucrative, and productive strategy. In addition, it allows the use of existing software or software knowledge to construct new software. In this section, we describe the subclasses of reuse followed by Object-Oriented Reuse process and then Ontological Reuse process.

5.2.1 Reuse Subclasses

Ontologies have great potential to deal with software reuse predicaments of various aspects such as domain specificity, fixed functionality, well bounded interfaces, performance expectations, and demonstrable excellence [HAM04, TA00]. Consequently, we have identified and explained range of subclasses of reuse as follows:

Software Component Reuse

Software component reuse is the software engineering practice of creating new software applications from existing components, rather than designing and building them from scratch. Reusable components can be requirements specifications, design documents, source code, user interfaces, user documentation, or any other items associated with software. All products resulting from SDLC have the potential for reuse. The practice of component reuse supports the motivation for development of customized applications. Its benefit includes reduced application development time, reduced application cost, and improved application quality [KUH98].

Software Architecture/ Design Reuse

The reusability of software design and software architecture refers to the re-application of representations of one system or component to the construction of similar ones in a problem domain. It is observed that reusability can be enhanced if the software design and software architecture are explicitly represented and if the representation can be easily understood and manipulated (modified and reconstructed) towards a varieties of target systems [SS99].

Software Requirements Reuse

Requirements reuse is an approach to systematically use existent requirements documents for reducing the general effort inside the software life cycle. From the point of view of improving requirements engineering, requirements reuse aids by recording the adopted suppositions, made decisions, and adopted alternatives for future reference. It provides the ease of the change management of requirements. Moreover, requirements reuse is helpful in the assistance, guidance and advising for the requirements engineer in the process of requirements acquisition [OM99].

Software Process Reuse

Software process reuse represents a new practice for software production in which a conceptual knowledge representation is used to represent and guide development activities. During software process reuse, process engineers specify a software process that is tailored for project goals and other resource constraints, and then enact the process as a guide for developers [HOL98].

Software Technology Reuse

Software technology reuse provides certain types of services to their users such as storage, searching, inspecting and retrieval of artifacts from different application domains, and of varying granularity and abstraction, loading, linking and invoking of stored artifacts, and specifying artifact relationships [BAR06].

Software Experience Reuse

Software experience reuse enables people to effectively reuse components. It is observed that, the visual interface design is perhaps even more important than the user need as a succinct way of communicating the purpose of the component to designers [ABH+99].

5.2.2 Object-Oriented Reuse Process

To effectuate reuse, three major engineering activities must be addressed as shown in Figure 5.1. Firstly, reusable artifacts must be intentionally designed and developed. Secondly, reusable artifacts must be represented, classified, and entered into and removed from appropriate repositories. And, subsequently tools and processes must be developed that support finding, understanding, modifying, and composing artifacts [WER97, SMJ02]. Now we discuss them as follows:

Development of Reusable artifacts

Development of reusable artifacts concerns with the work required to establish a set of software artifacts that can be reused by the software engineer. Its purpose is to identify, model, construct, catalog and disseminate a set of software artifacts that can be applied to existing and future software in a particular application domain.

Representing Reusable artifacts

The most difficult problem with reuse is developing a suitable representation for artifacts. In particular, it resembles a representation that encodes the semantics of artifacts. Users trying to solve a problem with own knowledge and semantics can locate an appropriate reusable artifact. Such artifacts must be retrievable by multiple pathways to support variety of different ways in which users may access them. Furthermore, representation allows for a variety of different perspectives on stored artifacts, and permit versioning and configuration management activities. Also, it allows for representation of partial and uncertain information. This allows artifact developers to evolve the designs over time by permitting well-defined aspects to be expressed with certainty, and less well-defined aspects to be left fuzzy.

Repository Reusable artifacts

Artifacts must be classified and entered into repositories, once artifacts have been represented. Classification of artifacts is an indexing issue. As such, artifacts are classified in order to indicate the type and relation to other artifacts.

There exist two well known schemes for repository classification accomplishment such as enumerative and faceted. Enumerative scheme divides the universe into a collection of domains and sub domains. However, faceted approach does not rely on a prior division of the universe into domains, but rather synthesizes a classification of an artifact based on the selection of properties from a collection of facets.

Supporting the Reuse of artifacts

Once artifacts have been developed, represented, and categorized into repositories, the next concern is to utilize this wealth of information. Software developers need tools and processes for finding, understanding and using reusable artifacts.

Finding artifacts

To find artifacts, users describe the requirements and tools included for requirement satisfaction. This simple declarative model is rarely achieved in practice as most representations are insufficient to support sophisticated queries and reasoning. Sophistication of techniques for finding information is dictated by the representation scheme. Hence, the extent of OOSE to support retrieval will be dictated by representation scheme.

Understanding artifacts

Once artifacts have been located, it is necessary to understand in order to use these artifacts. OOSE has potential to enhance the understandability of software artifacts. Strength of an object-oriented approach is that it offers a mechanism that captures a model of the real world termed as objects.

Using artifacts

This activity has been viewed as a fundamental part of development process. There exist varieties of different ways in which an artifact may be reused. A retrieved artifact that is useful without modification need only be integrated i.e. “plugged" into the system. However, if an artifact requires modification, it may be

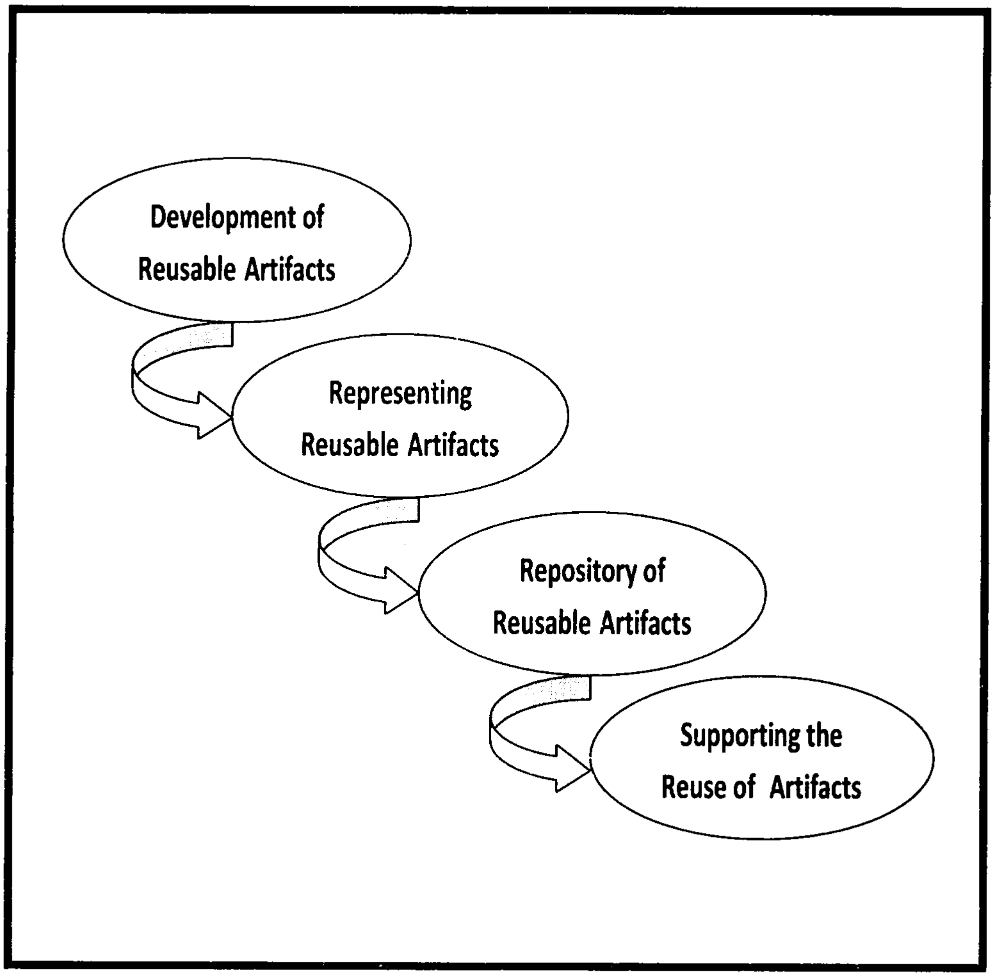

Development of Reusable Artifacts
Representing Reusable Artifacts
Repository of Reusable Artifacts
Supporting the Reuse of Artifacts

Figure 5.1: Object-Oriented Reuse Process

necessary to refine or compose it. Then, combination of retrieved artifacts is required. In an object-oriented system, the refinement and composition tasks are potentially simpler. By using inheritance, refinement is described as top down process of specifying the differences between inherited state and behavior of an existing object and requirements of desired object. With excellent support of encapsulation and message protocols, composition is a bottom up process of connecting together the proper object building blocks to form the desired component [SS04].

5.2.3 Ontological Reuse Process

Ontological reuse process starts with the identification of knowledge sources useful for the application domain that differs in represented content as well as in the formalization [SMJ02]. An automatic integration of the source knowledge does not mean only the translation of the representation languages to a common format, but also the matching of the resulting schemes. Ontological reuse process has been introduced early in the life-cycle of software development as it is a formal and well-documented process which is domain unambiguous and can be recreated as shown in Figure 5.2. The process is describes as follows:

Determine Scope

It refers to defining concepts in the domain (classes). There exists no correct ontology of a specific domain. Ontology is an abstraction of a particular domain, and there always subsists viable alternatives. This abstraction must be determined by the use to which the ontology kept and by future extensions that are already anticipated.

Define Taxonomy

It ensures arranging the concepts in a hierarchy (subclass super class hierarchy). Since, the hierarchy must be efficient or reliable hence user opinions may differ to select the type of hierarchy to define taxonomy. The types include top-down or a bottom-up fashion.

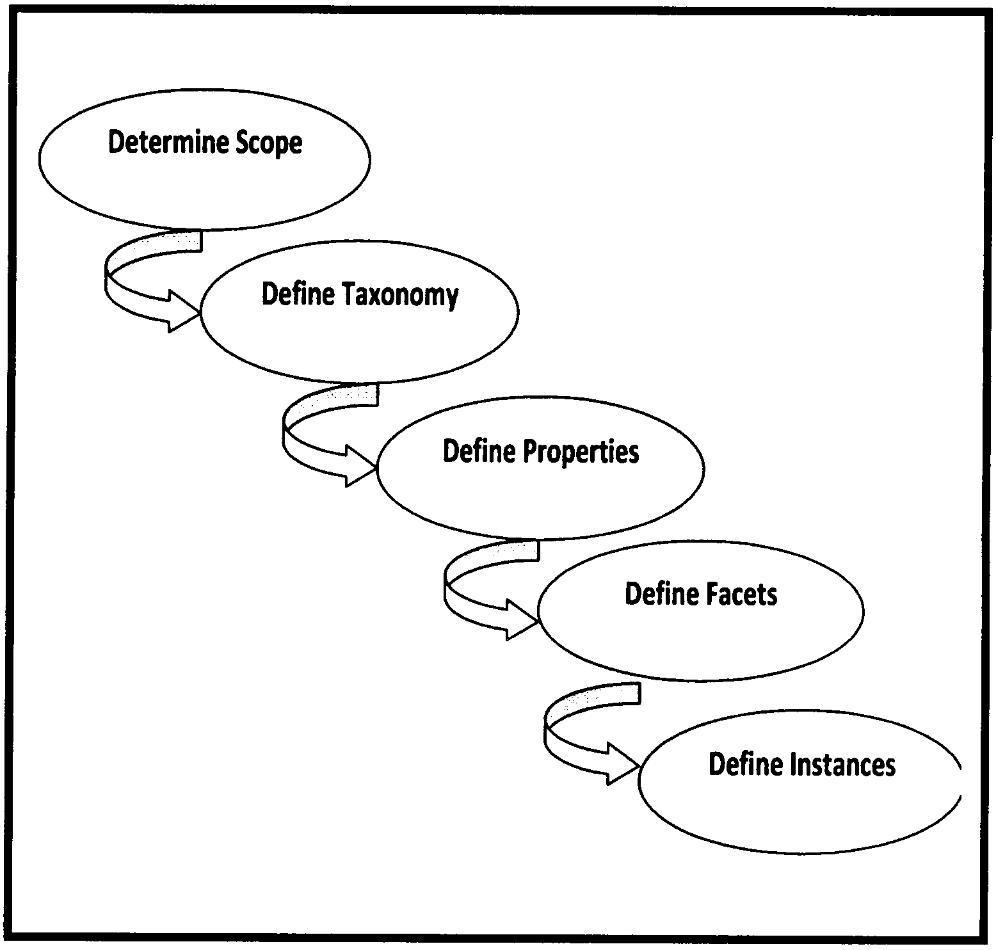


Figure 5.2: Ontological Reuse Process

Define Properties

It defines attributes or properties (slots) that classes can have and constraints on their values. While attaching properties to classes, it has been observed that it immediately provide statements about the domain and range of these properties. There exists a methodological apprehension between generality and specificity such as flexibility (inheritance to subclasses) and detection of inconsistencies and misconceptions.

Define Facets

It asserts defining individuals and filling in slot values with cardinality restrictions and relational characteristics such as symmetry, transitivity, inverse properties, and functional values.

Define Instances

Filling the ontologies with instances is a step concerned with creating a knowledge base. It defines individual instances of various classes and filling in specific property such as specific slot value information and additional slot restrictions.

5.3 *P4View* Approach Based Framework

The key challenge while managing and characterizing reusability in highly dynamic domains is to identify the relations and representations of software artifacts and resources involved. In such context, we have proposed *P4View* approach for building systems, adaptable to each user with common characteristics. This approach makes use of ontologies pact to the knowledge and experience of users, history of previous actions, goals, intentions, interests and preferences. Using this approach, *OntoP4ViewReuse* framework for software development is described. Framework emphasizes on different levels of abstraction that provides an unambiguous terminology, allowing its reuse and easy extension. The detailed description of *P4View* approach and *OntoP4ViewReuse* framework is as follows:

5.3.1 *P4View* Approach

We have proposed a *P4View* that resorts to available ontological knowledge and are implicitly tailored to specific application needs. In turn, it cannot be reused in different settings. While, in *P4View* approach additional ontological primitives like properties and axioms are supported explicitly. *P4* stands for *Pretence-Persuade-Problem-Product*, defining the various abstraction levels to be accomplished during the software development. Figure 5.3 represents the overview of *P4View* approach, references ontologies. Each of these views represents a meticulous attribute of the ontology and defined as follows:

Pretence View

Pretence view caters by high level ontology that includes representational, terminological and social ontologies. Representational ontology helps in identification of knowledge sources useful for the application domains that differ both in represented content and in formalization. In addition, terminological ontology describes general concepts that are independent of a specific domain or a problem such as space, material, objects etc. Lastly, social ontology includes the terms such as actor, position, role, authority, responsibility or commitment.

Problem View

An automatic translation of the source ontologies from a common format to the representation languages is carried out at *Problem view*. It is supported by domain ontology that is comprised of informational, intentional and static ontology. Informational ontology structures the standardized storage of information. While, intentional ontology describes aspects of world of intentions, goals, beliefs alternatives and elections of involved users. Lastly, static ontology describes the terms such as entity, object and relationship.

Persuade View

The identification of terms specific to the problem resolution methods and or tasks is involved at *Persuade view*. This view includes dynamic and method

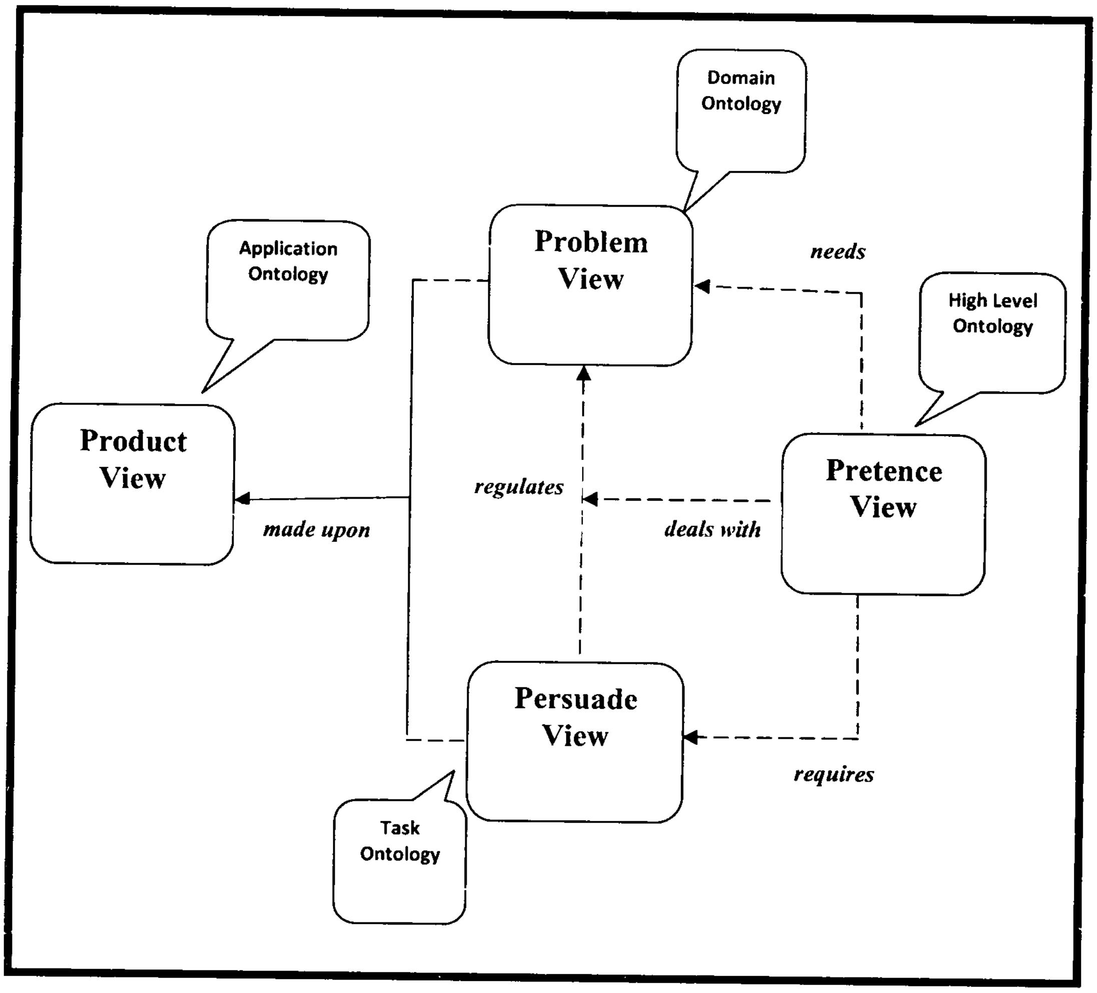


Figure 5.3 ***P4View*** **Approach**

ontology. Dynamic ontology articulates terms such as process, state or state transitions. In addition, matching of the ensuing method is accepted at *Persuade view*. Hence, task ontology signifies method ontology. It offers a reasonable point of view to the knowledge of the domain.

Product View

It categorizes the roles played by the domain entities when executing an activity. It includes the controlled vocabularies, informal and formal hierarchies, frames, value constraints and generic logical constraints prolonged with application ontology. Finally, application ontologies revealed the reuse source vocabularies to a large extent in *Product view*.

5.3.2 *OntoP4ViewReuse* Framework

Ontop4ViewReuse is based on ontology oriented systematic *P4View* approach for reusing. *OntoP4ViewReuse* bring about to apply the ontology of varying levels of notion such as high level, domain, task and application ontology. This cataloging of ontologies is useful for the development of reusable and high-quality software application. In addition, through ontologies, the eliciting and modelling of the knowledge is being carried out using *P4View* approach that concentrates on different levels of abstraction. Initially, the general knowledge of the domain is elicited and specified in one or more views and finally serves the next views to develop the specified application. As a result, phase wise procedure is introduced to construct *Ontop4ViewReuse* framework. These phases are described as below:

High Level Ontology Phase

We have consider all possible aspects of a system such as its type, associated directives and activities performed by people in various types of system with its own rules, to define the scope of this ontology phase. As depicted in Figure 5.4, *System* is composed of *System's Type* in which different *Activities* are performed by *Human Resources*. We also include the fact

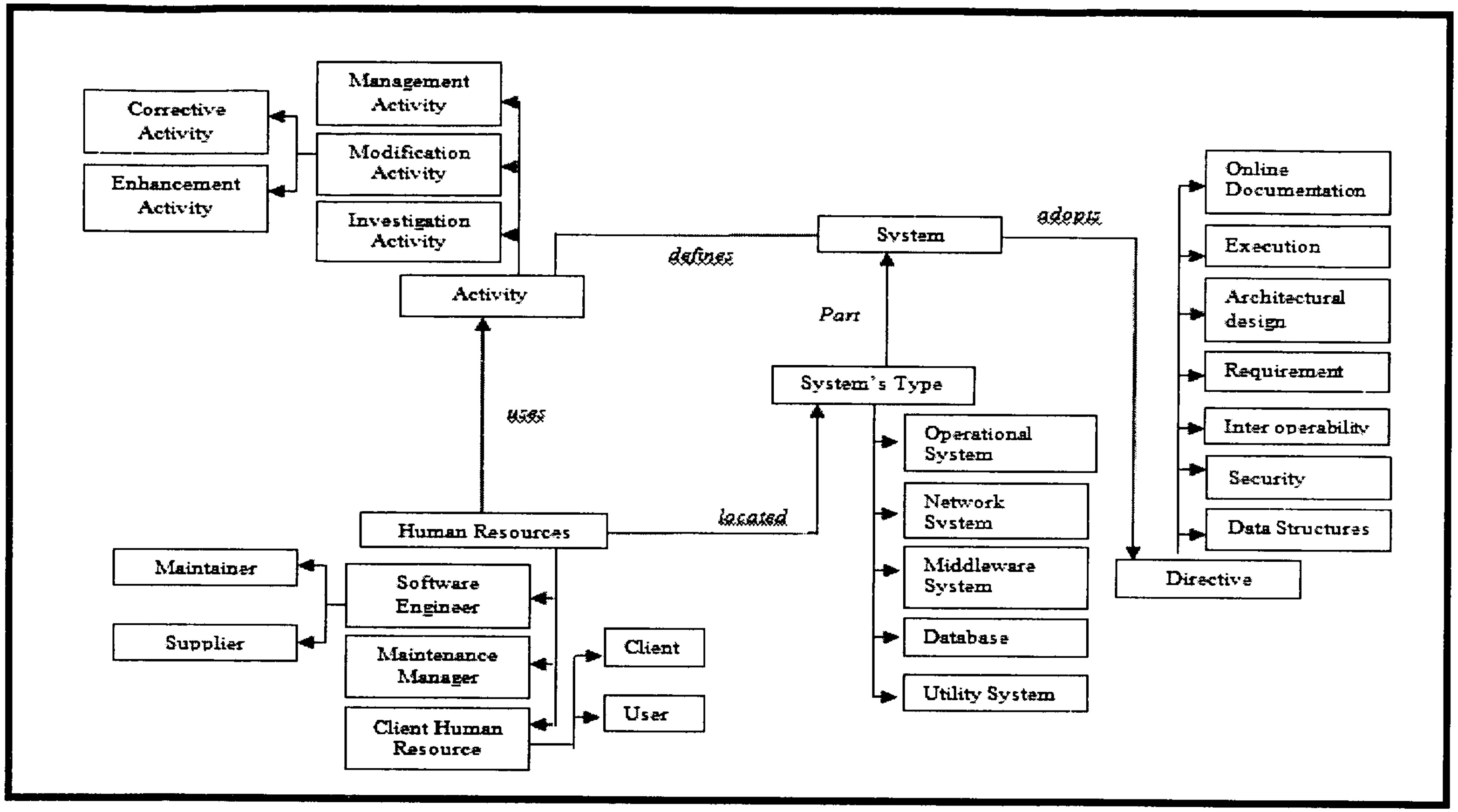


Figure 5.4 *Pretence View* using High level Ontology

that a system adopts *Directives* to be followed in the execution of the tasks. Database management systems, utilities and system softwares such as operational, network and middleware are included in *System's Type*.

Activities are majorly classified into three types such as investigation, modification and management. *Activity* uses one or more input artifacts and affects one or more output artifacts. It precedes some activity and or part of some other activity. Investigation activities focus on assessing the impact of undertaking the modification where as management activity contributes to the configuration control of the products. In addition, modification activity includes corrective and enhancement activity aiming at adaptive, preventive and perfective continuation during the product construction.

Directives support system such as online documentation to help and contrivance guidelines, Architectural design for dynamic library reuse and Requirement for change to specify the new one. Also, data structures such as structure of data files or databases are mentioned with Interoperability to feature the communication with other systems and Security to ensure the integrity of system. Finally, execution of system is included for performance or instability measurements.

Human Resources include software engineers such as suppliers and maintainers. Supplier develops the system and maintainer maintains the system. In addition, maintenance manager is responsible to conduct concerned maintenance procedures and client human resources include clients and users.

Domain Ontology Phase

In the context of software engineering a domain defines as an application area, for which software system has to be developed. Domain Ontology refers to reuse-based process used to define scope and structure. Also, it illustrates reusable attribute for various kinds of system having different domain specifications and support specifications. Taxonomy of domain can be decomposed into its *Job* and *Components* as shown in Figure 5.5. It consider and represent similarities and

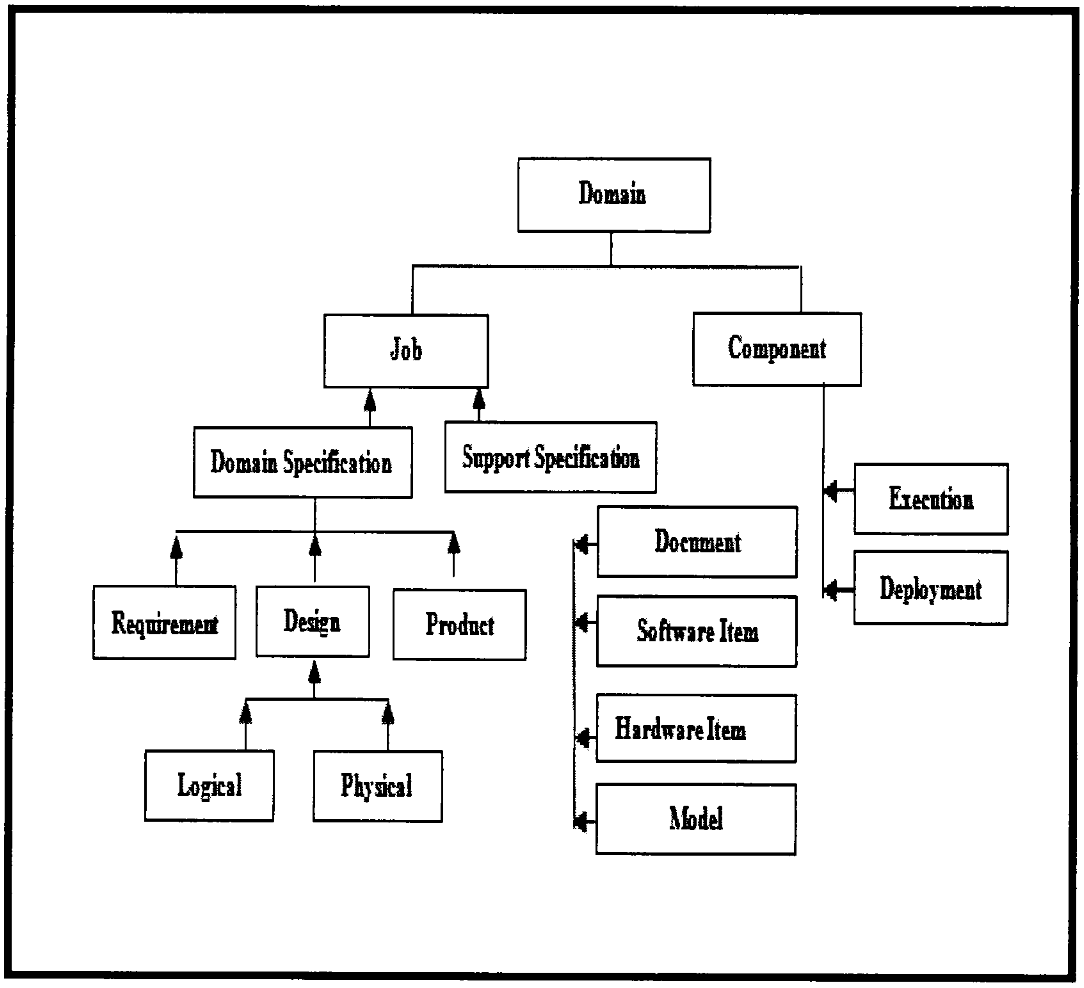


Figure 5.5 ***Problem View*** **using Domain Ontology**

difference between the systems within a domain. *Components* represent all the coded artifacts that compose the software program itself. These are classified into execution components generated for the software execution and deployment component for composing the executable program.

Job is decomposed into two kinds according to the type of specifications such as domain specifications and support specifications. Domain specifications are composed of requirement, design and product specifications for describing the system's behavior and structure. Different view models may be defined to redefine the design specifications at logical and physical level. Moreover, support specifications helps in operating the system such as document to illustrate the results obtained from the study of the reuse of requirements, software design, and generic architectures. Also, it includes identification of hardware to install the system and the compatibility of software with it. In addition, model illustrates information in an understandable fashion through formal presentation.

Task Ontology Phase

Tasks or procedures are the structured descriptions used in a software development activity such as *Methods, Techniques* and *Assertions* as shown in Figure 5.6. *Methods* are the kind of systematic procedures with semantic and syntactic definition to be followed. On the other hand, *Techniques* are the logical procedures less formal and rigorous than a method. *Techniques* begin with requirement elicitation that includes procedures (such as interviews and brainstorming etc.) to assist in the identification of requirements. Subsequently, modelling techniques adopts specific modelling language to define the systematic solution for a problem followed by programming technique (may be structured or object oriented). Consequently, testing techniques include such as white or black box etc. Lastly, maintenance techniques classified into reverse engineering, re engineering, impact Analysis and program comprehension to assist in the maintenance of program. Lastly, *Assertions* defines directives or the standards such as guidelines or norms defined to use the system.

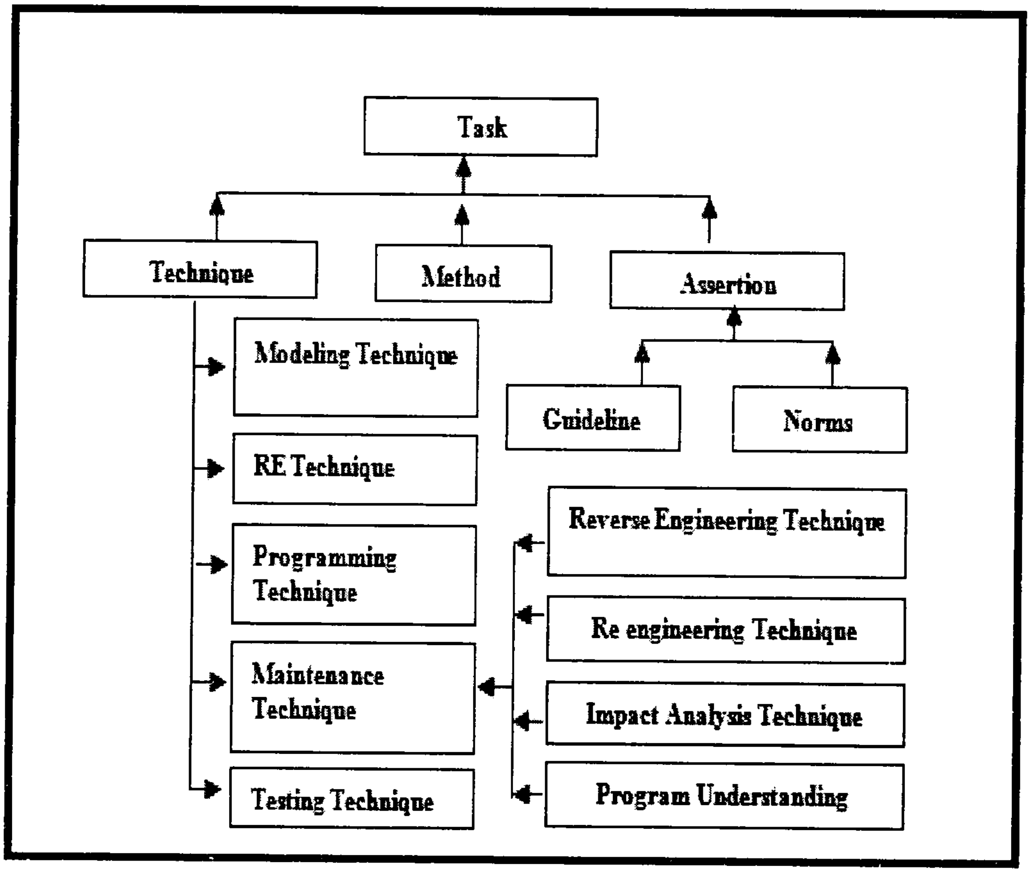


Figure 5.6 ***Persuade View*** **using Task Ontology**

Application Ontology Phase

Application Ontology organizes the process that builds products from software elements abstracted through domain and task ontology. Application ontology depicts *Concept* and *Task* that compose an application. Also, *Properties* associated with each *Concept* and *Restrictions* applied to an application is specified as illustrated in Figure 5.7. *Concept* is aiming at satisfying the application needs of a specific kind of user and performance expectations. Next, *Property* refers to a component that supply the functionality needed by the user. And, *Restrictions* signify to constraints that may be logical or value applied during the software development to validate the requirements. Lastly, *Procedures* indicate functionality of product that must be fixed, along with its preconditions and post-conditions. Thus, users will know exactly the product's function under all circumstances.

Now, we integrate all these phases in a single conceptual framework *OntoP4ViewReuse* as shown in Figure 5.8. The framework contemplates on different levels of abstraction namely; *Pretence, Problem, Persuade and Product views* consistent with types of defined ontologies. Levels of abstraction relate to the completeness, and to the value of reusable property. Level 1 abstraction signifies the constituents as agreed for repository population on the basis of generalized stipulate only. This level established with the components of high level ontology such as representational, terminological and social ontology. Representational ontology indicates *system, system's type.* Next, *Activities* and *directives* present terminological ontology *and human resources* are explicitly defined inside social ontology. The completeness of the high level ontology components is well recognized at this level.

Subsequently, Level 2 abstraction configures domain ontology constituents such as informational, intentional and statical ontologies. Informational ontology defines the *components*. Statical ontology describes domain specifications and intentional ontology indicates support specifications. As discussed earlier, these specifications are included in *Job* constituent.

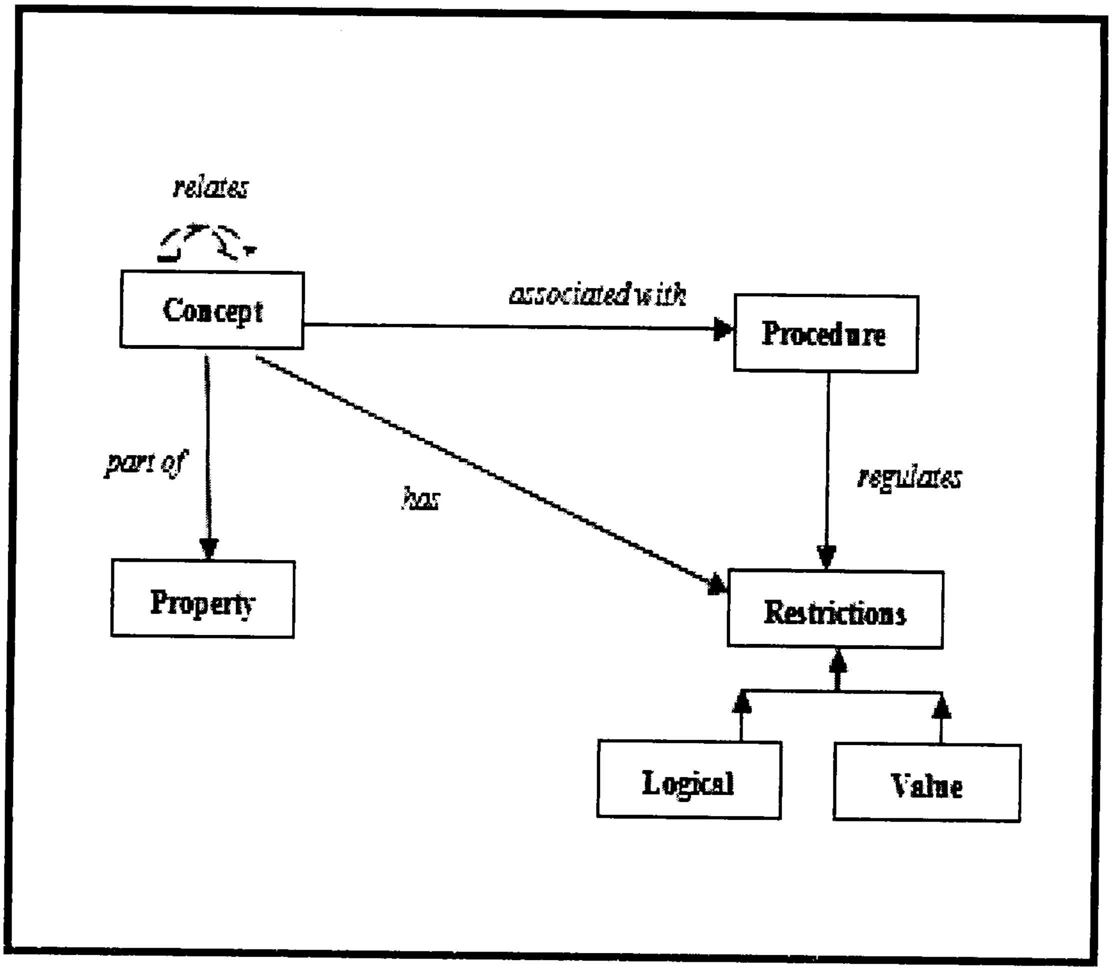


Figure 5.7: *Product View* using Application Ontology

Next, Level 3 abstraction renders task ontology. This level determines many idiosyncratic such as *methods, techniques* and *assertions* that are perceptibly delineate in methodical and dynamic ontologies of task ontology. Lastly, Level 4 abstraction structures the application ontology constituents such as vocabularies, hierarchies (formal and informal), frames and constraints (may be logical and or value) that is to be released to users and it verifies completeness. A *concept* that is part of *property* associated with *task* and *restrictions* offer a highest degree of abstraction.

5.3.3 Case Study

To tap the full potential of existing domain-relevant knowledge sources, ontology is being accepted. At this instant, reuse with the help of ontology is defined as the process in which ontological knowledge is used as input to generate new ontologies. Depending on the content of the knowledge sources and domain overlapping, the implications of reuse in the overall development process can be clarified. We address the reuse process to a greatest extent using *OntoP4ViewReuse* framework in the domains of e-Recruitment and e-Medicine.

Case I- e-Recruitment portal

e-Recruitment portal allows a uniform representation of job postings, job seeker profiles and semantic matching in job seeking and procurement tasks. It facilitates to support common practices from the industry and to maximize the integration of job seeker profiles and job postings from different organizations. High level ontology underlying this job portal is aligned to established domain-specific standards and classifications. The selection of high level ontologies is followed by the customization and integration to the new ontology. We identified the sub-domains of this system such as networked system type includes professional, educational and industrial areas. Next, domain ontology is used to define concepts representing competencies to describe job requirements as well as job seeker skills. Due to the domain setting, component classification standards such as the occupation component and the industrial sectors component have to be completely integrated in the new ontology. To extract the relevant fragments from task ontology, we compiled a small conceptual vocabulary from various job portals and job procurement web sites and matched these core concepts to the source ontology. The usage of the ontology in semantic matching tasks requires that it is represented in a highly formal representation language. For this reason the implementation of new ontology has realized by translating several semi-structured input formalisms using application ontology.

Case II- e-Medicine portal

We developed e-Medicine portal for lung pathology to analyze the practice in a retrieval system for representation content data in the medical domain. This e-Medicine portal provides a concept-based reuse technique and semantic annotation of pathology reports. To develop high level ontology, we identified anatomical, clinical and pathology-specific system type and separate the application relevant knowledge from the general purpose medical knowledge.

On the other hand, domain ontology covers both domain and application-relevant knowledge that is specific to the health-care institution involved in the project. Also, medical components such as digital anatomist are tailored to domain ontology. For this purpose domain experts identified four central concepts such as "lung", "pleura", "trachea" and "bronchia" and included to the task ontology. Also, this standard format for the representation of patient data and patient records and immunohistology guidelines used by domain experts in diagnosis procedures, significant parts of the pathology domain, are integrated in task ontologies.

A large part of pathology specific method such as vocabulary with a lexicon generated from an archive of medical reports resulted in further refinements of the application ontology. It is implemented to describe the anatomy of typical diseases aligned to generic and core medical concepts. Additionally, application ontology is needed for semantic annotation required a maximal coverage of the vocabulary used by domain experts in medical reports.

5.3.4 Benefits of *OntoP4ViewReuse*

We propose *OntoP4ViewReuse* framework using *P4View* approach to utilize the content of the source ontologies to a maximal extent depending on their particular domain and level of formality. Adopting *OntoP4ViewReuse* based software development process attracts a number of benefits to both the end-users and developers. These include the following:

Savings in costs and time: As a developer uses already pre-defined components, hence, the activities associated with components specification, design and implementation are now replaced with finding components, adaptation to suit new requirements, and their integration. Though, ontology based reuse certainly attract additional effort, time and cost. These costs, however, can be offset by savings in a number of different software projects.

Increase in productivity: It has been shown that reusable artefacts developed from *OntoP4ViewReuse* can be viewed as abstract level of concepts drawn from a given problem domain. Hence, working with such higher level of abstraction leads to an increase in development productivity

Increase in ease of maintenance: Systems constructed of reusable parts are usually simpler and more abstract. Also, the designs are closer to the problem domain and their consistency is greater. This of course has very positive impact on the quality of such systems maintenance.

Increase in reliability: *OntoP4ViewReuse* suggests that the life-span of reuse artifacts is much greater than that of any individual product. Thus, the reliability of such artifact is also increased. This also leads to an improved reliability of systems built of reusable components rather than of those built entirely from scratch.

High speed and low cost replacement of aging systems: Systems developed using *OntoP4ViewReuse* shares a very large collection of concepts via ontology, thus, have become significantly multifaceted. Such systems need less effort during porting or adaptation to new hardware software environments. Also, the reusable components of the system are technology intensive and very expensive to develop but sharing that cost across several systems certainly reduce it when a global replacement of computing resources have effect.

5.4 *Ontological Knowledge Modelling* Based Algorithm

Reuse also responds to an increasing insist for highly reliable, high excellence and less expensive systems. Accordingly, knowledge reuse benefits and improves the process planning in software development greatly. Process planning is an intermediate phase between design and implementation. Lucidity and prescribed specification of concepts play a key role in the inclusion of reuse during process planning. Therefore, a most important issue is to build a common conceptual base characterized by knowledge. Our exploration focuses on this task through developing *OntoReuseAlgo* based on *Ontological Knowledge Modelling.* The brief description of *Ontological Knowledge Modelling* is illustrated below:

5.4.1 *Ontological Knowledge Modelling*

We propose an *Ontological Knowledge Modelling* for knowledge integration and reuse towards process planning in software development. It constitutes *System Element Classification, Ontolayering Principle and Knowledge Reuse Scheme* to provide reusable and shareable engineering applications. The detailed description of these constituents is as follows:

System Element Classification

It is developed to capture important characteristics for reducing the growing complexity of information and increasing need to exchange it among various software applications. The classification includes abstract concepts such as Work units, Stages, Work products and Producer as shown in Figure 5.9. Work units constitute tasks or activities that software developers perform, and have a start and end time as well as duration. Subsequently, Stages describes major time frames that help work to provide temporal structure.

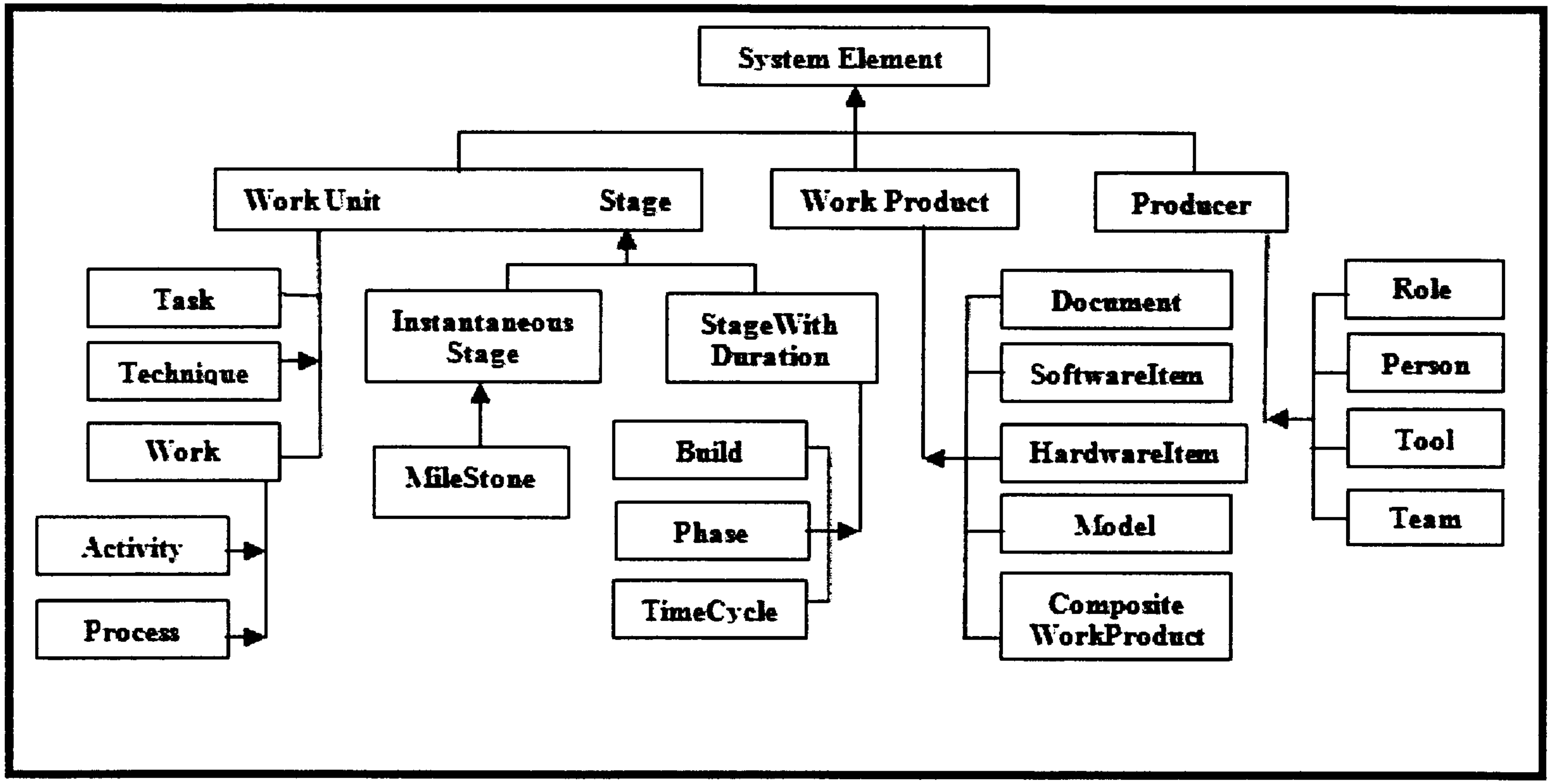


Figure 5.9: Classification of *System Elements*

Next, Work products such as documents or software, are intangible results of performing work units indicates creations and last change times with status. The status of work product may be initial, complete, accepted or approved. Finally, Producers includes people and teams that actually perform work units in order to create work products.

Ontolayering Principle

Ontolayering Principle focuses the ontology in a resource usage manner, specifically by understanding and dissimilating the information comprised by entities. The three prospects namely; Metamodel, Process and Product prospects have been defined around communities that network with ontology as shown in Figure 5.10. Meta-model prospect acts as a common standard determining the other prospect. Meta-model is intended to be used as an origin by method engineers so that the methodologies can be developed. Method engineers typically uses the concepts in meta- model prospect by sub typing and instantiation, thereby creating new concepts (subtype of existing ones) and entities (instances of concepts). All these new concepts and entities created by method engineers are seized to form a Process prospect. Software developers use it by creating the instances of concepts and also, by following the guidance explained by entities. Thus, the instances created by software developers are apprehended to form Product prospect.

Knowledge Reuse Scheme

It starts with formalizing the system element requirements according to representation approach of *System Element Classification.* Subsequently, identification of the related process concepts and entities that need to be revised according to *Ontolayering Principle* retrieved from the knowledge base. Consequently, modification of the producer entities, associated concepts based on *Ontolayering principle* and revision of the influencing product attributes conceded. Finally, simulation of the results is done if the reuse requirements are satisfied otherwise it considers changing a different process concept and entities.

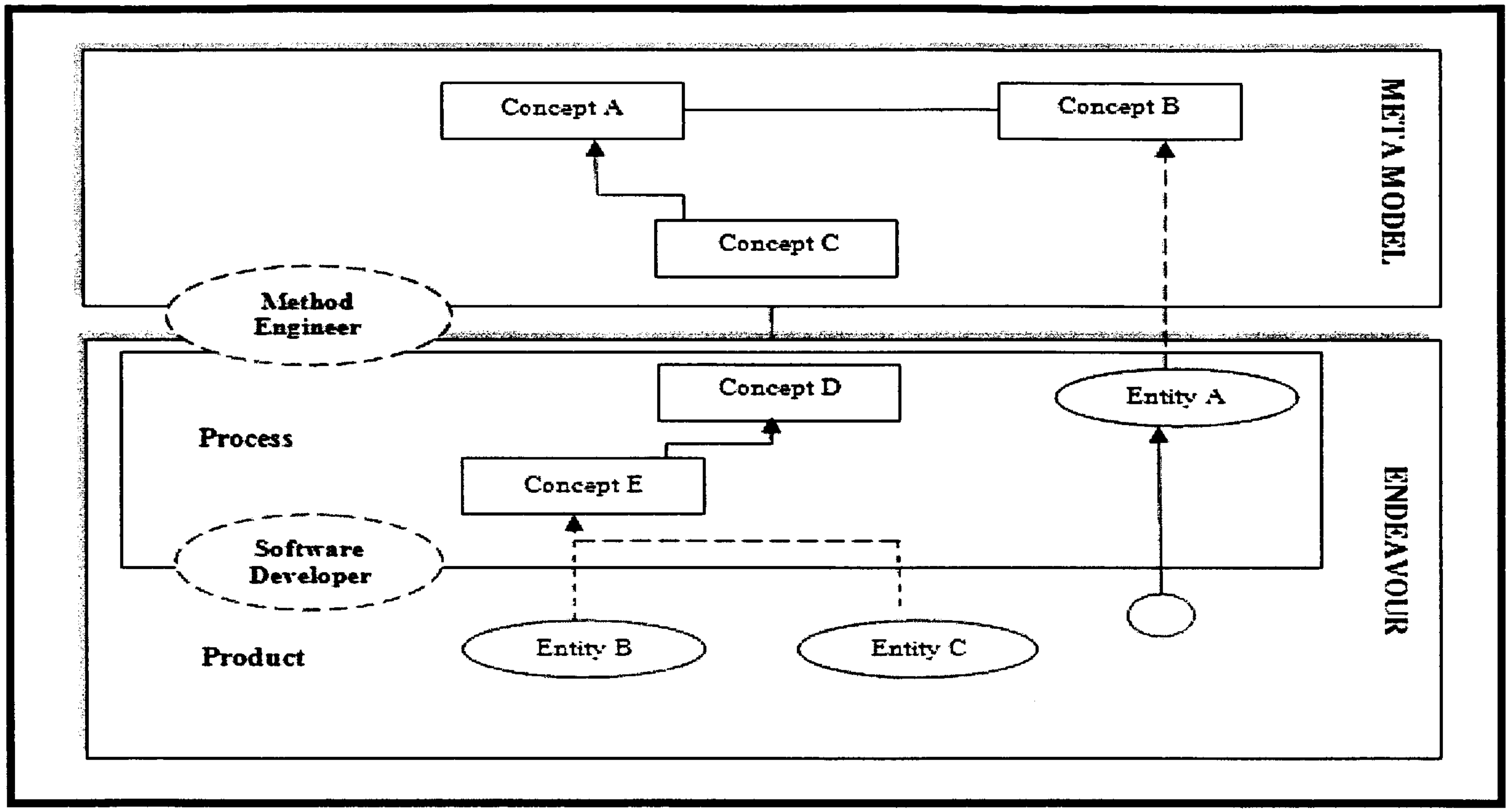


Figure 5.10: Architecture of *Ontolayering Principle*

5.4.2 Ontology Based Reuse Algorithm (*OntoReuseAlgo*)

We propose Ontology Based Reuse Algorithm *(OntoReuseAlgo)* using *Ontological Knowledge Modelling* approach to aid the product design of process plans. It is used to give a uniform representation of the involved information and starts with understanding the system elements. It includes identifying process concepts and entities that need to change followed by altering with *Ontolayering principle* and modify the producer entity and the associated concepts which help in revising the influencing product attributes for simulating the final process plan. Figure 5.11 shows overall procedure of the proposed approach and knowledge reuse strategy in process planning. The stepwise procedure of process planning task is as depicted:

Step I: Formalize the system elements according to the representation approach of *System Element Classification.*

Step II: Identify the related process concepts and entities that need to be added according to *Ontolayering principle* retrieved from the knowledge base.

Step III: Modify the producer attributes and associated concepts based on *Ontolayering principle.*

Step IV: Revise the influencing work product.

Step V: Simulation of the results in step IV. //if the knowledge reuse is satisfied, then shifts to step V; or else, shift to step II and consider changing a different process concept and entities.//

Different systems may use different concepts and terminology to express the same objective while same words may be used to represent different objective by different systems. Both situations hinder information communication. Therefore, step I uses *System Element Classification.* Conversely, process and product concepts that need to change are properly identified and revised through mappings of related *Ontolayering Principle* to certain process and product prospects in step II, III and IV. Lastly, *Knowledge Reuse Scheme* is applied to step V for providing vocabulary and the meaning of the terminology.

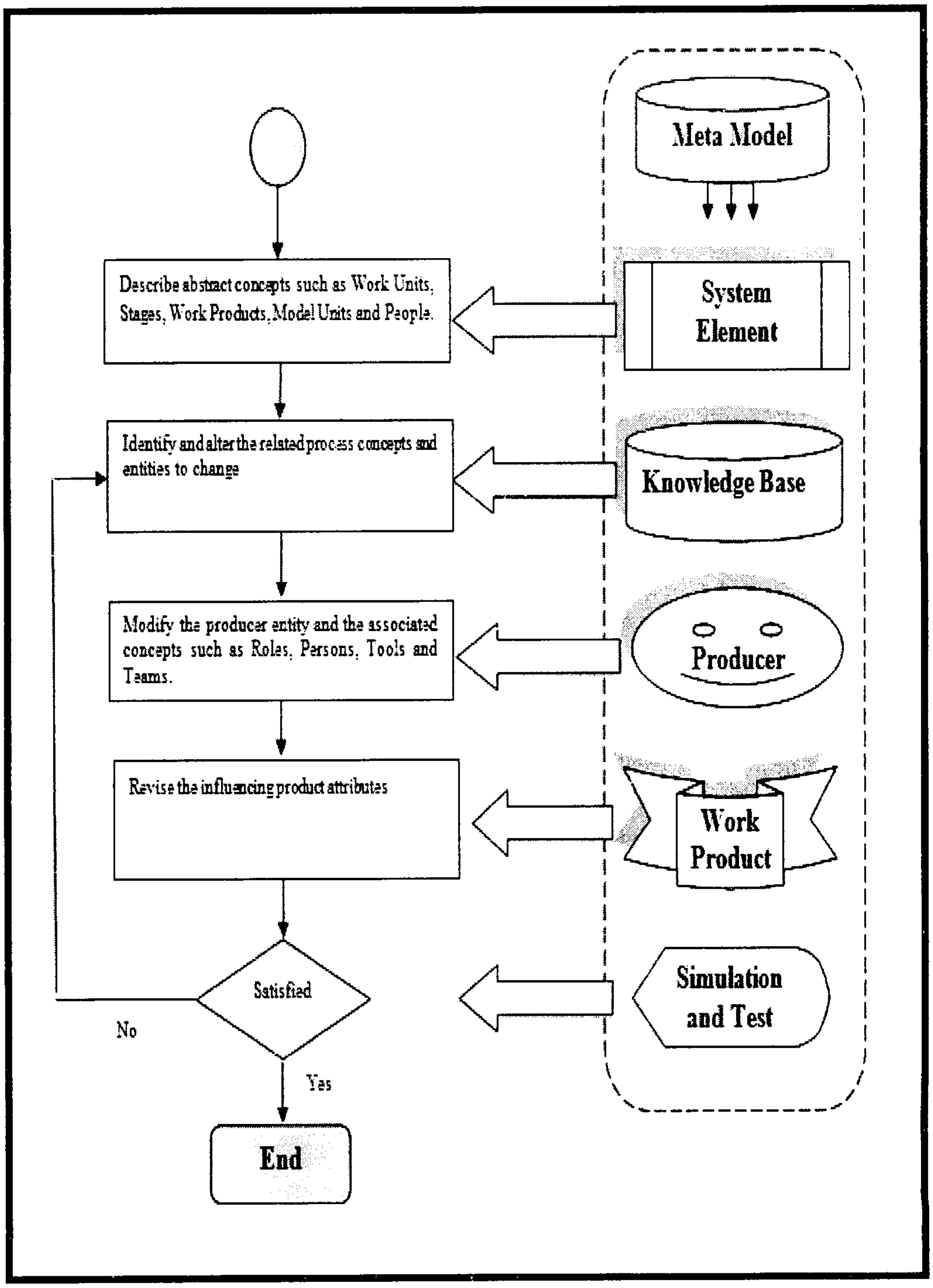

Meta Model
System Element
Knowledge Base
Producer
Work Product
Simulation and Test
Describe abstract concepts such as Work Units, Stages, Work Products, Model Units and People.
Identify and alter the related process concepts and entities to change
Modify the producer entity and the associated concepts such as Roles, Persons, Tools and Teams.
Revise the influencing product attributes
Satisfied
No
Yes
End

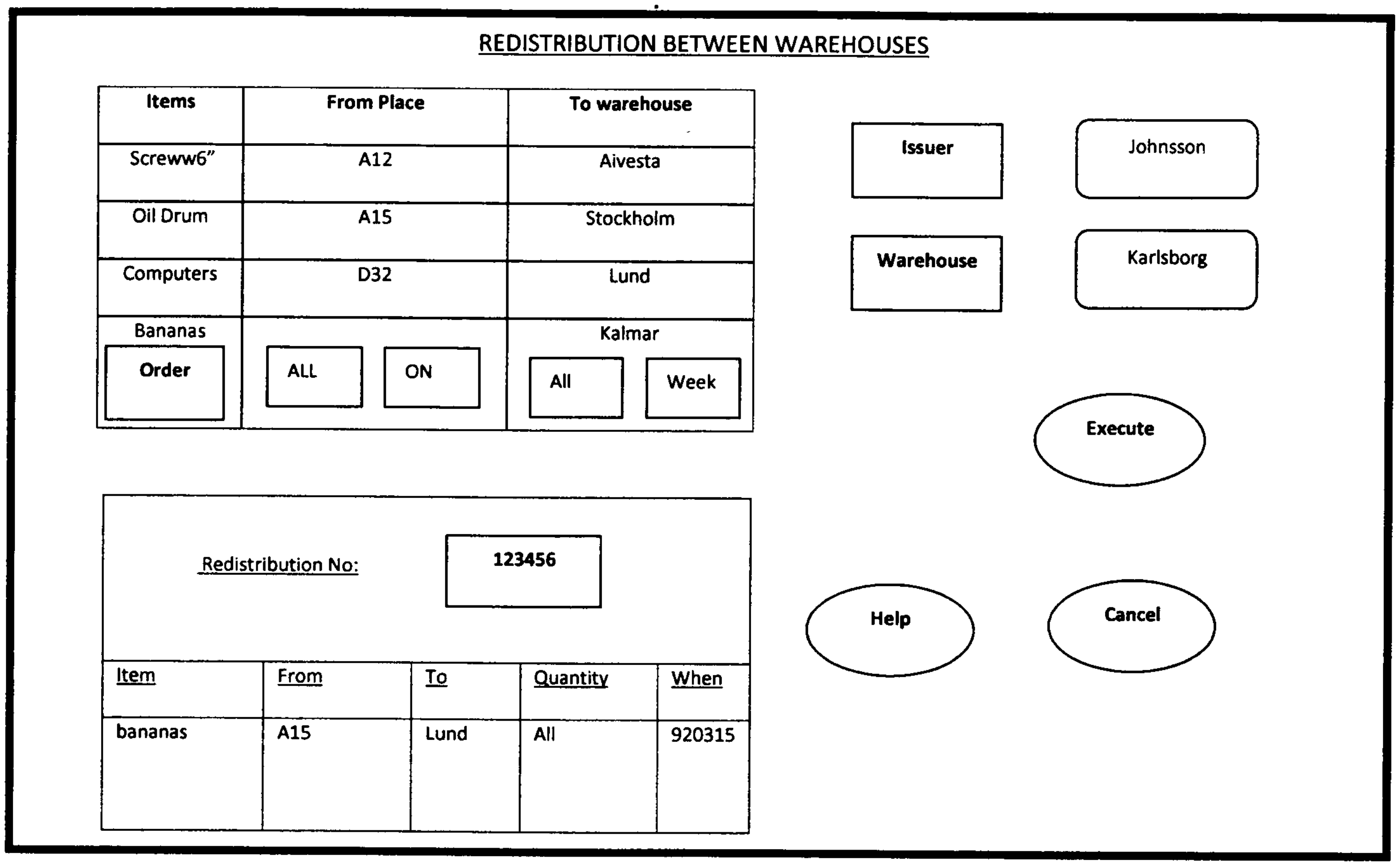
Figure 5.11: Ontology Based Reuse Algorithm (*OntoReuseAlgo)*

5.4.3 Case Study

We describe design of process planning for Inter-warehouse Management System using *OntoReuseAlgo*. The system is responsible for redistribution between different warehouses. The window for redistribution between warehouses is as shown in Figure 5.12. Various people are responsible for carrying out different processes such as foreman is responsible for warehouse management. While, warehouse worker works in a warehouse for loading and unloading. Subsequently, truck driver is accountable for transportation and forklift operator drives a forklift in one warehouse.

Now, on executing first step of *OntoReuseAlgo*, we analyze various system elements such as work units that include request for redistribution, fetching item from warehouse and delivers the item to the new warehouse. Then, producers constitute foreman, warehouse worker, truck driver and forklift operator. Consequently, work product comprised of initialization, loading and unloading as illustrated in Table 5.1.While, during the execution of second step, we identified various process concepts that need to be added. These processes include unexecutable request, wrong redistribution and unavailability of truck as depicted in Table 5.2. Accordingly, step three suggests modification of producer element of step one by including office personnel in it. Official personnel coordinate the transport requests that affect initialization, loading and unloading. Finally, execution of step four recommends a new influencing product attribute termed as planning as shown in Table 5.3. Therefore, revised work product comprised of initialization, planning, loading and unloading.

Lastly, we simulate work product that starts with completion of initialization work unit than planning, loading and unloading. This revised work product eases the redistribution between the warehouses.


REDISTRIBUTION BETWEEN WAREHOUSES

Items	From Place	To warehouse
Screww6"	A12	Aivesta
Oil Drum	A15	Stockholm
Computers	D32	Lund
Bananas		Kalmar

Redistribution No: 123456

Item	From	To	Quantity	When
bananas	A15	Lund	All	920315

Figure: 5. 12: Redistribution between Warehouses Window

Table 5.1: Inter-Warehouse Management System Work Product

S. No.	Work Units	Description
1	**Initialization** (when foreman gives request to do the redistribution)	1. The foreman gives a command for redistribution between warehouses
		2. The window in Figure 5.12 is presented to the foreman
		3. The items can be ordered in a number of ways with ORDER menu such as alphabetical, index, turnover of the items and storing order.
		4. In the 'From place' table we may choose to view either all places in the current warehouse or, if we have selected an item, the place where the item exists
		5. In the 'To warehouse' table we may select all warehouses or the warehouses that we have to transport to this week
		6. The 'Issuer' and 'warehouse' fields are automatically filled when the window pops up.
		7. The foreman selects an item by pointing to it and dragging it to the Redistribution form then selects from which place to take the items and to which warehouse to transport them.
		8. The foreman then gives the quantity to be moved and the date.
		9. It is possible to change the information when the form has been edited. When the foreman EXECUTES the redistribution, the transport is planned. It is also possible to CANCEL the redistribution. Selecting HELP shows window of information about the current window.
2	**Loading** (when truck fetches the item from the warehouse)	1. A Truck driver asks for a transportation request. The request is marked as ongoing.
		2. Give an appropriate request to the Forklift operators to have the items ready when and where the truck is expected.
		3. When the Warehouse Worker gets a request to fetch items at appropriate time, orders Forklift operators to move the items to the loading platform
		4. When the Truck driver arrives the items are loaded. The Truck driver tells the system when the truck is loaded and when it is expected to be at the new warehouse.
		5. Decrease the number of items in this ware house and mark the transport request as on transport
3	**Unloading** (when a truck delivers the items to the new warehouse)	1. When the truck has arrived at the new warehouse, the items are unloaded
		2. The Truck driver tells the system that the transport to this warehouse has been done.
		3. The Warehouse workers receive the items and determine a place for them in the warehouse
		4. Forklift operators are told to move the items to the new place in the new warehouse
		5. When the Truck driver confirms the insertion, the system updates the new place for the items
		6. The transportation time is recorded and stored in the system
		7. The Redistribution and the transport request are marked as performed.

Table 5.2: Identified Processes

S. No.	Process	Description
1	A request is not executable	The execution is interrupted and the Foreman issuing the request is informed
2	Redistribution is wrong	The warehouse place does not have enough items to move The destination warehouse is not appropriate to the item
3	No truck available	When performing loading, and unloading, there may not be any truck available at an appropriate time. Then notify the Foreman who should either delete the request or change it.

Table 5.3: Influencing Product Attribute

S. No	Planning	Description
1	To coordinate transports and issue transport requests	1. When the redistribution is executed the items to be moved are marked as move-pending
		2. Minimize the use of trucks on condition that all delivery dates should be held and the trucks should be compatible with any delivery requirements for the items.
		3. The transport requests are connected to a specific truck's transportation plan.

5.4.4 Benefits of *OntoReuseAlgo*

OntoReuseAlgo aims to improve the knowledge reuse in process planning for software development. It supports the application from three aspects such as *System Element Classification*, *Ontolayering Principle* and *Knowledge Reuse Scheme* for process planning. We have observed the following significant benefits:

- Through organizing and modelling the knowledge towards the characteristics of design processes, unnecessary search time can be avoided on irrelevant knowledge items.
- It allows explicit credentials for analysis and comparison of different domain theories.
- It describes knowledge acquisition approach to structure the entities and relations that need to be acquired in the domain.
- It provides a meta-level view (vocabulary and structure) on their domain which facilitates adequate system documentation and constructs reusable knowledge-system design.
- It can be used to define assumptions that enable knowledge exchange between different users.

5.5 Ontological Reuse (OnR) from O-O Reuse(OOR)

An available reuse methodology such as OOR addresses reusability issue only marginally. Though it mentions the possibility of reusing existing knowledge sources as input for the conceptualization phase, it fails to define precisely knowledge discovery and the subsequent evaluation of candidate knowledge. Also, it describes in detail to build and represent reusable artifacts, but furnish a relatively sketchy recommendation for supporting existing reusable artifacts. Figure 5.13 illustrates more pragmatic process OnR, which exploit OOR process to a maximal extent depending on the particular domain and level of formality. OnR process extremely addresses this issue in the context of knowledge customization/pruning, explicitly extracting relevant fragments from very comprehensive, general purpose ontologies [SS99, REI97]. In addition, OnR provides a detailed description of reuse process and its implications in the overall engineering process.

5.5.1 Mechanism of OnR Development

We propose a generic and incremental process that concentrates on vocabulary of the input sources. And, subsequently inserts additional information corresponding to application needs. OnR process taps the full potential of OOR process from development and representation of reusable artifacts to supporting the reusable artifacts. Figure 5.13 illustrates the mapping of OOR notions analogous to each of OnR notions. The development and representation of reusable artifacts of OOR are concerned with the identification, modelling, cataloging and disseminating a set of software artifacts that can be applied to existing and future software in a particular application domain.

Besides, it encodes the semantics of artifacts in such a way that a user, trying to solve a problem with own knowledge and semantics can locate an appropriate reusable artifact. Therefore, these notions mapped with determining the scope, taxonomy and properties. In OOR, once artifacts are represented, they must be classified and entered into repositories. As such, artifacts are classified in order to indicate the type and relation to other artifacts. There are two well known schemes for doing this repository classification: enumerative and faceted [KG02]. Formerly artifacts have been developed, represented and categorized into repositories; the next concern is to utilize this wealth of information specifically supporting the reusable artifacts. Thus, relates with defining the facets and instances of OnR.

5.5.2 Categorical Comparison of OOR and OnR

On the basis of aforementioned reuse subclasses, we have presented a comparative study to systematically exemplify the object oriented reuse versus ontological reuse as shown in Table 5.4. To begin with, Software Component

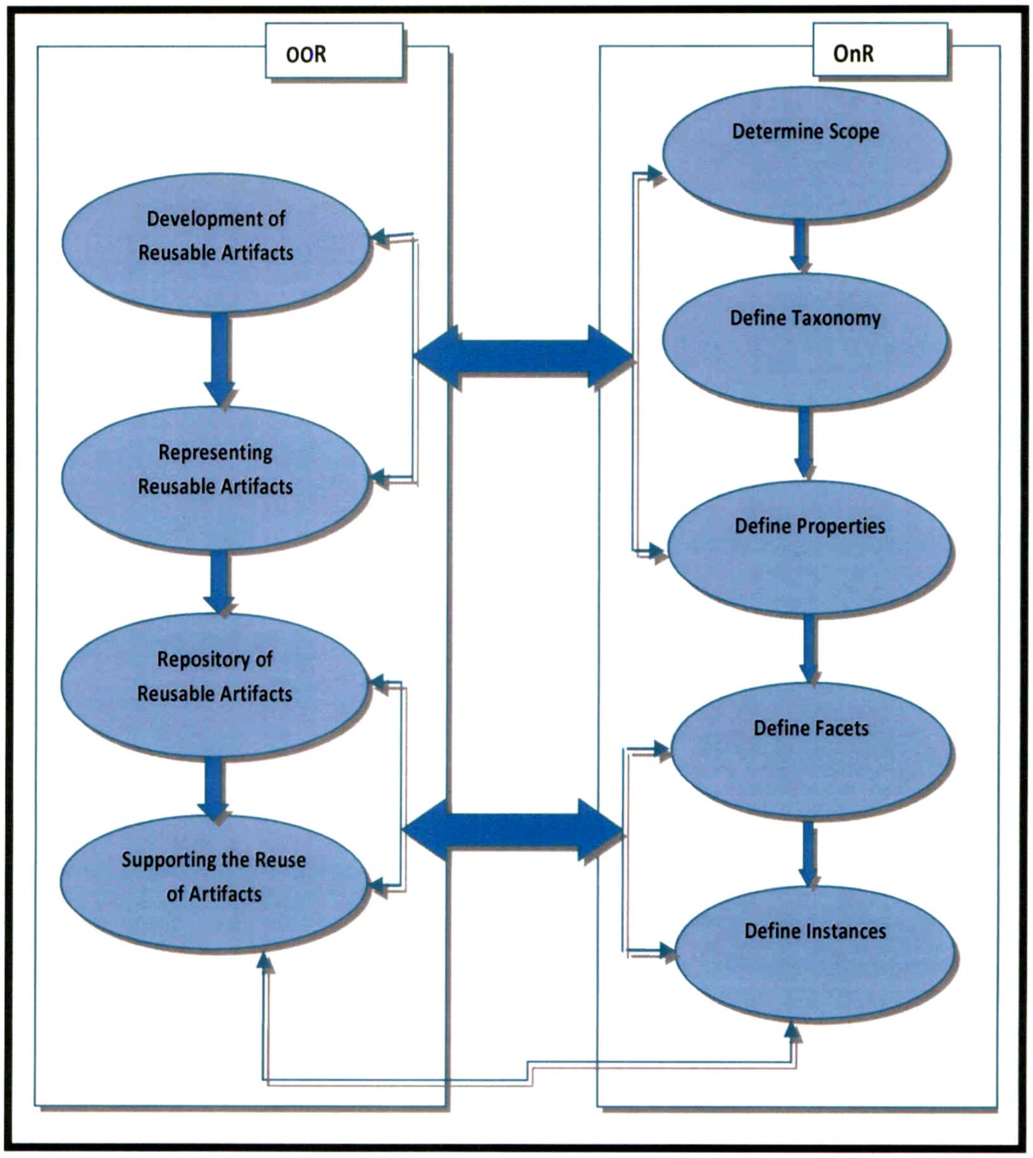

OOR
Development of Reusable Artifacts
Representing Reusable Artifacts
Repository of Reusable Artifacts
Supporting the Reuse of Artifacts
OnR
Determine Scope
Define Taxonomy
Define Properties
Define Facets
Define Instances

Figure 5.13: Object Oriented Reuse to Ontological Reuse

Table 5.4: Object Oriented Reuse vs. Ontological Reuse (Contd ...)

S. No.	Reuse Subclasses	Object oriented Reuse	Ontological Reuse
1.	**Software component reuse**	• Reduce the number of parameters. • Avoid using options in constraints. • Prohibit the direct access to instances. • Implies more cohesive and primitive operations.	• Enables multi faced description of components by lexically analyzed, stored, and indexed using the tokenization and indexing mechanisms. • Generating semantic instances for the concepts and relations. • Allows semantic matching based upon signature-based queries and metadata keyword queries.
2.	**Software architecture/design reuse**	• Identify the system's responsibilities at a given level of abstraction. • Scrutinize the system's environment to produce classes and objects. • Partitioning the class and the object structure into larger units for various attributes and services. • Identify generalization-specialization structure to capture inheritance.	• Allows analysis and comparison of different domain theories. • Structures knowledge acquisition for entities and relations. • Provide a metalevel view (vocabulary and structure) • facilitates adequate system documentation. • Define assumptions that enable knowledge exchange between different agents.
3.	**Software requirement reuse**	• Do domain analysis and prototyping. • Perform incremental development. • Reconcile conflicting sets of requirements. • Build flexibility by regular reviews.	• Defines the optional parts of candidate system. • Models the complex and alternative courses which seldom occur. • Outline separate sub courses which are executed only in certain cases. • Models the situation in which different modes can be inserted.

S. No.	Reuse Subclasses	Object oriented Reuse	Ontological Reuse
4.	**Software process reuse**	• Recognition of objects and operations. • Grouping the objects and operations. • Presents the pictographic description of objects. • Justifying the design decisions.	• Generate implementation for the Knowledge Processor. • Import /export model elements from /to the Repository. • Generate a generic Task from the Ontology. • Find an implementation from a Repository.
5.	**Software technology reuse**	• Provides diagram of each class hierarchy and collaboration for each subsystem. • Specifies contracts supported by each class and subsystem. • Presents computational description of each public object module.	• Based on open-source components and will be published as open-source. • Support interoperability, and standard-based solutions are preferred. • Readymade Tool-level extensibility.
6.	**Software experience reuse**	• Specify a set of basic building blocks for constructing primitive terms. • Define the set of consistent symbol structure states, or changes of states accompanied by interpretation rules and usage guidelines.	• Automating the process of creating application ontologies. • Provide means for visualizing, browsing and exploring domain.

subclass concerned with a search for components that supply the functionality needed by the user. Next, Software Architecture/ Design Reuse subclass claims to be more than just component reuse since it is one of the software elements to be reused during the software process. Software Requirements Reuse subclass is about sharing the requirements across various domains. Software Process Reuse is a kind of reuse deals with the construction of reusable software processes as a means of improving the organization's software process. To investigate the application domains of software technologies the Software Technology Reuse subclass has been devised. Software Experiences Reuse describes the methods that try to reuse every useful experience in software systems development.

5.5.3 Case Study

It is vital to practice OnR in software development for the precise granularity and for a high level of stability as indicated in Table 5.4. OOR practiced on domain level which restrains basic terms of a domain that combined and extended in OnR in order to describe more complex semantics. We consider Hydrology Plant Management System to observe effective use of reuse subclasses during OnR. The system is developed to formalize the concepts associated with it such as water level measurement, water body type and water discharge management. Water level measurement deals with required amount of water to initiate the working of plant and convert the hydro energy into electric energy. It is stated that every water body such as sea, river, pond etc. has a water level and a discharge and these qualities can be observed and managed for its use in plant. This general description provides an entry point for Software Requirement Reuse subclass. While, representation for specific water level measurement service is then sanctified at Software Process Reuse subclass. In particular, sea water level has to be higher than river or pond water level as sea water has more impurities than the others. Therefore, OnR offers a knowledge driven approach for dealing with such diversity.

Once the domain level is settled, components can be added or removed without the need of modifications on domain level which makes the application level highly flexible and consequently referred to Software Component Reuse subclass. The commitment to the same domain level makes OnR comparable with OOR. Also, the task of constructing application lies in the responsibility of the provider of the information source whereas the construction of domain knowledge is a joint effort of domain experts that propose Software Experience Reuse subclass. Additionally, all other peculiarities related to specific water level measurement are described at Software Technology Reuse subclass. This includes legal information to use the provided information and data representation issues. The concepts used to describe the knowledge acquisition and exchange aspects are taken from other types of domain such as measurement or data representation ontologies that constitute Software Architecture/ Design Reuse subclass.

As stated earlier that OOR caters at domain level only. Hence, it enables to extend the features of a particular application using various reuse subclasses and thus, restricted to that application only. For example, while developing Hydrology Plant Management System, OOR helps to add water harvest mechanism for utilization of waste water using reuse subclasses. Whereas, OnR caters at inter-domain level and hence enables to develop different applications with the help of various reuse subclasses. For example, Solar Plant Management System can be developed using OnR that converts solar energy into electric energy using the knowledge of Water Plant Management System. It includes the concepts such as solar power level measurement, solar energy source and release management.

5.5.4 Benefits of Ontological Reuse (OnR)

OnR achieves some lucidness of unclear concepts related with software reuse. A significant aspect of OnR suggests its independence from implementations or technological aspects. OnR allocates various software reuse subclasses with ensuing benefits such as:

- **Cost reduction:** It helps to reduce cost in terms of smaller number of software requirements specification, design, implementation and validation.
- **Higher reliability and quality**: OnR provides higher reliability and quality by components that are tested in previously functioning systems and thus are more reliable than new ones.
- **Risk reduction:** OnR reduces the risk factor as previously existing process implies determines lesser degree of uncertainty a with respect to cost estimation for the project.
- **Accelerated system development:** OnR provides software architecture/ or design reuse that facilitates in shorter development and validation times.
- **Effective use of specialists:** Instead of application specialists doing the same work in different projects, OnR helps these specialists to develop software that encapsulates the associated knowledge.

5.6 Summary

In this Chapter, we have highlighted the importance of reusability in software development process. Chapter starts with depiction of existing reuse subclasses followed by introduction of Object Oriented and Ontological Reuse process. As ontology based reuse is an emerging aspect and specially used for resolving scalability and heterogeneity issues. In this view, we have proposed reusable framework *OntoP4ViewReuse* based on ontology oriented systematic *P4View* approach for reusing. The necessity of *P4View* approach is to make available ontological knowledge that is implicitly tailored to specific application needs. *OntoP4ViewReuse* bring about to apply the ontology of varying levels such as high level, domain, task and application ontology. This cataloging of ontologies is useful for the development of reusable and high-quality software systems. Consequently, we have explored a range of benefits of using *OntoP4ViewReuse.* In addition, to build a common conceptual base characterized by knowledge, Ontology Based Reuse Algorithm *(OntoReuseAlgo)* for process planning has been proposed. It has supported the application through *system element classification, ontolayering principal* and *knowledge reuse scheme.* Also, the significant benefits of *OntoReuseAlgo* have been drawn. In addition, Ontological Reuse (OnR) has been devised from Object-Oriented Reuse (OOR) and effectiveness of OnR has been highlighted with comparative study based on software component, architecture, requirement, process, technology and experience reuse subclasses. Lastly, benefits of OnR have been delineated.

CHAPTER 6

Ontology Oriented Software Reliability Quantification

6.1 Introduction

Software reliability has become progressively more important since the rate of software application crashes grows and as these crashes increasingly impact software performance. Software reliability is defined as the probability of failure-free software operation for a specified period of time in a specified environment [CG07, CMI07]. Hence, it expresses the continuity of correct service delivery. It is a multi-dimensional property including other customer satisfaction factors such as functionality, usability, performance, serviceability and maintainability etc. Also, software reliability is generally accepted as the major factor since it quantifies software failures, which can make a powerful system inoperative. It is found that achieving highly reliable software from the customer's perspective is a challenging task for all software engineers and reliability engineers [GLT98, HM01].

In view of this, various techniques have been proposed to trap the software reliability achievement problems based on software fault lifecycle that includes fault prevention, fault removal, fault tolerance and fault forecasting. Fault prevention is the initial defensive mechanism against unreliability that avoids fault occurrences. But, fault prevention mechanisms cannot guarantee avoidance of all software faults. When faults are injected into the software, fault removal becomes mandatory for protection. It detects fault by verification and validation, and at the same time eliminates them. Next, fault tolerance provides service complying with the specification in spite of faults having occurred or occurring. Lastly, fault forecasting estimates the presence of faults and the occurrences and consequences of failures and becomes main focus of software reliability modelling. Reliability models are typically based on measurement-based models [GPT01, GPM+01]. These models employed in isolation at the later stage of the software development process. On the other hand, early software reliability prediction models are often insufficiently formal to be analyzable and not usually connected to the target system [GPH+05].

Accordingly, there exists need for effective software reliability achievement techniques to improve reliability from product and process aspects that can be certified, generalized and refined. Thus, we introduce Ontology Oriented Reliability (*OnO-Reliability*) development that enhances each phase of Object-Oriented Reliability (OO-Reliability) development. Next, ontology based protocol *OntoReliability* for developing specifications is suggested that leverages reliability analysis in early stages of software development. It is practised by taking into account the meta level structure elicited in the requirements phase. Lastly, we quantify reliability using Ontological Reliability Quantification Method (*ORQM*). This method provides means to accomplish empirical value to reliability of various projects by identifying the architectural styles such as communication, deployment, domain-driven and structure.

6.2 Background

Traditionally, software reliability quantifies software failures, which can make a powerful system inoperative [LYU07]. The study of reliability for the quantification of the operational behaviour of software systems with respect to user requirements is defined as software reliability development [LYU07]. The classic software reliability development process follows the elicitation of four major steps namely; reliability objective, operational profile, reliability modelling and reliability validation. A reliability objective relates to the reliability goal of the software is from the viewpoint of the customer. This reliability objective is related to kind of system failure the user wants to measure. For this purpose, a well defined view of failure classification has been be made. These failures can be permanent, transient, recoverable and corrupting. For each class of failure identified, the reliability requirement then be defined using an appropriate reliability metric. For instance, a system that can recover without operator intervention could have its reliability measured by the intensity that a failure causes inconvenience to the user [SOM01].

The operational profile relates to the information obtained through the system operation on a certain environment. The construction of an operational profile is important in order to select test cases according to the usage of the system [LYU96]. This is related to concept that the software reliability is affected by software failures [SOM01]. Also, the importance of operational profile is endorsed by the fact that software reliability is tightly related to the environment where the software is being executed. In particular, a software component relies on various software and hardware resources to be deployed. Software resources comprise those software elements required to execute a component, such as operating systems, middleware, databases and so on. If during the execution of a software component a resource fails, the component requiring that resource will automatically fail, unless fault tolerance techniques are applied.

Reliability modelling is essential to the reliability prediction and estimation process. Most of the reliability modelling approaches attempt to predict software reliability in the later stages of the life cycle [KM97, KMY91]. The most successful techniques in the literature are probably those classified as Software Reliability Growth Models (SRGM). Those models have been widely used to predict reliability in the later phase of software by modelling the number of faults and the failure rate as testing progresses. As result of those testing, it is expected a growth in the reliability. Some SRGMs can also be used to estimate software reliability by adding other important factors that affect final software quality. The software reliability estimation determines if a product meets its reliability objective and is ready for release [LHC+05]. To carry this out, failure data should be collected during system testing which are then fit into a reliability model. Although a wide number of reliability models can be found, it is sufficient to consider a dozen models, which provide various estimates of software reliability [GUA98]. It is important, though, that the number of tests executed is enough in order to have a reasonable confidence over the estimated values [KM97]. Alternatively, a usage model with a population set that properly characterizes the system can be used [BL75, TRA95]. Using these statistical methods, the best estimates of reliability are obtained during testing [LYU96]. If the reliability objective is not met, more testing will be applied in an iterative process. The last part concerning the software reliability development process consists of the reliability validation. The validation of the system is dependent on the nature of the system. But, in general it consists in monitoring or observing the faults and their consequent failures. Statically, the detected errors can then be eliminated through testing techniques.

6.3 Ontology Oriented Reliability

Basically, reliability accomplishment in software systems is essential to promote software excellence. Object Oriented Reliability (OO-Reliability) development practices are rapidly adopted to address this issue. But, it is observed that OO-Reliability largely depends on conventional testing to validate correctness of system behaviour. Also, it is not adequate to attain the needed reliability for complex systems on account of the intrinsic incompleteness of conventional testing. Therefore, Ontology Oriented Reliability (*OnO-Reliability*) development process is stranded to extend the scope of OO-Reliability development process and we discuss firstly OO-Reliability development process in this section. Then after, we discuss *OnO-Reliability* development process in subsequent section. Also, some attributes related to process, product and resources are identified for comparing OO-Reliability and *OnO-Reliability* development process supported by case study. Lastly, comparative analysis OO-Reliability and *OnO-Reliability* development process is depicted.

6.3.1 OO-Reliability Development Process

Object Oriented Reliability (OO-Reliability) provides expected confirmation to model-based verification techniques applied to software project. The complexity level of these executable models are far less than the equivalent procedure oriented programs to which these models are translated [HM01, BMM99]. Consequently, the following phases of the OO-Reliability reduce the complexity of the system at the implementation level and shown in Figure 6.1.

Abstraction of Implementation Details

It is observed that relationships between objects at analysis level are represented as associations. It constructs state transitions without reference to the internal states of objects. Thus, abstraction of implementation details separate specification of class models and behaviour models separates specification of data from control.

Hierarchical System Representation

Hierarchical system representation support modular designs and encourage software developers to decompose a system into subsystems, derive interfaces that summarize the behaviour of each system, and then perform analysis, validation and verification, using interfaces in place of the details of the subsystems.

Structural Design Rules

Structural design rules is a set of design rules and recommendations that constrain the structural design of system models to conform to space modularity. The systems become space modular when system elements can be analyzed in isolation. It supports existing verification techniques developed for software systems.

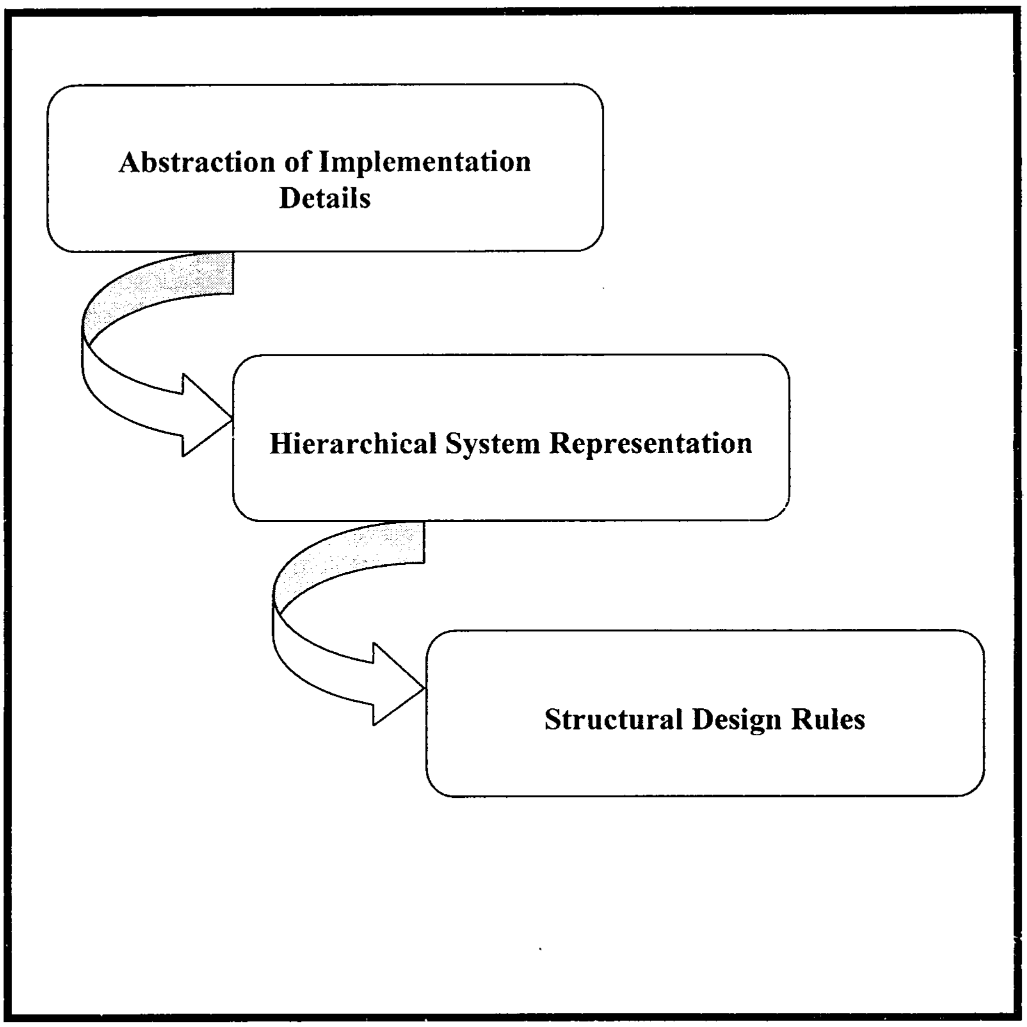

Abstraction of Implementation Details
Hierarchical System Representation
Structural Design Rules

Figure 6.1 Object Oriented Reliability Development Process

6.3.2 *OnO-Reliability* Development Process

We propose Ontology Oriented Reliability (*OnO-Reliability*) development process to explore OO-Reliability since OO-Reliability is not sufficiently proficient to address the issues related to external world representation in the users' intentions. Also, it lacks in the identification of critical domains in the application and request verification of information exchange with respect to these critical domains. *OnO-Reliability* development process commenced reliability with abet of *Onto-self-ensuring recognition ordeal*, *Onto-multiple requests/ confirmation* and *Onto-immunity management routine* phases as illustrated in Figure 6.2. These are discussed as follows:

Onto-Self-ensuring Recognition Ordeal Phase

According to ordeal, the software reliability is gained by complete knowledge representation with respect to all desires and domains that is consistent across users. It includes knowledge developers to web interest users and build ontology content. It controls the changes to the beliefs vigilantly with a meta level structure specifically using ontology for contextual discrimination. Hence, abstraction of implementation details of OO-Reliability development process becomes the subset of it. Thus, it provides an approach of common user building ontology so that specifications of high quality are built and reliability improved.

Onto-multiple Requests/ Confirmation Phase

It identifies critical domains in the application and request verification of information exchange with respect to these critical domains. By incorporating feedback loop of requests and confirmations, variations may be minimized. Also, it helps to generate hierarchy of goals or a single goal and checks the consistency across all subsystems and interfaces in a system. Thus, *Onto-multiple Requests/ Confirmations* intended to investigate potential causes and consequences during system development and consequently improve the hierarchical system representation of OO-Reliability development process.

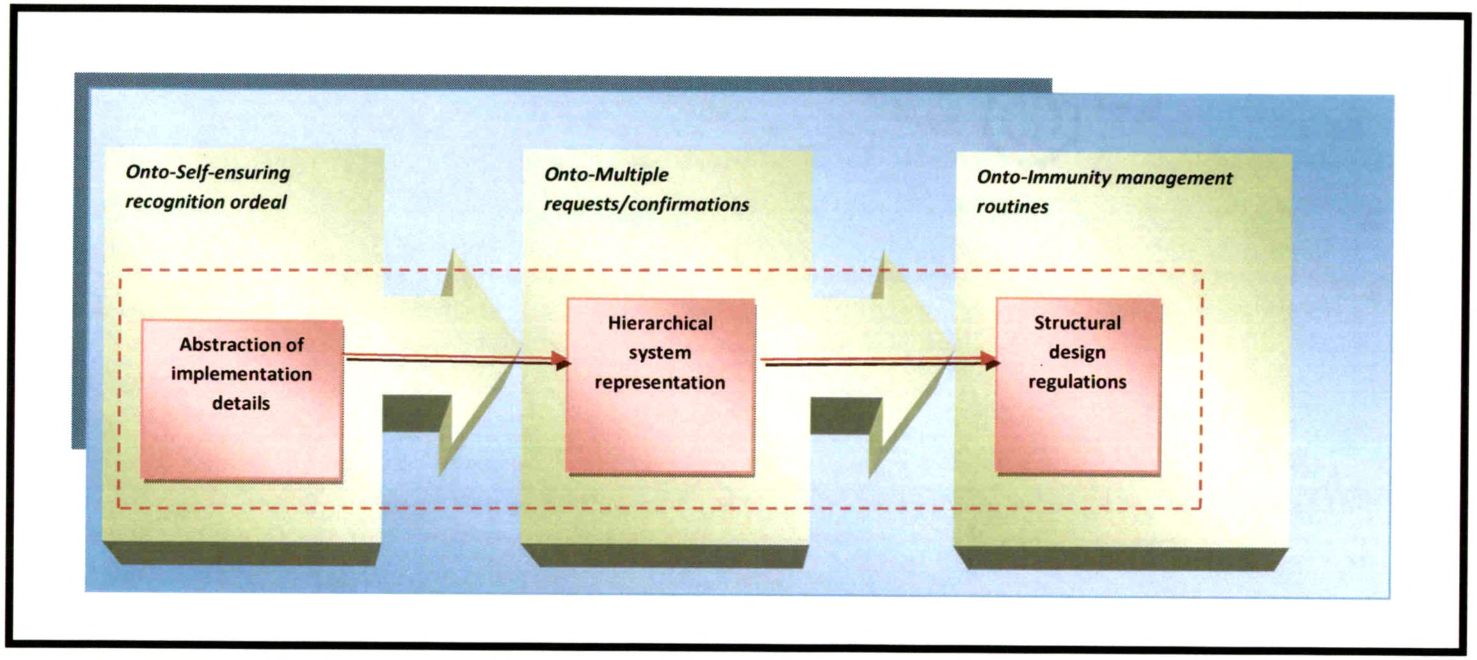

Onto-Self-ensuring recognition ordeal
Abstraction of implementation details
Onto-Multiple requests/confirmations
Hierarchical system representation
Onto-Immunity management routines
Structural design regulations

Figure 6.2 Ontology Oriented Reliability Development Process

Onto-immunity Management Routine Phase

It detects violations and moderates the effectoric action during system execution and as a result, structural design regulations are included to ensure OO-Reliability. *Onto-immunity management routines* offer extension that makes use of alternate group of instructions necessary for system accomplishment. These routines help to acquire establishment of point during the system execution denominated *Ontoimmune* point. This *Ontoimmune* point is responsible for preservation of appropriate information for subsequent improvement. *Ontoimmune points* are said to be active from the moment that are established until the moment in that are discarded. *Onto-immunity management routines* work for the period corresponding to interval between the establishment and discard of *Ontoimmune points.*

6.3.3 Comparison of OO-Reliability and *OnO-Reliability*

Generally, reliability of software system depends on various attributes such as resources, process and product [HM01]. Resource attributes refers to human, reusable software component and environmental resources thus includes user's skills, software development environment. Product attributes are software characteristics that count on software structures thereby comprised of architecture and modelling etc. On the other hand, process attributes constitute phases, activities and resources used during a project. Therefore, these signify software operations, design methodologies and practices etc. Now, we discuss these attributes for OO-Reliability and *OnO-Reliability* as follows:

Resource Attributes

It is observed that resource attributes of the software system are considered as a major factor in reliability achievement. During OO-Reliability achievement resource attributes expresses the user types, external systems, and the system itself restricted to that domain whereas *OnO-Reliability* determines list of user types, external systems, and the system by considering all related domains to acquire knowledge and thus helps in improving the scope of OO-Reliability.

Product Attributes

Product attributes of OO-Reliability such as architecture restricted to view level such as concepts and components required for candidate system while in *OnO-Reliability* focused on style level and thus include communication, deployment, domain and structure based architectures. Next, OO-Reliability refers object modelling and class modelling whereas *OnO-Reliability* follows architecture style modelling.

Process Attributes

Process attributes refer to component consistency checking mechanisms in OO-Reliability while domain consistency checking mechanisms is preferred in *OnO-Reliability*. Subsequently, OO-Reliability follows OOSDLC to achieve reliability whereas OOLC is desired in *OnO-Reliability.*

6.3.4 Case Study and Comparative Analysis

We present a case study to observe the differences between OO-Reliability and *OnO-Reliability* on the basis of resource, product and process attributes. Our case study comprises of Patient Care System (PCS). PCS is used for a reliable document delivery between patient clients and hospitals. The main function of PCS is to transmit health care information between hospitals (service providers) and customers (patients). Thus, it is extremely important that PCS must possess high reliability. While achieving OO-Reliability for PCS, the resource attributes refer to patients as user types, hospital administration and health care information entity sets as system. Next, product attributes signifies doctor information and patient report generation components. Then, process attribute refers to formal verification of components used on the basis of OOSDLC such as report generation as per tests prescribed by doctor. On the other hand, during *OnO-Reliability* achievement, the resource attributes comprised of user types belongs to different domain. Hence, user types include patients, students and guests synonymous to passenger in Railway reservation System, Student Evaluation system and Hotel Management system respectively. Subsequently, product attributes include style based modelling such as PCS belongs to domain oriented architecture style and hence helps in reliability achievement in ontological manner. Lastly, process attributes indicate OOLC for *OnO-Reliability* achievement.

As discussed earlier, OO-Reliability and *OnO-Reliability* is developed with same objective to obtain reliable product. OO-Reliability follows conventional testing to validate reliability of system behaviours. On the other hand, *OnO-Reliability* aims to facilitate knowledge management within internal and external communities and support knowledge acquisition, knowledge storage, and knowledge exchange to ensure reliable product. However, OO-Reliability and *OnO-Reliability* possess equivalences based on their achievement phases. OO-Reliability assumes re-implement the subsystem as an executable specification in the form of an Object-Oriented Analysis (OOA) model. On the other hand, *OnO-Reliability* is set of specifications acquired by common user ontology. OO-Reliability applies model checking to OOA model to validate its behaviour at all possible subsystem of the system. *OnO-Reliability* incorporates feedback loop of user's request and confirmations for system validation. OO-Reliability generates the software system by compilation of the validated and verified OOA model whereas *OnO-Reliability* attempts to define consistency checking mechanism for software system. Thus, it is analysed that use of ontology improves the reliability to an optimum extent. *OnO-Reliability* includes all possible resource, product and process attributes across all inter related domains to check the uniformity and constancy. Hence *OnO-Reliability* helps in OO-Reliability advancement.

6.4 Ontological Specifications

Reliability accord scheduled prior to software development is attracting a growing attention among software engineers and reliability experts. Software specification decisions have a direct impact on system aspects such as overheads, time-to-market, and quality [MUS04]. This consideration results in software reliability accord in the phase of software specification development. In this perspective, reliability of a software system is defined as the probability that a system will perform as per its specifications [PHA00, MH06]. In addition, all required characteristics of the software to be implemented are determined by specifications. Thus, it becomes the starting point of any software development process. Specifications are also an important means of communication between users and developers. Since ontology is the explicit specialization of conceptualization that describes domain knowledge, is widely applicable in software reliability engineering [WS00]. Consequently, *OntoReliability* protocol is proposed to serve as an evolutionary approach for software reliability advancement and is discussed in detail in subsequent section. It describes concepts that are endorsed by users, during specification phase in software engineering. Accordingly, faulty knowledge representation is incised and the approved knowledge by most users can be acquired.

6.4.1 *OntoReliability* Protocol

We propose *OntoReliability* protocol for developing software specifications in order to improve the software reliability as shown in Figure 6.3. For an immediate reflection of the consequences of the specifications and for an early substantiation, specifications must be more accomplishable. *OntoReliability* protocol makes specifications an optimal communication mode between users and developers for the intended system behaviour discussion. It integrates five specifications ranging from *OntoRelSpecifications1* to *OntoRelSpecifications5*. We discuss these specifications individually along with their components now.

Layer I-OntoRelSpecifications1

It describes complete knowledge representation with respect to all desires that must be consistent across all domain users. Therefore, *OntoRelSpecifications1* are composed of description, preconditions and post conditions. Description is the characterization of user task that comprehensively defines the intended purpose and environment for software under development. It fully describes the user needs that a software is expected to perform. Subsequently, precondition is a condition or predicate that must always be true just prior to the execution of some operation in a formal specification. If a pre-condition is violated, the effect of the particular section becomes undefined and thus may or may not carry out its intended work. Security problems can arise due to incorrect preconditions. Thus, preconditions are proposed in order to get acquaintance of the existing environment autonomy, constraints and controls. Later, the post conditions are mentioned to envisage the consequences of specification implementation. It is a statement or statements describing the condition that is true when the operation has completed its task. If the operation is correct and the pre-condition(s) met, then the post-condition is guaranteed to be true.

Later II-OntoRelSpecifications2

Layer II signifies the predicted process execution in advance along with the alternate completing approach, if obligatory. Standard courses define the requirements for any data or initialization sequences that are specific to a given site, mission or operational mode. On the other hand, proxy courses specify the site or mission-related features that should be modified to adapt the software to a particular installation. Various attributes are taken into account while designing the standard and proxy courses. Firstly, system does not debilitate. Next, system may undergo several updates during the life cycle and system fixes may introduce new problems. Then, software testing is usually incomplete due to encounter of large number of states. In addition, standard and proxy courses specify the normal and special operations required by the user such as various modes of operations in the user organization. Subsequently, periods of interactive operations and periods of unattended operations as well as data processing support functions such as backup and recovery operations are also included.

Layer III-OntoRelSpecifications3

Layer III illustrates the occurrence of the exception is assumed to be immediately captured and can be automatically stored in exception repository according to different classifications. These anticipated exceptions includes as service unavailability, deadline expiry, external trigger and rationality violation. In a software process, the services binding it evolve autonomously and their coupling is highly loose. Since, software processes usually have long-running duration, some services may be invalid or the executing results of some services are not the same as anticipated and thus termed as service unavailability. Next, deadline expiry occurs due to incapability of monitoring the execution details of each service. Deadline for service execution in a software process is specified to indicate when the execution should be completed. Then, external trigger triggers to a process execution and often used as a means of signalling the occurrence of an event that impact on the process and requires some form of handling. Although triggers can be anticipated at design time, it is not predictable if or when such triggers will occur. For this reason, they are ideally suited to resolution via exception handling. Lastly, rationality violation compose of the software process which is bound with services must be rational. But it is not easy to ensure the rationality of the process at design phase.

Layer IV-OntoRelSpecifications4

These specifications specifically focused on inclusions, primacy, and rate of uses. The inclusions define the features that must be mentioned for ease of users such as confining the system decision information, authentication requisitions, and security check services. Subsequently, primacy determines the relative necessity of requirements. Whereas, all requirements are mandatory but some are more critical than others such as *High, Medium* and *Low*. These values have been set to remove the off beam comprehension on the basis of users' evaluation for particular functionalities. In addition, proper prioritization of requirements provides schedule modification, improved customer satisfaction and lowers the risk of cancellation. Next, rate of use indicates the unit of measurement for specification usage.

Layer V-OntoRelSpecifications5

These specifications comprised of exceptional requirements and remarks and concerns. The exceptional requirements specify all the software requirements at a level of detail sufficient to enable designers to design a system to satisfy those requirements. Every stated requirement should be externally perceivable by users, operators, or other external systems. These requirements should include at a minimum a description of every input into the system, every output from the system and all functions performed by the system in response to an input or in support of an output. In addition, notes and issues contain information of a general or explanatory nature that may be helpful, but is not mandatory for intended use, definitions used and abbreviations used. Also, indicates changes from previous issue but not applicable for the initial issue. Revisions shall identify the method used to identify changes from the previous issue in order to sustain the trustworthiness.

6.4.2 Case Studies

In this section, experimentation has been conceded using *OntoReliabilty* protocol, to produce the software specifications for different applications of various domains. It includes description, preconditions, post conditions, normal flow, alternative flows, exceptions, inclusions, priority, frequency of uses, special requirements and notes and issues. We have presented two case studies in this section.

Case I- Airline Flight Reservation System (AFRS)

AFRS is a web-based application that can accept client requests, list searched results, process booking, payment, modification and cancellation to existing reservations. Users do not have to personally go to the counter or contact airline representatives, but only access AFRS through any browser to book their flights. Customers have internet access to AFRS internet-based user interface to book their flights, prefer any date and time, favor any airline as well as various demands such as arrival time, flight class or non-stop whereas an administrator is managing AFRS's back-end databases. Administrator may wish to add/delete any information in existing reservation or user registration database. Administrator may needs to create and manage temporary views of fetched records from the databases of airlines and airports. We have developed the specifications for placing, change/ cancel reservation and add/ delete flight information/ user reservations using *OntoReliabilty* as shown in Table 6.1, 6.2 and 6.3 respectively.

Table 6.1 Specifications for Placing Reservation

OntoRelSpecifications1	Description	A customer accesses the AFIRS from the Internet, optionally search for specific ticket/flight information of interest, selects ticket(s), and places reservation.
	Preconditions	None
	Post conditions	Database of available tickets is updated to reflect items in this order. Remaining tickets number is updated.
OntoRelSpecifications2	Standard course	**1.0 Order a Single Ticket** • Customer uses the web interface to enter a certain query to view flight information for a specified interest. • System displays available flight information satisfied the query. • Customer selects one or more items from page. Customer can also click on a particular ticket to see the detailed information. • System displays reservation with detailed price information including all taxes. • Customer confirms reservation or requests to modify reservation (back to step 3). • Customer specifies payment method. • Customer indicates that reservation is complete. • System confirms acceptance of the order. • System sends Customer an e-mail confirming order details, price, and additional links to access the ticket details or for potential modification. • System stores order in database, and updates available ticket information (database).
	Proxy course	**1.1 Order multiple tickets** (branch after step 8) • Customer asks to place another reservation. • Return to step 2. **1.2. Order the Last minute deals** (after step 2) • Customer orders the daily special from the menu. • Return to step 5.
OntoRelSpecifications3	Exceptions	**1.0.E.1 Concurrent access from multiple users (when there is less available ticket than potential users, demand surpass supply)** (at step 1) • System informs Customer that ticket no longer available. • 2a. Customer cancels the ticket order. • 2b. System terminates. • 3a. Customer requests to select another ticket. • 3b. System restarts. **1.0.E.2 Cutoff time for available ticket (the cutoff time is usually 5 hours before the departure time of the flight)** (at step 1) • System informs Customer that the cutoff time policy occurs. • 1a. System denies the access to the particular ticket information terminates. **1.2.E.1 the user input query is not reasonable (e.g. departure time is behind arrival time)** (at step 1) • System informs Customer of right form of query to input. • Customer changes query.
OntoRelSpecifications4	Includes	None
	Primacy	High
	Rate of Use	Approximately 400 users, average of one usage per day
OntoRelSpecifications5	Exceptional Requirements	Customer shall be able to cancel the order at any time prior to confirming the order. Customer shall be able to view all tickets he reserved within the previous six months. (Priority = medium)
	Remarks and concerns	The default time zone of departure/arrival information is the local time zone of specific city. If customer doesn't need to have an account until reservation is placed.

Table 6.2 Specifications for Change/ Cancel Reservation

OntoRelSpecifications1	Description	Customers who have reservations in AFIRS should be able to modify or cancel these reservations before a certain cutoff time.
	Preconditions	Customer is logged into AFIRS.
	Post conditions	Customer has placed certain actions on existing reservations.
OntoRelSpecifications2	Standard course	**2.0 Reservation modification or cancellation** • Customer requests to change or cancel reservation. • System invokes Authenticate User's Identity. • System verifies Customer's identity and provides the login view menu for customer. • Customer clicks on the reservation section and chooses one of the reservations to modify or cancel. • Customer confirms desire to do modification or cancellation. • System checks the cutoff time and permit the modification/cancellation requested by customer. • System asks Customer to confirm his or her decision. • System sends corresponding update information to the database of ticket/flight information. • System informs Customer the change and provides confirmation number of the transaction.
	Proxy course	None
OntoRelSpecifications3	Exceptions	**E.1 Customer identity authentication fails** (at step 2) • System gives user two more opportunities for correct identity authentication. • 2a. If authentication is successful, Customer proceeds. • 2b. If authentication fails after three tries, System notifies Customer, logs invalid authentication attempt, and terminates. **E.2 The cutoff time policy is applied** (at step 6) • System informs Customer that he cannot make the modification/cancellation and explains why. • System terminates.
OntoRelSpecifications4	Includes	Authenticate User's Identity
	Primacy	High
	Rate of Use	Once per user on average
OntoRelSpecifications5	Exceptional Requirements	User authentication is performed per corporate standards for medium-security applications.
	Remarks and concern	Expect low frequency of executing this use case. But relatively high frequency during the season (Christmas)

Table 6.3 Specifications for Add/ Delete Flight Information and User Reservations

OntoRelSpecifications1	Description	The Administrator may modify the flight information and prices for a specified date to reflect changes in availability or prices or to define last minute deal. Administrators can also Add/Delete User Reservations in some cases.
	Preconditions	Database already exists in the system.
	Post conditions	Modified database has been saved.
OntoRelSpecifications2	Standard course	**Update/Add/Delete Flight information/User reservations** • Menu Manager requests to view the menu for specific ticket/flight information. • System displays the menu. • Menu Manager modifies the menu to add new information, remove or change items, create or change deal, or change prices, number of seats available etc. (invoke the database management language module through interface) • Menu Manager requests to save the modified menu. • System saves modified menu. • If the change is about user reservations, send notification to users by e-mail
	Proxy course	None
OntoRelSpecifications3	Exceptions	**E.1 No item exists for specified information** (at step 1) • System informs Administrator that no menu exists for the specified date. • System asks Administrator if he would like to add a new item. • 3a. Administrator says yes. • 3b. System invokes Database interface. • 4a. Menu Manager says no. • 4b. System terminates. **E.2 Item specified is the past information** (at step 1) • System informs Administrator that the item requested cannot be modified. • System terminates.
OntoRelSpecifications4	Includes	None
	Primacy	High
	Rate of Use	Approximately 20 times per week by one user
OntoRelSpecifications5	Exceptional Requirements	The Administrator may cancel out of the modification function at any time. If any item has been changed, the system shall request confirmation of the cancellation.
	Remarks and concern	If the Administrator is doing modification of certain information, that information should be temporally invisible/ inaccessible for customers.

Case II- Web Accessible Alumni Database System (WAADS)

WAAD encompass numerous files and information from the Alumni Database, as well as files on the department server system. This system is completely web-based, linking to WAAD and the remote web server from a standard web browser. WAAD operated from the departmental server and connects Alum to the University Web Server. University Web Server passes Alum to the Departmental Server. The Departmental Server then interact with Alumni database and allows to transfer data to and from a database. The system will consist of Alumni Home page with five selections. The first selection is to fill out a survey. The questions on the survey will be created by a designated faculty member. The survey asks Alum, questions concerning their degree, job experience, how well their education prepared them for their job. This information will be retained on the departmental server and an e-mail will be sent to the designated faculty member as shown in Table 6.4. The second selection is to the Entries section as illustrated in Table 6.5. There are two choices on this page. One choice is to add a new entry. A form is presented to the Alum to be filled in. Certain fields in the form will be required, and list boxes will be used where appropriate. A password typed twice will be required of all new entries.

Third selection of the Entries page is to update an Alum entry as depicted in Table 6.6. A form presented allowing the Alum to enter the year of graduation and then to select themselves from a list. A password is required before the information presented to the Alum to be updated. The fourth selection is to search or e-mail Alum as shown in Table 6.7. A form is presented requiring the requested Alum's year of graduation. The requesting Alum search a table to see if the requested Alum is in the database, and if so non-sensitive information be returned. At this time, Alum can select to e-mail the Alumnus or search for another Alumnus. If Alum chooses to e-mail the Alumnus, a form is presented for the message to be entered with the sending Alum's name and e-mail. The message with all necessary information forwarded to the requested Alum. The e-mail address of the requested Alum will not be seen by the sending Alum as a privacy measure. All pages will return the Alum to the Alumni Home Page.

Table 6.4 Specifications to Access Alumni Home Page

OntoRelSpecifications1	Description	The Departmental Web Server is waiting on an Alum to connect
	Preconditions	Alum is connected to the Internet and on the Home Page
	Post conditions	The Alum is on the Alumni Home Page
OntoRelSpecifications2	Standard course	• The Alum connects to the University Web Server. • The Alum selects the Alum link on the Home Page. • The University Web Server passes the Alum to the Alumni Home Page.
	Proxy course	• None
OntoRelSpecifications3	Exceptions	If there is a connection failure the Departmental Server returns to the wait state
OntoRelSpecifications4	Includes	Alum authentication
	Primacy	High
	Rate of Use	Approximately 100 users, average of one usage per day
OntoRelSpecifications5	Exceptional Requirements	University Web Server sends the Alum to the Departmental Server. Departmental Server presents the Alum with the Alumni Home Page.
	Remarks and concern	None

Table 6.5 Specifications of Survey

OntoRelSpecifications1	Description	The Alum chooses to fill out a survey
	Preconditions	The Alum is connected to the Internet and on the Alumni Home Page
	Post conditions	The survey record is created in the Survey Table of the Alumni Database.
OntoRelSpecifications2	Standard course	• The Departmental Server presents the Alum with a form. • The Alum fills in the form and click submit • The Departmental Server checks to see if all required fields are not empty. • If the required fields are not empty, the Departmental Server creates a new record then in Survey Table of the Alumni Database. • If any of the required fields are empty, the Departmental Server returns a message and returns the Alum to the Survey form. • The Departmental Server returns the Alum to the Alumni Home Page
	Proxy course	• None
OntoRelSpecifications3	Exceptions	If the connection is terminated before the form is submitted, the fields are all cleared and the Departmental Server is returned to the wait state.
OntoRelSpecifications4	Includes	Alum authentication
	Primacy	High
	Rate of Use	Approximately 100 users, average of one usage per day
OntoRelSpecifications5	Exceptional Requirements	None
	Remarks and concern	None

Table 6.6 Specifications to Create New Entry

OntoRelSpecifications1	Description	The Alum chooses to create a new entry on the Entries page
	Preconditions	The Alum must be connected to the Internet and on the Entries page.
	Post conditions	A record is created in the Alumni Table of the Alumni Database.
OntoRelSpecifications2	Standard course	• The Alum clicks on add a new entry. • The Departmental Server returns a form. • The Alum fills in the form and clicks submit. • The Departmental Server checks to see if any required field is empty. • If any required field is empty the Departmental Server will send a message and return the Alum to the new entry form page. • If no required field is empty the Departmental Server will create a new record in the Alumni Table in the Alumni Database, and return the Alum to the Alumni Home Page. • The Alum may select Cancel. • If the Alum selects Cancel, the form is cleared and the Alum is returned to the Alumni Home page.
	Proxy course	• None
OntoRelSpecifications3	Exceptions	• If the connection is terminated before the form is submitted, the fields are cleared and the Departmental Server is returned to the wait state. • If the connection is terminated after the form is submitted, but before the Alum is returned to the Alumni Home Page, the record is created in the Alumni Table of the Alumni Database.
OntoRelSpecifications4	Includes	None
	Primacy	High
	Rate of Use	Approximately 100 users
OntoRelSpecifications5	Exceptional Requirements	None
	Remarks and concern	None

Table 6.7 Specifications to Update an Entry

OntoRelSpecifications1	Description	The Alum chooses to update an existing entry in the Alumni Database
	Preconditions	The Alum must be connected to the Internet and on the Entries Page.
	Post conditions	The record in the Alumni Table of the Alumni Database has been updated and the Alum is returned to the Alumni Home Page.
OntoRelSpecifications2	Standard course	• The Alum clicks on update an entry link. • The Departmental Server returns a form. • The Alum enters his/her year of graduation. • The Departmental Server queries the Alumni Database for that particular year and returns a table of all graduates from that year in a form with radio buttons and requesting their password. • If the password does not match the Departmental Server returns a message and allows the Alum to try again. • If after 3 tries the password does not match, the Departmental Server will return a message telling the Alum to contact the designated faculty member to receive their password. • If the password matches go to 8. • The Departmental Server returns a form with the data for that Alum in it and a message to update the data they wish and click submit. • The Departmental Server with replaces the old data with the new data and returns the Alum to the Alumni Home Page.
	Proxy course	• If after three attempts to match the name and password the Departmental Server will return a message and block the Alum from the update section.
OntoRelSpecifications3	Exceptions	• If the connection is terminated before the form is submitted, the fields are cleared and the Departmental Server is returned to the wait state. • If the connection is terminated after the form is submitted, but before the Alum is returned to the Alumni Home Page, the record in the Alumni Table of the Alumni Database is updated and the Departmental Server is returned to the wait state
OntoRelSpecifications4	Includes	None
	Primacy	High
	Rate of Use	None
OntoRelSpecifications5	Exceptional Requirements	None
	Remarks and concern	None

Table 6.8 Specifications for Searching an Alumni/ E-mail an Alumni Entry

OntoRelSpecifications1	Description	The Alum chooses to search/e-mail Alum.
	Preconditions	The Alum is connected to the Internet and on the Alumni Home Page.
	Post conditions	The Alum receives the information on the requested Alum, receives e-mail confirmation message, or is returned to the Alumni Home Page
OntoRelSpecifications2	Standard course	• The Alum clicks on e-mail an alumni link. • The Departmental Server returns a form. • The Alum fills in the form and clicks submit. • The Departmental Server checks to see if any required fields are empty. • If any required fields are empty the Departmental Server returns a message and the form. • If none of the required fields are empty the Departmental Server queries the Alumni Database for the requested Alum's entry. • The Departmental Server returns the non-private information on the requested Alum and a message stating if the requested Alum will accept e-mails. • If the requested Alum is not in the Alumni Database, the Departmental Server returns a message and the Alum is returned to the Home Page. • If the requested Alum will accept e-mails, the Alum can select E-mail this Alum. • If not the Alum can select Search for another Alum or return to Alumni Home Page. • If the Alum chooses to Search for Alum go to step 2. • If the Alum selects return to Alumni Home Page the Departmental Server returns the Alum to the Alumni Home Page. • The Departmental Server presents the Alum with a form to fill out and a place for the message. • The Alum selects send. • The Department Server will forward the e-mail with all necessary information to the requested Alum. • The Departmental Server returns a message and returns the Alum to the Alumni Home Page
	Proxy course	• None
OntoRelSpecifications3	Exceptions	• If the connection is terminated before the information is returned, the Departmental Server is returned to the wait state. • If the connection is terminated after the information is returned, the Departmental Server is returned to the wait state
OntoRelSpecifications4	Includes	None
	Primacy	Medium
	Rate of Use	None
OntoRelSpecifications5	Exceptional Requirements	None
	Remarks and concern	None

6.4.3 Benefits of *OntoReliabilty* protocol

The following statements recapitulate the use of *OntoRelSpecifications* developed using *OntoReliabilty* protocol.

- *OntoRelSpecifications* allow demonstrating the behavior of a software system before it is actually implemented. These reflect three positive consequences for software development. Firstly, executable components are much earlier available than in the traditional life-cycle. Therefore validation errors can be corrected immediately without incurring costly redevelopment. Next, requirements that are initially unclear can be clarified and completed by hands-on experience with the executable specifications. Then, execution of the specification supplements inspection and reasoning as means for validation. This is especially important for the validation of non-functional behavior.
- *OntoRelSpecifications* are constructive, these explicitly do not only demand the existence of a solution, conversely actually construct it.
- *OntoRelSpecifications* do not necessarily constrain the choice of possible implementations because only minimal design and implementation decisions are necessary to get executability. In addition, these decisions are revisable.

6.5 Software Reliability Quantification

Software reliability quantification plays a very significant role for software consistency and excellence. However, the conventional software quantification method mostly focuses on evaluation by use of failure data which is gained only after testing or usage in the late phase of the software life cycle. We use ontology to obtain and quantify the software reliability with the help of software architecture style. Ontology allows developers and users to better understand software architecture and reliability terminologies. Therefore, an Ontological Reliability Quantification Method *(ORQM)* is instigated that focuses on various software categories correlative with architecture style and concerned class of project parameters. Now, we describe these parameters with terms required for reliability quantification, *ORQM* and case studies to demonstrate the viability of this method.

6.5.1 Terminology

Our Ontological Reliability Quantification Method (*ORQM*) use some standard terms along with some new terms needed for reliability quantification. We discuss these terms with the suitable examples in this section.

Project Category (*PC*)

Project Category is defined on the basis of high level patterns and principles commonly used for application development. For example, these categories may include communication, deployment, domain and structured etc.

It is observed that these PCs are referred to various architectural styles. An architectural style is a set of principles that provides an abstract framework for a family of projects. Architectural styles can be organized by their key focus area. Table 6.9 lists the major areas of focus and the corresponding architectural styles. Communication category comprised of service-oriented architecture and message bus styles. Moreover, client/server and *n*-tier architecture styles are included in deployment category and domain driven design is included into domain category. Lastly, structure category constitutes component-based and layered architecture.

Project Parameters

Individual project attributes that affect reliability quantification in a project are known as project parameters.

For example, domain alignment, abstraction and interoperability etc. may be considered as project parameters since these affect the run time behaviour of project. Other parameters such as autonomous, distributable, authentication and authorization etc. also play critical role in key design principles and centralized implementation. Thus, we have classified project parameters affecting reliability in three classes mainly; quality attributes, devise ideologies and crosscutting concerns and described these classes as follows:

Table 6.9: Various Project Categories with Architecture Styles

Category	**Architecture style**	**Description**
Communication	Service-Oriented Architecture	Refers to applications that expose and consume functionality as a service using contracts and messages.
	Message Bus	Prescribes use of a software system that can receive and send messages using one or more communication channels, so that applications can interact without needing to know specific details about each other.
Deployment	Client/Server	Segregates the system into two applications, where the client makes requests to the server. In many cases, the server is a database with application logic represented as stored procedures.
	N-Tier / 3-Tier	Segregates functionality into separate segments in much the same way as the layered style, but with each segment being a tier located on a physically separate computer.
Domain	Domain Driven Design	Focused on modelling a business domain and defining business objects based on entities within the business domain.
Structure	Component-Based Architecture	Decomposes application design into reusable functional or logical components that expose well-defined communication interfaces.
	Layered Architecture	Partitions the concerns of the application into stacked groups.

Quality Attributes (Q)

It is defined as the overall factors that affect run-time behavior, system design and user experience. There exist various kinds of quality attributes depending upon various project categories. Q_i, $i=1..., l$; represent l kinds of quality attributes.

For example, maximum number of communication oriented projects must consider coupling, simplicity and scalability at higher priority but not other project category needs to consider these attributes at the same priority level as illustrated in Appendix 6.1.

Devise Ideology (D)

It pertains to the key design principles using some specific criteria such as costs minimization and maintenance requirements. There may exist various kinds of devise ideologies. Let D_j, $j=1..., m$; represent m kinds of devise ideologies.

For example, a project can have variety of devise ideologies, each with its own specific set of constraints such as physical separation of components across different servers, a limitation of compatibility, composition no context specifications etc. as depicted in Appendix 6.2.

Crosscutting Concerns (C)

Crosscutting concerns are the features of a project that may apply across all layers, components, and tiers. These are also the areas in which high-impact design mistakes are most often made. Therefore, it represents key areas of design that are not related to a specific application. Let C_k, $k=1..., n$; represents n kinds of crosscutting concerns.

Examples of crosscutting concerns include authentication, authorization, caching, communication, configuration and exception management etc. These crosscutting concerns vary across project categories as shown in Appendix 6.3

Weights (*w*)

Weight is a value assigned to each project parameter depending upon its priority in specific type of project category. All the project parameters such as quality attributes, devise ideologies and crosscutting concerns may have different priorities in quantifying reliability of a project.

For example, while considering quality attributes, higher priority may be assigned to flexibility as compared to scalability in a communication oriented projects. On the other hand, scalability is assigned higher priority than flexibility in deployment oriented projects. Similarly, weights may assign to devise ideologies and crosscutting concerns.

Effective Mean (*EM*)

Average of weights assigned to quality attributes (*Q*), devise ideologies (*D*) and crosscutting concerns (*C*) of the project is said to be an effective mean.

Deviation Factor (*DF*)

It is defined as the variability for every class of parameter under consideration and is denoted by *DF*. For example, *DF*(*i*) for quality attributes, *DF*(*j*) for devise ideology and *DF*(*k*) for crosscutting concerns.

Total Deviation Factor *(TDF)*

It is the sum of deviation factors (DFs') corresponding to every class of parameter.

Project Reliability (*R*)

It is ratio of total observed variability (*TDF*) captured across class of parameters to the total ideal variability of equivalent class of parameters (TDF_{ideal}).

6.5.2 Ontological Reliability Quantification Method (*ORQM*)

We propose Ontological Reliability Quantification Method (*ORQM*) that includes project parameters on the basis of project category for reliability quantification of project. We incorporate three classes of project parameters such as quality attributes, devise ideology and crosscutting concerns which are differ in numbers and weights as per the project category. The stepwise description of *ORQM* is as follows:

Step I: *Identification of project category.*

User must identify the PC of current project first.

Step II: *Identification of project parameters and allocation of corresponding weights.*

Identify project parameters depending on project category and assign weights.

Step III: *Computation of Effective Means (EMs) of various parameters.*

EMs corresponding to each class of project parameter related to each project i=1,2,3,..., N are as shown in equations (6.1), (6.2) and (6.3) respectively.

$$EMQ(i) = \frac{1}{l}\sum_{\alpha=1}^{l} Q(i,\alpha) \qquad \text{... (6.1)}$$

$$EMD(i) = \frac{1}{m}\sum_{\beta=1}^{m} D(i,\beta) \qquad \text{... (6.2)}$$

$$EMC(i) = \frac{1}{n}\sum_{\gamma=1}^{n} C(i,\gamma) \qquad \text{... (6.3)}$$

Step IV: *Computation of Deviation Factors (DFs) and Total Deviation Factor (TDF) of various parameters.*

DFs for each class of project parameters as well as *TDF* related to each project i=1,2,3,..., N are as shown in equations (6.4), (6.5), (6.6) and (6.7) respectively.

$$DFQ(i) = \frac{1}{l}\sum_{\alpha}^{l}(Q(i,\alpha) - EMQ(i))^2 \qquad \text{... (6.4)}$$

$$DFD(i) = \frac{1}{m}\sum_{\beta}^{m}(D(i,\beta) - EMD(i))^2 \qquad \text{... (6.5)}$$

$$DFC(i) = \frac{1}{n}\sum_{\gamma}^{n}(C(i,\gamma) - EMC(i))^2 \qquad \text{... (6.6)}$$

$$TDF(i) = \sum_{i=1}^{N}\big(DFQ(i) + DFD(i) + DFC(i)\big) \qquad \text{... (6.7)}$$

where N stands for total number of projects.

Step V: *Calculation of Ideal Total Deviation Factor (TDF_{ideal}) of various project categories.*

TDF_{ideal} refers to total deviation factor of an ideal project (i.e. the project possessing all the project parameters) and calculated using equations (6.1) to (6.7). TDF_{ideal} varies with the type of project category.

Step VI: *Calculation of Project Reliability*

$$R(i) = TDF(i) / TDF_{ideal}. \qquad \text{... (6.8)}$$

We have developed program in C for computation of reliability of various projects whose execution results are shown in Appendix 6.4.

6.5.3 Case Studies

We consider different applications to analyze results obtained from *ORQM*. Our study included petite projects of four *PC* namely; communication, deployment, domain and structured oriented projects. We use three classes of project parameters namely; quality attributes, devise ideologies and crosscutting concerns corresponding to each project category.

Case I- Communication Oriented Projects

It is assumed that communication oriented PCs can accommodate many quality attributes Q_i ranging from q_1 to q_{10}, *devise ideologies* D_j ranging from d_1 to d_5 and crosscutting concerns C_k ranging from c_1 to c_6 as shown in Table 6.10. Each project parameter is assigned some weight depending upon the frequency of its occurrence in maximum number of projects of that category. For example, domain alignment quality attribute is present in very few projects and therefore assigned weight as 1. Whereas coupling quality attribute is present in every project under study and hence assigned the weight as 10. Now, we consider five devise ideologies such as autonomous, distributable, loosely coupled, share

Table 6.10 Communication Oriented Projects

			P1	*P2*	*P3*	***P4***	*P5*	*P6*	*P7*
Q	q_1	Domain Alignment	1	1	0	**1**	0	1	0
	q_2	Abstraction	0	2	0	**2**	2	0	2
	q_3	Discoverability	0	3	3	**3**	0	3	3
	q_4	Interoperability	0	0	4	**0**	0	4	4
	q_5	Rationalization	5	5	5	**5**	5	5	0
	q_6	Extensibility	6	6	6	**6**	6	6	6
	q_7	Flexibility	0	0	7	**7**	0	7	7
	q_8	Scalability	8	8	0	**0**	0	8	0
	q_9	Simplicity	0	0	0	**0**	9	0	0
	q_0	Coupling	10	10	0	**10**	0	0	0
D	d_1	Autonomous	0	1	0	**1**	1	1	0
	d_2	Distributable	2	0	2	**2**	0	2	2
	d_3	Loosely coupled	3	3	3	**3**	3	3	3
	d_4	Share schema and contract	4	4	4	**4**	4	0	4
	d_5	Compatibility	5	0	5	**0**	0	0	0
C	c_1	Instrumentation and logging	0	1	0	**1**	1	1	0
	c_2	Authentication.	2	2	2	**0**	2	2	0
	c_3	Authorization	3	3	3	**0**	3	3	0
	c_4	Exceptn mgmt	4	0	4	**4**	4	0	4
	c_5	Communication	5	0	5	**5**	0	0	5
	c_6	Caching	0	0	6	**6**	0	0	0

schema and contract, compatibility and hence allocate the weights ranging from 1 to 5 respectively. Then, for crosscutting concerns such as instrumentation and logging, authentication, authorization, exception management, communication and caching, we are assign weights ranging from 1 to 6 respectively.

We attempt to quantify reliability for communication-oriented projects P_1 to P_7 using *ORQM* as shown in Table 6.11. Let us consider project P_4 for reliability calculation. P_4 considers the quality attributes such as domain alignment, abstraction, discoverability, rationalization, extensibility, flexibility and coupling. Next, it contains devise ideologies namely; autonomous, distributable, loosely coupled, share schema and contract and computability. Lastly, instrumentation and logging, exception management, communication and cashing are the crosscutting concerns present in project P_4. The assignment of weights is highlighted in Table 6.10. While executing *ORQM* for reliability quantification, the values of *EMQ*(*4*), *EMD*(*4*), *EMC*(*4*) are calculated to be 4.86, 2.5 and 4 respectively with the equations (6.1) to (6.3). Next, *DFQ(4),DFD(4)* and *DFC*(*4*) are calculated with equations (6.4) to (6.6) and the values are 8.41, 1.25 and 3.50 respectively. Subsequently, *TDF*(*4*) is calculated to be 13.16 with the help of equation (6.7). Then, TDF_{ideal} for communication oriented projects has been calculated using the same procedure and its value is 13.17 as highlighted in Table 6.11. Thus, the Reliability *R*(*4*)for project P_4 comes to be 0.99.

Case II- Deployment Oriented Projects

Let us consider now deployment oriented projects having illustrious combinations of Q_i from ranging from q_1 to q_9, D_j ranging from d_1 to d_3 and C_k ranging from c_1 to c_6 as shown in Table 6.12. Quality attributes such as maintainability, scalability, flexibility, availability, security, central access, supportability, usability and integrity as quality attributes allocated weights ranging from 1 to 9 respectively; devise ideologies having separation of concerns, event based notification and delegated event handling with corresponding weights ranging from 1 to 3; and crosscutting concerns projects possessing authentication,

Table 6.11 Reliability Computation of Communication Oriented Projects

Project	*Quality Attributes*										*Devise Ideologies*					*Crosscutting Concerns*						*TDF*	*R*
	q1	*q2*	*q3*	*q4*	*q5*	*q6*	*q7*	*q8*	*q9*	*q10*	*d1*	*d2*	*d3*	*d4*	*d5*	*c1*	*c2*	*c3*	*c4*	*c5*	*c6*		
P1	1	0	0	0	5	6	0	8	0	10	0	2	3	4	5	0	2	3	4	5	0	11.70	0.888383
P2	1	2	3	0	5	6	0	8	0	10	1	0	3	4	0	1	2	3	0	0	0	11.37	0.86329
P3	0	0	3	4	5	6	7	0	0	0	0	2	3	4	5	0	2	3	4	5	6	10.15	0.770691
P4	**1**	**2**	**3**	**0**	**5**	**6**	**7**	**0**	**0**	**10**	**1**	**2**	**3**	**4**	**0**	**1**	**0**	**0**	**4**	**5**	**6**	**13.16**	**0.999349**
P5	0	2	0	0	5	6	0	0	9	0	1	0	3	4	0	1	2	3	4	0	0	9.06	0.687927
P6	1	0	3	4	5	6	7	8	0	0	1	2	3	0	0	1	2	3	0	0	0	6.32	0.479589
P7	0	2	3	4	0	6	7	0	0	0	0	2	3	4	0	0	0	0	4	5	0	4.36	0.330802
P_{ideal}	**1**	**2**	**3**	**4**	**5**	**6**	7	**8**	**9**	**10**	**1**	**2**	**3**	**4**	**5**	**1**	**2**	**3**	**4**	**5**	**6**	**13.17**	**1**

Table 6.12 Deployment Oriented Projects

			P8	*P9*	*P10*	*P11*	*P12*	*P13*	*P14*
Q	q_1	Maintainability	1	1	1	1	1	1	1
	q_2	Scalability	2	0	2	0	2	0	0
	q_3	Flexibility	0	3	0	3	0	3	3
	q_4	Availability	4	4	0	4	0	4	4
	q_5	Security	5	5	5	5	5	0	5
	q_6	Central Access	6	0	6	0	6	6	6
	q_7	Supportability	7	7	0	7	7	0	0
	q_8	Usability	8	8	8	8	8	8	0
	q_9	Integrity	0	9	9	0	0	9	9
D	d_1	Separation of concerns	1	1	1	1	1	1	1
	d_2	Event based notification	2	2	2	2	2	0	2
	d_3	Delegated event handling	3	3	3	0	3	0	0
C	c_1	Authentication.	1	1	1	0	1	1	1
	c_2	Authorization	2	2	2	2	2	2	2
	c_3	Exceptn mgmt	0	3	3	0	3	3	0
	c_4	Communication	4	4	4	4	0	0	4
	c_5	Cryptography	5	0	0	5	0	0	0
	c_6	Sensitive data	6	0	0	0	0	0	6

authorization, exception management, communication, cryptography and sensitive data acquired with weights ranging from 1 to 6 respectively.

ORQM is executed for deployment-oriented projects P_8 to P_{14} for reliability quantification and depicted in Table 6.13. It is found that minimum value of reliability of project is 0.59 and maximum is 0.95.

Case III- Domain Oriented Projects

While domain oriented projects are studied, it is observed that the quality attributes such as Q_i ranging from q_1 to q_9, devise ideologies D_j ranging from d_1 to d_5 and crosscutting concerns C_k ranging from c_1 to c_5 as shown in Table 6.14 helps to quantify project reliability. Therefore, quality attributes such as communication, extensibility and testability etc. are weighted ranging from 1 to 9, devise ideologies such as pensiveness, composition and legacy etc. are having weights ranging from 1 to 5. On the other hand, crosscutting concerns such as cashing, data validation and configuration management etc. hold weights ranging from 1 to 5 correspondingly. Next, P_{15} to P_{21} domain-oriented projects are considered for reliability quantification using *ORQM* as shown in Table 6.15. It is established that domain oriented projects having reliability value as 0.32 for project P_{18} which is minimum and 0.73 for project P_{20} as highlighted in Table 6.15. These values imply that the project P_{18} is 32% reliable whereas project P_{20} is 73% reliable.

Case IV- Structured Oriented Projects

Lastly, we quantify reliability for structured oriented projects. The required quality attributes, devise ideologies and crosscutting concerns are observed to allocate weights. Table 6.16 illustrates Abstraction, isolation, manageability, performance, reusability and testability etc. and possesses weight in a order of 1 to 10. While, devise ideologies consist of reusable, replaceable, not context specified etc. are having 1 to 5 weights correspondingly. In addition, authentication, audit and logging and communication hold 1 to 5 weights in that order and constitute crosscutting concerns. Finally, Table 6.17 shows *ORQM* execution for P_{22} to P_{28} structured oriented projects. It is ascertained that

Table 6.13 Reliability Computation of Deployment Oriented Projects

Project	***Quality Attributes***									***Devise Ideologies***			***Crosscutting Concerns***						***TDF***	***R***
	q_1	q_2	q_3	q_4	q_5	q_6	q_7	q_8	q_9	d_1	d_2	d_3	c_1	c_2	c_3	c_4	c_5	c_6		
P8	1	2	0	4	5	6	7	8	0	1	2	3	1	2	0	4	5	6	7.78	0.821944
P9	1	0	3	4	5	0	7	8	9	1	2	3	1	2	3	4	0	0	7.10	0.749919
P10	1	2	0	0	5	6	0	8	9	1	2	3	1	2	3	4	0	0	7.50	0.792467
P11	1	0	3	4	5	0	7	8	0	1	2	0	0	2	0	4	5	0	6.77	0.714678
P12	1	2	0	0	5	6	7	8	0	1	2	3	1	2	3	0	0	0	7.37	0.778718
P13	**1**	**0**	**3**	**4**	**0**	**6**	**0**	**8**	**9**	**1**	**0**	**0**	**1**	**2**	**3**	**0**	**0**	**0**	**5.59**	**0.590074**
P14	**1**	**0**	**3**	**4**	**5**	**6**	**0**	**0**	**9**	**1**	**2**	**0**	**1**	**2**	**0**	**4**	**0**	**6**	**9.03**	**0.953502**
P_{ideal}	1	2	3	4	5	6	7	8	9	1	2	3	1	2	3	1	5	6	9.47	1.000235

Table 6.14 Domain Oriented Projects

			P15	*P16*	*P17*	*P18*	*P19*	*P20*	*P21*
Q	q_1	Communication	1	0	1	1	1	1	1
	q_2	Extensibility	2	0	2	2	2	2	0
	q_3	Testability	3	3	3	3	3	0	3
	q_4	Simplicity	4	4	0	4	0	4	0
	q_5	Highly cohesive	5	5	5	0	0	5	0
	q_6	Understanding	6	0	6	0	0	0	0
	q_7	Manageability	7	0	7	0	0	0	0
	q_8	Integrity	0	8	0	0	8	8	8
	q_9	Decoupling	0	9	0	0	9	0	0
D	d_1	Pensiveness	1	0	1	1	1	1	0
	d_2	Composition	2	2	2	2	2	2	2
	d_3	Legacy	3	0	0	0	0	0	3
	d_4	Encapsulation	4	4	0	4	0	0	4
	d_5	Binding	0	5	0	0	0	5	5
C	c_1	Cashing	1	1	1	1	0	1	0
	c_2	data validation	2	2	2	2	2	2	0
	c_3	Config. Mgmt	3	0	0	0	3	3	0
	c_4	Authorization	0	4	0	4	0	4	4
	c_5	Exceptn Mgmt	0	5	0	0	5	0	5

Table 6.15 Reliability Computation of Domain Oriented Projects

Project	*Quality Attributes*									*Devise Ideologies*					*Crosscutting Concerns*					*TDF*	*R*
	q_1	q_2	q_3	q_4	q_5	q_6	q_7	q_8	q_9	d_1	d_2	d_3	d_4	d_5	c_1	c_2	c_3	c_4	c_5		
P15	1	2	3	4	5	6	7	0	0	1	2	3	4	0	1	2	3	0	0	6.69	0.631492
P16	0	0	3	4	5	0	0	8	9	0	2	0	4	5	1	2	0	4	5	6.76	0.637933
P17	1	2	3	0	5	6	7	0	0	1	2	0	0	0	1	2	0	0	0	4.38	0.413126
P18	**1**	**2**	**3**	**4**	**0**	**0**	**0**	**0**	**0**	**1**	**2**	**0**	**4**	**0**	**1**	**2**	**0**	**4**	**0**	**3.44**	**0.324689**
P19	1	2	3	0	0	0	0	8	9	1	2	0	0	0	0	2	3	0	5	6.52	0.615293
P20	**1**	**2**	**0**	**4**	**5**	**0**	**0**	**8**	**0**	**1**	**2**	**0**	**0**	**5**	**1**	**2**	**3**	**4**	**0**	**7.83**	**0.739208**
P21	1	0	3	0	0	0	0	8	0	0	2	3	4	5	0	0	0	4	5	4.78	0.451066
P_{ideal}	1	2	3	4	5	6	7	8	9	1	2	3	4	5	1	2	3	4	5	10.59	1

Table 6.16 Structured Oriented Projects

			P22	*P23*	*P24*	*P25*	*P26*	*P27*	*P28*
Q	q_1	Abstraction	1	1	1	1	0	0	1
	q_2	Isolation	2	2	0	0	0	0	2
	q_3	Manageability	3	3	3	3	3	3	3
	q_4	Performance	4	4	4	0	4	4	4
	q_5	Reusability	5	0	0	5	5	5	0
	q_6	Testability	6	0	0	6	0	0	6
	q_7	Ease of Deployment	7	7	0	0	0	0	7
	q_8	Reduced outlay	0	8	8	0	8	8	8
	q_9	Ease of development	0	0	0	0	0	9	0
	q_0	Techcomplexity	0	0	10	10	10	0	0
D	d_1	Reusable	1	0	1	1	1	0	1
	d_2	Replaceable	2	2	2	0	2	0	2
	d_3	No context spec	0	3	3	3	3	3	3
	d_4	Independent	4	4	0	0	0	4	0
	d_5	High Cohesion	5	0	0	0	0	0	0
C	c_1	Authentication	1	1	1	0	1	0	0
	c_2	Audit &logging	2	2	2	2	2	0	2
	c_3	Communication	0	0	0	3	3	3	0
	c_4	Exceptn Mgmt	4	0	0	0	0	0	0
	c_5	Validation	5	5	0	0	0	5	0

Table 6.17 Reliability Computation of Structured Oriented Projects

Project	***Quality Attributes***										***Devise Ideologies***					***Crosscutting Concerns***					***TDF***	***R***
	q_1	q_2	q_3	q_4	q_5	q_6	q_7	q_8	q_9	q_0	d_1	d_2	d_3	d_4	d_5	c_1	c_2	c_3	c_4	c_5		
P22	1	2	3	4	5	6	7	0	0	0	1	2	0	4	5	1	2	0	4	5	9.00	0.734694
P23	**1**	**2**	**3**	**4**	**0**	**0**	**7**	**8**	**0**	**0**	**0**	**2**	**3**	**4**	**0**	**1**	**2**	**0**	**0**	**5**	**10.03**	**0.818601**
P24	**1**	**0**	**3**	**4**	**0**	**0**	**0**	**8**	**0**	**10**	**1**	**2**	**3**	**0**	**0**	**1**	**2**	**0**	**0**	**0**	**5.92**	**0.482993**
P25	1	0	3	0	5	6	0	0	0	10	1	0	3	0	0	0	2	3	0	0	6.25	0.510204
P26	0	0	3	4	5	0	0	8	0	10	1	2	3	0	0	1	2	3	0	0	8.13	0.663946
P27	0	0	3	4	5	0	0	8	9	0	0	0	3	4	0	0	0	3	0	5	6.61	0.539592
P28	1	2	3	4	0	6	7	8	0	0	1	2	3	0	0	0	2	0	0	0	6.63	0.540892
P_{ideal}	1	2	3	4	5	6	7	8	9	10	1	2	3	4	5	1	2	3	4	5	12.25	1

reliability of structured oriented projects under study varies from 0.48 to 0.81. Project P_{24} has minimum reliability and 0.97 and project P_{23} is having maximum reliability as highlighted in Table 6.17. These values conclude that the project P_{24} is 48% reliable whereas project P_{23} is 81% reliable.

6.5.4 Observations

Reliability quantification of software projects is an exigent job due to its varying setting. Major challenge that lies in quantification is kind of project categories and affecting project parameters. With the help of *ORQM*, quantification may be accomplished on the basis of weight allocation to project parameters with corresponding project categories. These are useful in identifying the early scale of project reliability and establishing the software excellence. We also presented four cases concerned with project categories such as communication, deployment, domain and structure. Further, we computed certain statistics such as effective mean *EM*, deviation factor *DF* and total deviation factor *TDF* of these project cases. Further, we have computed software reliability *R* of every project of various project categories. And, we have observed some facts as follows:

- *ORQM* provides the facility to improve the traditional reliability checking mechanism by considering the architecture style thereby providing the scope of improvement in reliability estimation and the actual facts to user and developer.
- *ORQM* computes minimum reliability for communication oriented projects as 47% and maximum reliability as 99%. Thus, a direct measure may be provided for reliability quantification for any project.
- It is observed that, reliability of most of deployment oriented projects under study ranges from 71.4% to 79.2%. For these projects, quality attributes such as maintainability, security and usability as well as devise ideologies such as separation of concerns and event based notification and crosscutting concern mainly authorization plays vital role.

- *ORQM* quantifies maximum reliability of domain oriented project under study as 73% and minimum as 32%. It is found that, these values are typically less than corresponding maximum and minimum values of reliability for other projects of different categories.
- *ORQM* eliminates the need of failure data and experts. Therefore, an average project developer can quantify reliability more precisely.
- *ORQM* provides flexibility on number and type of vital project parameters and project category depending on the project behavior and team makeup.
- *ORQM* also resolves the limitations of reliability engineering by associating weights to each of the project parameter according to project category.
- Ontological approach for reliability quantification of software projects leads to a step towards the engineering practices thereby establishing the fact that these methods are not informal methods.

6.6 Summary

Software reliability achievement is a challenging task due to its dependency on users' perspective. We have introduced ontological approach for reliability achievement over object-oriented approach. Then, a comparative analysis has been presented and scope of Ontology Oriented Reliability (*OnO-Reliability*) has been outlined. In addition, ontological specifications have been developed using *OntoReliability* protocol. We have presented some case studies to understand the application of *OntoReliability* protocol for software specification development. Subsequently, the benefits have been discussed. Lastly, we have attempted to quantify the reliability using various project parameters. For the same reason, we have introduced Ontological Reliability Quantification Method (*ORQM*). These project parameters vary in their number and type as per the category of project. Therefore, we have considered the project category as a prerequisite for computing the reliability. We conducted a study of different project case as per the category with varying number and type of parameters and establish the fact that *ORQM* generates direct empirical value for software reliability. Finally, we conclude that *ORQM* is not a informal method but found to be a highly useful in absence of reliability experts and historical failure data.

Project Parameters

1. Quality Attributes

S. No.	Quality Attributes	Description
1	Domain Alignment	Reuse of common services with standard interfaces. It increases business and technology opportunities and reduces cost.
2	Abstraction	Related to autonomous services and accessed through a formal contract.
3	Discoverability	Expose descriptions that allow other applications and services to locate them and automatically determine the interface.
4	Interoperability	Ability of a system or different systems to operate successfully by communicating and exchanging information with other external systems written and run by external parties. An interoperable system makes it easier to exchange and reuse information internally as well as externally.
5	Rationalization	Provide specific functionality, rather than duplicating the functionality in number of applications.
6	Extensibility	Applications can be added to or removed from the bus without having an impact on the existing applications.
7	Flexibility	Ability to change easily to match changes in business or user requirements, simply through changes to the configuration or parameters that control routing.
8	Scalability	Multiple instances of the same application can be attached to the bus in order to handle multiple requests at the same time.
9	Simplicity	To support a single connection to the message bus instead of multiple connections to other applications.
10	Maintainability	Ability of the system to undergo changes with a degree of ease. These changes could impact components, services, features, and interfaces when adding or changing the functionality, fixing errors, and meeting new business requirements.
11	Availability	Enabling systems using easily scalable components.
12	Security	Capability of a system to prevent malicious or accidental actions outside of the designed usage, and to prevent disclosure or loss of information. A secure system aims to protect assets and prevent unauthorized modification of information.
13	Central Access	Ease to administer to access and updates the data.
14	Supportability	Ability of the system to provide information helpful for identifying and resolving issues when it fails to work correctly.

15	Usability	Defines how well the application meets the requirements of the user and consumer by being intuitive, easy to localize and globalize, providing good access for disabled users, and resulting in a good overall user experience.
16	Integrity	Defines the consistency and coherence of the overall design. This includes the way that components or modules are designed, as well as factors such as coding style and variable naming.
17	Communication	Defines to communicate business knowledge and requirements using a common business domain language.
18	Testability	Measure of how easy it is to create test criteria for the system and its components, and to execute these tests in order to determine if the criteria are met.
19	Highly cohesive	Locating related methods and features in an object, and using different objects for different sets of features
20	Understanding	Maps the application more closely to the real world
21	Decoupling	Provide alternative implementations without affecting consumers of the interface
22	Isolation	Upgrades to individual layers in order to reduce risk and minimize impact on the overall system.
23	Performance	Indication of the responsiveness of a system to execute any action within a given time interval.
24	Reusability	Defines the capability for components and subsystems to be suitable for use in other applications and in other scenarios. Reusability minimizes the duplication of components and also the implementation time.
25	Ease of Deployment	Replace existing versions with no impact on the other components or the system as a whole
26	Reduced outlay	Allows to spread the cost of development and maintenance
27	Ease of development	Provide defined functionality, allowing development without impacting other parts of the system
28	Techcomplexity	Mitigate complexity through the use of a component container and its services
29	Manageability	Separation of core concerns helps to identify dependencies, and organizes the code into more manageable sections.

Project Parameters

2. Devise Ideologies

S. No.	Devise Ideologies	Description
1	Autonomous	Each service is maintained, developed, deployed, and versioned independently.
2	Distributable	Services can be located anywhere on a network, locally or remotely, as long as the network supports the required communication.
3	Share schema and contract	Services share contracts and schemas when they communicate, not internal classes.
4	Compatibility policy	Defines features such as transport, protocol, and security.
5	Separation of concerns	Divide UI processing concerns into distinct roles; for example MVC has three roles: Model, View, and Controller. Model represents data, View represents UI; and Controller handles requests, manipulates the model, and performs other operations.
	Event based notification	Used to provide notifications to the View when data managed by the Model changes.
7	Replaceable	Components may be readily substituted with other similar components.
8	Delegated event handling	Handles events triggered from the UI controls in the View.
9	No context specification	Components are designed to operate in different environments and contexts
10	Independent	Provides minimal dependencies on other components. Therefore, components can be deployed into any appropriate environment without affecting other components or systems
11	Pensiveness	reduce a complex operation into a generalization that retains the base characteristics of the operation.
12	Composition	Objects can be assembled from other objects, and can choose to hide these internal objects from other classes or expose them as simple interfaces.
13	Legacy	Objects can inherit from other objects, and use functionality in the base object or override it to implement new behavior
14	Encapsulation	Expose interfaces that allow the caller to use its functionality, and do not reveal details of the internal processes or any internal variables or state.
15	Binding	Allows to override the behavior of a base type that supports operations in your application by implementing new types that are interchangeable with the existing object.

Project Parameters

3. Crosscutting Concerns

S. No.	Crosscutting Concerns	Description
1	Instrumentation and logging	Instrument all of the business-critical and system-critical events, and log sufficient details to recreate events in your system without including sensitive information.
2	Authentication.	Determine to authenticate users and pass authenticated identities across the layers.
3	Authorization	Ensure proper authorization with appropriate granularity within each layer, and across trust boundaries.
4	Exception management	Catch exceptions at functional, logical, and physical boundaries; and avoid revealing sensitive information to end users.
5	Communication	Choose appropriate protocols, minimize calls across the network, and protect sensitive data passing over the network.
6	Caching	Determine what data to cache, where to cache the data, and a suitable expiration policy
7	Cryptography	Refers to application enforces confidentiality and integrity
8	da:a validation	Technique choose for validating on length, range, format, and type; constrain and reject input invalid values; sanitize potentially malicious or dangerous input
9	Configuration Management	Determine the information configurable, to store configuration information and to protect sensitive configuration information,

ORQM Execution Results for Communication Oriented Projects

Project	*Quality Attributes*										*Devise Ideologies*					*Crosscutting Concerns*						*EMQ*	*EMD*	*EMC*	*DFQ*	*DFD*	*DFC*	*TDF*	*R*
	q1	*q2*	*q3*	*q4*	*q5*	*q6*	*q7*	*q8*	*q9*	*q10*	*d1*	*d2*	*d3*	*d4*	*d5*	*c1*	*c2*	*c3*	*c4*	*c5*	*c6*								
P1	1	0	0	0	5	6	0	8	0	10	0	2	3	4	5	0	2	3	4	5	0	6	3.5	3.5	9.20	1.25	1.25	11.7	0.88
P2	1	2	3	0	5	6	0	8	0	10	1	0	3	4	0	1	2	3	0	0	0	5	2.67	2	9.14	1.56	0.67	11.3	0.86
P3	0	0	3	4	5	6	7	0	0	0	0	2	3	4	5	0	2	3	4	5	6	5	3.5	4	1.20	3.75	5.20	10.1	0.77
P4	**1**	**2**	**3**	**0**	**5**	**6**	**7**	**0**	**0**	**10**	**1**	**2**	**3**	**4**	**0**	**1**	**0**	**0**	**4**	**5**	**6**	4.86	2.5	4	8.41	1.25	3.50	13.1	0.99
P5	0	2	0	0	5	6	0	0	9	0	1	0	3	4	0	1	2	3	4	0	0	5.5	2.67	2.5	6.25	1.56	1.25	9.06	0.68
P6	1	0	3	4	5	6	7	8	0	0	1	2	3	0	0	1	2	3	0	0	0	4.86	2	2	4.98	0.67	0.67	6.32	0.47
P7	0	2	3	4	0	6	7	0	0	0	0	2	3	4	0	0	0	0	4	5	0	4.4	3	4.5	3.44	0.67	0.25	4.36	0.33
*P*ideal	1	2	3	4	5	6	7	8	9	10	1	2	3	4	5	1	2	3	4	5	6	5.5	3	3.5	8.25	2.00	2.92	13.1	1.00

ORQM Execution Results for Deployment Oriented Projects

Project	*Quality Attributes*									*Devise Ideologies*			*Crosscutting Concerns*						*EMQ*	*EMD*	*EMC*	*DFQ*	*DFD*	*DFC*	*TDF*	*R*
	q1	*q2*	*q3*	*q4*	*q5*	*q6*	*q7*	*q8*	*q9*	*d1*	*d2*	*d3*	*c1*	*c2*	*c3*	*c4*	*c5*	*c6*								
P8	1	2	0	4	5	6	7	8	0	1	2	3	1	2	0	4	5	6	4.71429	2	3.6	3.68	0.67	3.44	7.78	0.821944
P9	1	0	3	4	5	0	7	8	9	1	2	3	1	2	3	4	0	0	5.28571	2	2.5	5.19	0.67	1.25	7.10	0.749919
P10	1	2	0	0	5	6	0	8	9	1	2	3	1	2	3	4	0	0	5.16667	2	2.5	5.59	0.67	1.25	7.50	0.792467
P11	1	0	3	4	5	0	7	8	0	1	2	0	0	2	0	4	5	0	4.66667	1.5	3.666667	5.08	0.13	1.56	6.77	0.714678
P12	1	2	0	0	5	6	7	8	0	1	2	3	1	2	3	0	0	0	4.83333	2	2	6.04	0.67	0.67	7.37	0.778718
P13	1	0	3	4	0	6	0	8	9	1	0	0	1	2	3	0	0	0	5.16667	1	2	4.92	0.00	0.67	5.59	0.590074
P14	1	0	3	4	5	6	0	0	9	1	2	0	1	2	0	4	0	6	4.66667	1.5	4.333333	3.99	0.25	4.79	9.03	0.953502
Pideal	1	2	3	4	5	6	7	8	9	1	2	3	1	2	3	4	5	6	5	2	3.5	4.89	1.67	2.92	9.47	1.000235

ORQM Execution Results for Domain Oriented Projects

Project	***Quality Attributes***									***Devise Ideologies***					***Crosscutting Concerns***					***EMQ***	***EMD***	***EMC***	***DFQ***	***DFD***	***DFC***	***TDF***	***R***
	q1	*q2*	*q3*	*q4*	*q5*	*q6*	*q7*	*q8*	*q9*	*d1*	*d2*	*d3*	*d4*	*d5*	*c1*	*c2*	*c3*	*c4*	*c5*								
P15	1	2	3	4	5	6	7	0	0	1	2	3	4	0	1	2	3	0	0	4	2.5	2	4.00	1.69	1.00	6.69	0.631492
P16	0	0	3	4	5	0	0	8	9	0	2	0	4	5	1	2	0	4	5	5.8	3.66	3	2.37	0.89	3.50	6.76	0.637933
P17	1	2	3	0	5	6	7	0	0	1	2	0	0	0	1	2	0	0	0	4	1.5	1.5	4.00	0.13	0.25	4.38	0.413126
P18	1	2	3	4	0	0	0	0	0	1	2	0	4	0	1	2	0	4	0	2.5	2.3	2.33	0.71	1.17	1.56	3.44	0.324689
P19	1	2	3	0	0	0	0	8	9	1	2	0	0	0	0	2	3	0	5	4.6	1.5	3.33	4.83	0.13	1.56	6.52	0.615293
P20	1	2	0	4	5	0	0	8	0	1	2	0	0	5	1	2	3	4	0	4	2.66	2.5	4.29	0.73	2.81	7.83	0.739208
P21	1	0	3	0	0	0	0	8	0	0	2	3	4	5	0	0	0	4	5	4	3.5	4.5	3.71	0.69	0.38	4.78	0.451066
Pideal	1	2	3	4	5	6	7	8	9	1	2	3	4	5	1	2	3	4	5	5	3	3	6.29	1.50	2.80	10.59	1.0000

ORQM Execution Results for Structured Oriented Projects

Project	*Quality Attributes*										*Devise Ideologies*					*Crosscutting Concerns*					*EMQ*	*EMD*	*EMC*	*DFQ*	*DFD*	*DFC*	*TDF*	*R*
	q1	*q2*	*q3*	*q4*	*q5*	*q6*	*q7*	*q8*	*q9*	*q10*	*d1*	*d2*	*d3*	*d4*	*d5*	*c1*	*c2*	*c3*	*c4*	*c5*								
P22	1	2	3	4	5	6	7	0	0	0	1	2	0	4	5	1	2	0	4	1	4	3	3.00	4.00	2.5	2.50	9.00	0.734694
P23	1	2	3	4	0	0	7	8	0	0	0	2	3	4	0	1	2	0	0	1	4.166667	3	2.67	6.47	0.666667	2.89	10.03	0.818601
P24	1	0	3	4	0	0	0	8	0	10	1	2	3	0	0	1	2	0	0	1	5.2	2	1.50	5.00	0.666667	0.25	5.92	0.482993
P25	1	0	3	0	5	6	0	0	0	10	1	0	3	0	0	0	2	3	0	1	5	2	2.50	5.00	1	0.25	6.25	0.510204
P26	0	0	3	4	5	0	0	8	0	10	1	2	3	0	0	1	2	3	0	0	6	2	2.00	6.80	0.666667	0.67	8.13	0.663946
P27	0	0	3	4	5	0	0	8	9	0	0	0	3	4	0	0	0	3	0	0	5.8	3.5	4.00	5.36	0.25	1.00	6.61	0.539592
P28	1	2	3	4	0	6	7	8	0	0	1	2	3	0	0	0	2	0	*0*	1	4.428571	2	2.00	5.96	0.666667	0.00	6.63	0.540892
Pideal	**1**	**2**	**3**	**4**	**5**	**6**	**7**	**8**	**9**	**10**	**1**	**2**	**3**	**4**	**5**	**1**	**2**	**3**	**4**	**1**	**5.5**	**3**	**3**	8.25	2.00	2.00	12.25	1

CHAPTER 7

Ontology Based Secured Project Development

7.1 Introduction

The key benefits of Ontology Based Projects (*OBPs*) lie in increasing the abstraction and building secured software. *OBPs* are eulogized by software developers and practitioners due to the knowledge-intensive features. These projects ensure greatest ontology convention for being knowledge-intensive in nature. Ontology refers to the terms used to describe and represent an area of knowledge. It is observed that *OBPs* have achieved range of benefits by using different knowledge representation perspectives. At the same time, while developing *OBPs* from different perspectives, various side effects may occur due to unvisualized states. These states constitute mainly uncertainty, variability, ambiguity and complexity. At the same time, these states possess very important functional characteristics and thereby may disfigure the project security. However, security has been considered as an innate property of software project and expected to be en suite [EMB04, MC06]. Consequently, effective software security control becomes significant in *OBPs* due to the expediency, flexibility and comprehensibility.

Consequently, many formal methods have been used for resolving the security concerns in *OBPs*. These include Model Driven Architecture (MDA), Model Driven Development (MDD), Model Based Testing etc. [MHG+05]. These are disciplined methods, with the incorporation of mathematically based techniques for the specification, development, and verification of software. For the same reason, vulnerabilities can result from functionally incorrect implementations. Formal methods improve software security but all these formal methods have limitations of scale and applicability in principle [FE09, MF11].

Also, a formal method cannot prove that a formal specification that captures a user's intuitive understanding of a system and furthermore cannot prove that an implementation runs correctly [FEN11, RK11]. In addition, formal methods without modularization capabilities and scope-delimiting rules are difficult to use on large systems at any but the highest level of abstraction [CHK+07]. Hence, development of secured environment is required for *OBPs*. An Abstraction Method *(AM)* is suggested that caters *OBPs* developed from various perspectives such as generality, requirement, reuse and reliability engineering. Moreover, benefits associated with these perspectives induce the unvisualized states and impact security aspects of *OBPs*. Eventually, it leads to identify the security factors compensating the effect of unvisualized states.

7.2 Background

We have performed a systematic study to identify the benefits obtained in *OBPs* developed using different perspectives, discussed in previous chapters. Firstly, we have developed the projects termed as Ontology Driven Information Systems *(ODISs)* from the generality perspective. It has been carried out by mapping the Object-Oriented and Ontology-Oriented SDLC [SI11a]. Furthermore, projects with requirement engineering perspective have been developed using Ontology Aided Requirement Engineering *(OntoAidedRE)* Model. It is intended to enable knowledge driven requirement engineering by encapsulating the ontology [SI11b]. However, an *Ontop4ViewReuse* framework has been developed to endow with reuse perspective in software applications. This framework uses ontology oriented systematic *P4View* approach for reusing. *P4* stands for *Pretence-Persuade-Problem-Product views* and represent a meticulous attribute of the ontology [SI11d]. Lastly, to grow with reliability perspective *OntoReliability* Protocol has been proposed for developing Ontology Oriented Specifications *(OntoRelSpecifications)* [SI12a]. Now, we discuss *OBPs* along with the benefits and unvisualized states incurred due to these benefits in subsequent sub sections.

7.2.1 Ontology Based Projects (*OBPs*)

The projects developed with ontology based mechanism are known as Ontology Based Projects *(OBPs)*. We have developed Ontology Based Projects *(OBPs)* on the basis of aforesaid perspectives. Therefore, *OBPs* developed using generality perspective refers to Generality Oriented *OBPs*. Subsequently, Requirement engineering Oriented *OBPs* signifies *OBPs* developed using requirements engineering perspective. While, *OBPs* build up with reusability perspective denotes Reusability Oriented *OBPs*. Lastly, Reliability Oriented *OBPs* specify the *OBPs* extended via reliability perspective. We briefly describe the types of *OBPs* in this section.

7.2.1.1 Generality Oriented Ontology Based Projects

Generality Oriented Ontology Based Projects in form of *ODIS* describes specification of concepts and relations that exist in the domain, definitions, properties and constraints. Various projects from the generality perspective have been developed by mapping Object-Oriented and Ontology-Oriented software development life cycles as discussed in Chapter 3. Firstly, ontological approach involves ontology development mapped with domain analysis. Next, plotting of ontologies into object models mapped with designing infrastructure specification. Lastly, construction and implementation mapped with development of reusable component.

7.2.1.2 Requirement Oriented Ontology Based Projects

Requirement Oriented Ontology Based Projects aim to enable knowledge driven requirement engineering by encapsulating ontology. *OntoAidedRE*, provides the inter-relationships between different domains and multidisciplinary environment. In addition, it emphasized on and extracted from the endeavor depending on the requirement type. It is comprised of four layers namely; *OntoPre, OntoInput, OntoSystem and OntoOutput* requirements. *OntoPre Requirements* are dedicated for the use of system whereas *OntoInput Requirements* refer to initial system qualifying terms. Then, *OntoSystem*

Requirements have comprised of three components namely; *OntoSystem Operational, OntoSystem Control and OntoSystem Parameter Requirements. OntoSystem Operational Requirements* illustrate system access procedures and *OntoSystem Control Requirements* depicts system control procedures. Next, *OntoSystem Parameter Requirements* present system parameterized procedures. Finally, *OntoOutput Requirements* reflects the system eventual presentation as conferred in Chapter 4.

7.2.1.3 Reusability Oriented Ontology Based Projects

Ontop4ViewReuse framework, based on *P4View* approach for reusing is developed to obtain Reusability Oriented Ontology Based Projects as discussed in Chapter 5. It represents various types of ontology such as high level, domain, task and application ontologies that caters *Pretence, Problem, Persuade* and *Product views* respectively. It started with *Pretence view* by identification of knowledge sources useful for the application domains, which differ both in the represented content and in the formalization. Subsequently, an automatic translation of the source ontologies from a common format to the representation languages has carried out at *Problem view.* In addition, matching of the ensuing method is accepted at *Persuade view.* Finally, the application ontologies revealed the reuse source vocabularies to a large extent in *Product view.*

7.2.1.4 Reliability Oriented Ontology Based Projects

OntoReliability Protocol is suggested for developing software specifications *OntoRelSpecification1* to *OntoRelSpecification5* to attain Reliability Oriented Ontology Based Projects in Chapter 6. It improves external world representation in the users' intentions. Also, it controls the changes to the beliefs vigilantly with meta level structure specifically using ontology for contextual discrimination. *OntoRelSpecifications1* are composed of description, preconditions and post conditions and describes complete knowledge representation with respect to all desires that must be consistent across all domain users. Next, *OntoRelSpecifications2* signify the predicted process execution in advance along with the alternate completing approach. Then, *OntoRelSpecifications3* illustrate the occurrence of the exception assumed to be immediately captured and can be automatically stored in exception repository according to different classifications. These anticipated exceptions have included as service unavailability, deadline expiry, external trigger and rationality violation. *OntoRelspecification4* specifically focused on inclusions, priority, and frequency of uses and *OntoRelspecification5* comprised of special requirements and notes and issues.

7.2.2 Benefits of *OBPs*

Ontology has been used in software development for enriching the softwares with a range of perspectives. As discussed earlier, these perspectives include generality, knowledge intensive requirement engineering, reusability and reliability to build *OBP*s. These *OBP*s are acquired with various benefits and we discuss them as follows:

Conceptual Integrity

Conceptual integrity is of the essence in system development and cannot be easily perceived as a coherent system. The word conceptual is associated with the cognitive process of concept formation that involves the conscious recognition and identification of elements of our experience. However, the word integrity is associated with the idea of 'being one'. Hence, it defines the consistency and coherence of the overall design. This includes the way that components or modules are designed. In obtaining conceptual integrity, we concern with the emergence of concepts from experience [LINK7].

Maintainability

It is the ability of a system to undergo changes with a degree of ease. These changes could impact components, services, features, and interfaces when adding or changing the functionality, fixing errors, and meeting new business requirements. In addition, once a piece of system has failed it must be possible to get it back into an operating condition as soon as possible and termed as maintainability [BDB06].

Table 7.1: Benefits associated with *OBPs*

S. No.	Benefits	Description
B1	Conceptual Integrity	It defines the consistency and coherence of the overall design.
B2	Maintainability	It is the ability of the system to undergo changes with a degree of ease.
B3	Recoverability	It is the capability for components and subsystems to be suitable for use in other applications and in other scenarios.
B4	Accessibility	It is the proportion of time that the system is functional and working.
B5	Interoperability	It is the ability of a system or different systems to operate successfully by communicating and exchanging information with other external systems written and run by external parties.
B6	Manageability	It defines how easy it is for system administrators to manage the application, usually through sufficient and useful instrumentation exposed for use in monitoring systems and for debugging and performance tuning.
B7	Performance	It is an indication of the responsiveness of a system to execute any action within a given time interval.
B8	Scalability	It is ability of a system to either handle increases in load without impact on the performance of the system.
B9	Supportability	It is the ability of the system to provide information helpful for identifying and resolving issues when it fails to work correctly.
B10	Testability	It is a measure of how easy it is to create test criteria for the system and its components, and to execute these tests in order to determine if the criteria are met.
B11	Usability	It defines how well the application meets the requirements of the user and consumer by being intuitive, easy to localize and globalize.

Recoverability

It specifies the capability for components and subsystems to be suitable for use in other applications and in other scenarios. Recoverability minimizes the duplication of components and also the implementation time [LINK7].

Accessibility

It is a property of a system that allow for the widest possible range of users to access the software system's functionality It identifies the proportion of time that the system is functional and working. It can be measured as a percentage of the total system downtime over a predefined period. Accessibility is affected by system errors, infrastructure problems, malicious attacks, and system load [LINK7].

Interoperability

It is the ability of a system or different systems to operate successfully by communicating and exchanging information with other external systems written and run by external parties. An interoperable system makes it easier to exchange and reuse information internally as well as externally [LINK8].

Manageability

It signifies how easy it is for system administrators to manage the application, usually through sufficient and useful instrumentation exposed for use in monitoring systems and for debugging and performance tuning [LINK7].

Performance

It is an indication of the responsiveness of a system to execute any action within a given time interval. It can be measured in terms of Latency or Throughput. Latency is the time taken to respond to any event and Throughput is the number of events that take place within a given amount of time [LINK7].

Scalability

Scalability is the property of reducing or increasing the scope of methods, processes, and management according to the problem size. It is skill of a system to either handle increases in load without impact on the performance of the system, or the ability to be readily enlarged. And, it's an ability for an increasing number of users to easily share a single system [LINK7].

Supportability

It is an inherent capacity of shipped software to allow easier diagnose of any problems in the field. In other words, it means to "be supported" in the area to provide information helpful for identifying and resolving issues when it fails to work correctly [LINK7].

Testability

It is a measure of how easy it is to create test criteria for the system and its components, and to execute these tests in order to determine if the criteria are met. Good testability makes it more likely that faults in a system can be isolated in a timely and effective manner [LINK7].

Usability

It defines degree of application meets the requirements of the user and consumer by being intuitive and easy to localize and globalize. It is conceived as the most general ergonomic quality concept that applies to all kinds of interaction between a user and a product (software) within a given context of use. Historically, the concept of usability is defined in multiple ways such as semantics, features and operations. In Semantic base, usability is equated to user-friendliness, without formal definition of the properties of the construct. In addition, in features base, usability is equated to the presence or absence of certain features in the user interface. Lastly, operations base pursue usability in terms of performance and affective levels manifest by users for certain task [LINK7].

7.2.3 Unvisualized States

The problem with *OBP*s is that it contains numerous invisible flaws that are often located and exploited by assailants to compromise the software's security and aforementioned benefits. Such exploitable flaws are referred to as unvisualized states namely; complexity, variability, ambiguity and uncertainty. These unvisualized states are described as follows:

Complexity(C)

Complexity is the difficulty to maintain, change and understand software. Three specific types of complexity that affect a developer's ability to comprehend software have been identified such as problem complexity, system design complexity, and procedural complexity. Problem complexity is a function of the problem domain. While, System design complexity addresses the mapping of a problem space into a given representation. Structural complexity and data complexity are the two types of system design complexity defined for structured systems. Structural complexity addresses the concept of coupling. Data complexity addresses the concept of cohesion. The complexity of a system is based on the sum of the structural and data complexity for all modules in the system. These measures address information system complexity at the system and module levels. Procedural complexity is associated with the logical structure of a module. This approach to complexity measurement assumes that length of module or the number of logical constructs that determines the complexity [YK+06].

Variability (V)

Variability is the ability of a core asset to be efficiently extended, changed, customized, or configured for use in a particular context. Variability is divided into external variability that is variability of domain artefacts that is visible to users and internal variability is the variability of domain artefacts that is hidden from users. Both external and internal variability are important to the success of developing software system. However, external variability is important as it is visible to users and relates to requirements defined at early development stages where errors or inaccuracies are relatively inexpensive and easy to detect and correct [MJJ06].

Table 7.2: Unvisualized States incurred in *OBPs*

Unvisualized state	**Description**
Complexity(C)	It refers to the variety or diversity of some aspect of a task, such as number and diversity of inputs and/or outputs, number o f separate and different actions or tasks to produce the end product of a project and number of specialties involved on a project.
Variability (V)	It is the ability of an asset to be efficiently extended, changed, customized or configured for use in a specific context.
Ambiguity (A)	It is defined as the inability to recognize and articulate relevant erratic term and their functional relationship.
Uncertainty (U)	It signifies the absence of information about a given jeopardy, which in turn leads to the inability to accurately predict the outcome of a given system

Uncertainty (U)

It is a state of not knowing or not realizing and described as totally unforeseen, not expected to happen and not imaginable. Uncertainty has been categorized along four dimensions such as variation, foreseen uncertainty, unforeseen uncertainty and chaos or turbulence. Variation in activity durations, costs and the exact performance level delivered by resources is a common source of project uncertainty. This implies that the nature and sequence of the relevant activities as well as the objectives of the project are well known. Thus, project plan is detailed and stable but project schedules and budgets exhibit variation around the projected values. However, foreseen uncertainties are identified, but uncertain influences in a project. Unforeseen uncertainty is not formally identified in project planning stage, specifically, it is not anticipated. Chaos or turbulence refers to the fundamental uncertainty about the basic structure of the project plan itself [MF11].

Ambiguity (A)

Ambiguity indicates causing perplexity or indecision. In the same way, during software project development, ambiguity admits more than one possible interpretation. Ambiguity is notably incompatible with the goal of producing deterministic software and covers lexical, syntactic, vagueness and language errors [MGM03].

7.3 Secured Software Environment for *OBPs*

Secured software environment defines software environment that is able to resist most assails, and helps a software to recover quickly with a minimum of damage from the very few assails that it cannot tolerate. There exists three main objectives of assails on software such as these either try to sabotage the software by causing it to fail by modifying it or by executing malicious logic embedded in it. Next, it may become unavailable by subverting software or by changing its operations. Lastly, to learn more about the software's operation and environment so that software can be targeted more effectively. The subversion and sabotage of software always results in the violation of the software's security as well as other benefits associated with the software. Thus, we describe related security attributes as follows:

Role Definition

It is defined as the ability to clearly assign and monitor roles to the people involved in the project. These roles constitute Leader, Member and Contributor. Project team leader is a person provides leadership and guidance to the team and takes responsibility for results of teamwork. Team leader role involves development and encouragement of the team through training, leading, motivation, recognition, rewarding and other activities that stimulate or force team members to do the required tasks. However, a project team member involved in doing assigned tasks. Team members directly access the project and actively evolve its processes and subordinated to the team leader. Lastly, a project team contributor is a person or an organization that participates in teamwork but is not actually involved in performing tasks and carrying out project team responsibilities. Contributors help to improve project through giving valued suggestions, expert judgment and consultation and are not responsible for the project results [MAR03].

Intricacy

Intricacy or project intricacy statement is a means used to describe the major expected outputs of a project including the key milestones, high level requirements, assumptions, and exclusions. Key milestones are zero-duration events that mark progress across the project timeline. Furthermore, high Level requirements are the specifications for the project described at a summary level and usually include a technical description. At last, assumptions and exclusions are the specific disclaimers and decisions about the project that are used to clarify project scope [MF11].

Table 7.3: Required Security Attributes in *OBPs*

S. No.	Security Attributes	Description
1	**Role definition**	It is defined as the ability to clearly assign and monitor roles to the people involved in the project.
2	**Intricacy**	It signifies the magnitude or the scope of the project.
3	**Technology acquisition**	It concerns the sum of new hardware and software, as well as the number of vendors involved.
4	**Resources**	These refer to both finance and effort assets allocated to the project.
5	**Expertise**	It is the ability to define the purpose of the project, and to provide consistent working principles.
6	**Methodology**	It is the set of guidelines and skills required to design, develop and implement the project.
7	**User Support**	It is the amount represented by a combination of level of user enthusiasm, user preparedness for the new system, and level of user feedback.
8	**User Experience**	It consists of a blend of tacit knowledge and historical project repositories relevant to the current project.
9	**User Conflicts**	It considers poor communication or hostility between user and designer team members.
10	**Size**	It refers to the number people involved in the project.
11	**Review**	It illustrates the set of precautionary, threat based and leverage based schemes.
12	**Personnel Changes**	It is the degree of dynamicity in involved IS development personnel.

Technology Acquisition

Technology acquisitions provide technological inputs to the acquiring system. Accordingly, potentially expand the acquirer's knowledge base and provide scale, scope, and recombination benefits. However, technology acquisitions entail a disruption in organizational routines. Further, this disruption is most likely in the set of routines that are closest to the technological subsystem of system. Thus, technological acquisitions can also have a negative impact on the innovation output of the acquirers. On balance, assessing whether technological acquisitions have a positive or negative impact on post acquisition innovation output is likely to depend upon the quantity and nature of knowledge elements that they bring to the acquiring system [LINK7].

Resources

These refer to both finance and effort assets allocated to the project such as time, human resources, computer resources and money. Resources can be viewed from three standpoints such as availability, elastic and plastic, shared and dedicated. Available resources include human resources that are available in the same quantity day-after-day and spending these resources does not deplete them. Next, elastic Resources imply the supply can be increased or decreased such as human resources and money whereas plastic resources supply cannot be extended such as time. Lastly, shared resources are needed for short durations but utilized for the entire duration of the project such as DBAs and functional specialists etc. and dedicated resources should be dedicated to the project for the required duration such as computer systems, and programmers [MF11].

Expertise

It is the ability to define the purpose of project, and to provide consistent working principles. It makes available with the analysis of the existing functional and operational aspects along with technological and architectural analysis. Also, it supports in writing of the mock-up process, security recommendations and requirements. Besides, it is assistance to the definition and to the enhancement of the physical security (access) in accordance with the different types of involved people and security areas. Consequently, it holds up to the implementation of a standard regarding to the nomenclature and the management of the different equipment of the information system [RHT+01].

Methodology

It is a structure or a plan that controls process of developing information system and includes the pre-definition of specific deliverables and artefacts that are created and completed by a project team to develop or maintain an application. One software development methodology is not necessarily suitable for use by all projects. Each of the available methodologies is best suited to specific kinds of projects, based on various technical, organizational, project and team considerations [RK11].

User Support

It is the amount represented by a combination of level of user enthusiasm, user preparedness for the new system, and level of user feedback. User support provides application program supported by performing installation, configuration, and software maintenance of existing application programs. Subsequently, it offers software technical support and troubleshoots application program problems. In addition, it develops documentation for all new and modified software programs and conducts user training and develops user documentation for application and system operating manuals [SFO03].

User Experience

It consists of a blend of tacit knowledge and historical project repositories relevant to the current project. User experience considers wider relationship between the product and user in order to investigate the individual's personal experience of using it. It encompasses all aspects of a product that users experience directly and perceive, learn, and use including its form, behaviour and content. Learnability, usefulness, and aesthetic appeal are key factors in users' experience [SFO03].

User Conflict

User conflict is inevitable in a project environment as change seems to be expected. When project team members interact during the course of completing the tasks and responsibilities, there always exist a potential for conflict. In fact, it is virtually impossible for people with diverse background skills and norms to work together, make decisions, and try to meet project goals and objectives without conflict. Over the years, three distinct views have evolved about conflict in projects and organizations. Traditional view assumes that conflict is dreadful, always has a negative impact, and leads to declines in performance as the level of conflict increases. However, behavioural view also known as human relations view advocates acceptance of conflict and rationalizes its existence. Lastly, interactionist view assumes that conflict is necessary to increase performance. Interactionist view encourages conflict based on the belief that a harmonious, peaceful, tranquil, too-cooperative project organization is probable to become static and unable to respond to change and innovation [SFO03].

Size

Size refers to an indication of the overall effort to be expended or the number of people working on the project. Software projects have been divided into large, medium, and small size. Major differences between project size determined by the estimated total labour hours (the level of effort) required to complete the project. Next, the use of cutting edge or existing technology and type and extent of both user and system interface requirements. The project's contribution to, and impact on, the activities carried out by the system users and other departmental organizations [PM95].

Review

A review in software engineering domain constitutes formal and informal reviews. The attributes of these two classes are controls, group dynamic, and procedures. A review without any controls or defined procedures is informal. The majority of the reviews discussed in software engineering standards and guidelines are formal. To be formal, a review must be systematic. Formal review includes defined entry and exit criteria, a definite list of participant roles, documented procedures for required output documents [PM95].

Personnel Change

Personnel referred to the people employed on a software system and Personnel Change is the degree of dynamicity in involved system development personnel. In other words, it is the transfer of responsibilities from owner person/ group to another person/ group responsible for completing the work [PM95].

7.3.1 Abstraction Method (*AM*)

Abstraction Method *(AM)* provides to software acquirers and users the justifiable confidence that software will consistently exhibit security even when the software comes under assail. *AM* enables formalization of security attributes with respect to set of benefits associated to various perspectives and unvisualized states associated with these benefits. Consequently, *AM* is a step towards increasing recognition and applicability to the range of *OBPs.* The step wise procedure is as follows:

Step I: *Identification of benefits with various perspectives.*

Firstly, benefits must be identified associated to various perspectives as shown in Table 7.4.

Step II: *Mapping the benefits concerning each unvisualized state.*

Then, benefits are mapped to each unvisualized state such as complexity, variability, ambiguity and uncertainty.

Step III: *Finding security attributes to ensnare unvisualized states.*

Various security attributes such as role definition, intricacy, technology acquisition and resources etc. are identified to trap these unvisualized states.

Step IV: *Allocation of security attributes to acquire secured environment.*

Security attributes are allocated to remove the unvisualized states create secured environment.

Table 7.4: Perspective-Wise Benefits

Perspectives	Benefits Associated
Generality Engineering	Scalability
	Supportability
	Usability
	Conceptual Integrity
	Interoperability
	Accessibility
Requirement Engineering	Scalability
	Supportability
	Usability
	Performance
	Accessibility
	Manageability
Reuse Engineering	Scalability
	Usability
	Conceptual Integrity
	Interoperability
	Performance
	Maintainability
	Recoverability
	Manageability
Reliability Engineering	Scalability
	Usability
	Conceptual Integrity
	Performance
	Testability

7.3.2 Case Study

In this section, we have considered *OBP*s to analyze the execution of *AM*. Our study concentrated on two aspects of security establishment; firstly impact of unvisualized states incurred in *OBPs* due to aforesaid benefits and secondly, impact of security attributes on these unvisualized states. In this section, we first describe the research setup for our case study. *OBP*s are categorized on the basis of various development perspectives. Category I consists of *OBP*s developed with generality perspective whereas Category II possesses *OBP*s emphasized on requirement engineering perspective. Category III contains *OBP*s build with reusability perspective and Category IV includes *OBP*s having reliability perspective. Table 7.5 covers 57 *OBP*s from four aforesaid categories for the study of *AM*. Description of projects is illustrated in Appendix7.1.

It has been found during the rigorous study of various *OBPs* that the functional characteristics of associated benefits affect the degree of dependency on unvisualized state. We have studied the dependencies of each benefit on each kind of unvisualized state in exhaustive manner as illustrated in Table 7.5. For example, in first project P1 developed with generality perspective, we have identified various benefits such as conceptual integrity, accessibility, interoperability, scalability, supportability and usability. And, all these benefits lead to unvisualized states that affect project security. We have recognized security attributes such as role definition, resources, expertise, user support and review standards to ensnare unvisualized states and hence retain the security. On the other hand, project P12 developed with requirement engineering perspective includes benefits namely; conceptual integrity, manageability and scalability and again all unvisualized states incurred. To overcome their effect and protect the security we incorporate resources, user support, user conflicts, review standards and personnel changes security attributes. Accordingly, we allocate different security attributes to aid the project security and hence remove unvisualized states caused by various benefits associated with aforesaid perspectives. Figure 7.1 shows the secured environment achieved accordingly.

Table 7.5: Categorization of Benefits &Associated Unvisualized States **(Contd ...)**

Category	Project	Benefits											Unvisualized States			
		B1	*B2*	*B3*	*B4*	*B5*	*B6*	*B7*	*B8*	*B9*	*B10*	*B11*	C	V	A	U
Category I	P1	1	0	0	1	1	0	0	1	1	0	1	1	1	1	1
	P2	0	0	0	0	0	0	0	0	1	0	1	1	1	1	1
	P3	1	0	0	0	1	0	0	1	0	0	0	1	1	1	1
	P4	1	0	0	0	1	0	0	1	1	0	1	1	1	1	1
	P6	0	0	0	1	0	0	0	0	0	0	0	1	1	1	1
	P7	0	0	0	1	0	0	0	0	1	0	1	1	1	1	1
	P8	1	0	0	1	1	0	0	1	0	0	0	1	1	1	1
	P9	1	0	0	1	1	0	0	1	1	0	1	1	1	1	1
	P10	1	0	0	0	0	0	0	0	0	0	1	1	1	1	1
Category II	P11	0	0	0	0	0	0	1	0	1	0	1	1	1	1	1
	P12	0	0	0	0	0	1	0	1	0	0	0	1	1	1	1
	P13	0	0	0	0	0	0	1	1	1	0	1	1	1	1	1
	P14	0	0	0	1	0	1	0	0	0	0	0	1	1	1	1
	P15	0	0	0	1	0	1	1	0	1	0	1	1	1	1	1
	P16	0	0	0	1	0	1	0	1	0	0	0	1	1	1	1
	P17	0	0	0	1	0	1	1	1	1	0	1	1	1	1	1
	P18	0	0	0	1	0	0	0	1	0	0	1	1	1	1	1
	P19	0	0	0	0	0	0	1	0	1	0	1	1	1	1	1
	P20	0	0	0	0	0	0	0	1	0	0	0	1	1	1	1
	P21	0	0	0	0	0	0	1	1	1	0	1	1	1	1	1
	P22	0	0	0	1	0	1	0	0	0	0	0	1	1	1	1
	P23	0	0	0	1	0	1	1	0	1	0	1	1	1	1	1
	P24	0	0	0	1	0	1	0	1	0	0	0	1	1	1	1
	P25	0	0	0	1	0	1	1	1	1	0	1	1	1	1	1

Category	Project	Benefits											Unvisualized States			
		B1	***B2***	***B3***	***B4***	***B5***	***B6***	***B7***	***B8***	***B9***	***B10***	***B11***	**C**	**V**	**A**	***U***
Category III	P27	0	1	0	0	0	0	1	0	0	0	1	1	1	1	1
	P28	1	1	1	0	1	0	0	1	0	0	0	1	1	1	1
	P29	1	1	1	0	1	0	1	1	0	0	1	1	1	1	1
	P30	0	1	0	0	0	1	0	0	0	0	0	1	1	1	1
	P31	0	1	0	0	0	1	1	0	0	0	1	1	1	1	1
	P32	1	1	1	0	1	1	0	1	0	0	0	1	1	1	1
	P33	1	1	1	0	1	1	1	1	0	0	1	1	1	1	1
Category IV	P34	0	0	1	0	0	0	1	0	0	1	0	1	1	1	1
	P35	0	0	0	0	0	0	1	0	0	1	1	1	1	1	1
	P36	0	0	1	0	0	0	0	1	0	0	0	1	1	1	1
	P37	0	0	1	0	0	0	1	1	0	1	1	1	1	1	1
	P38	0	0	0	0	0	0	0	0	0	0	0	1	1	1	1
	P39	0	0	0	0	0	0	1	0	0	1	1	1	1	1	1
	P40	0	0	1	0	0	0	0	1	0	0	0	1	1	1	1
	P41	0	0	1	0	0	0	1	1	0	1	1	1	1	1	1
	P42	0	0	0	0	0	0	0	0	0	0	0	1	1	1	1
	P43	0	0	0	0	0	0	1	0	0	1	1	1	1	1	1
	P45	0	0	1	0	0	0	0	1	0	0	0	1	1	1	1
	P46	0	0	1	0	0	0	1	1	0	1	1	1	1	1	1
	P48	0	0	0	0	0	0	0	0	0	0	0	1	1	1	1
	P49	0	0	0	0	0	0	1	0	0	1	1	1	1	1	1
	P50	0	0	1	0	0	0	0	1	0	0	0	1	1	1	1
	P51	0	0	1	0	0	0	1	1	0	1	1	1	1	1	1
	P52	0	0	0	0	0	0	1	1	0	1	0	1	1	1	1
	P54	0	0	0	0	0	0	1	0	0	1	1	1	1	1	1
	P55	0	0	1	0	0	0	0	1	0	0	0	1	1	1	1
	P56	0	0	1	0	0	0	1	1	0	1	1	1	1	1	1
	P57	0	0	0	0	0	0	0	0	0	0	0	1	1	1	1

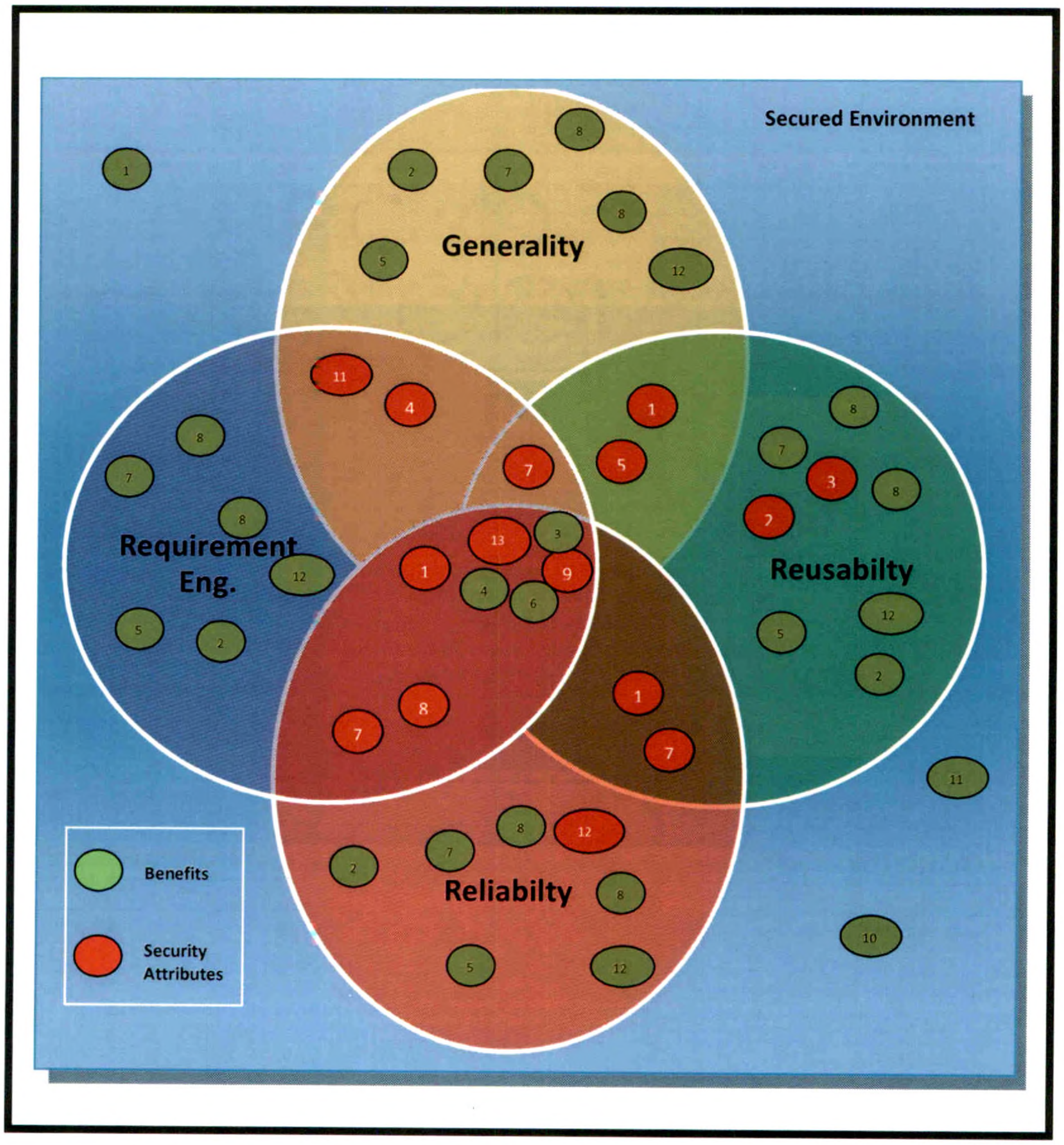


Figure 7.1: Secured Environment for Ontology Based Projects

7.4 Results

With the help of case study, the performance of *AM* has been analyzed on the basis of allocation of security attributes with the associated benefits. Some observations are as follows:

- Complexity is incurred due to accomplishment of benefits such as scalability, maintainability, testability and manageability and it may be incised once intricacy, technology acquisition and resources are well engrossed.
- Variability is attributable to the benefits such as supportability and recoverability but user support, user conflicts and reviews may help to ease variability.
- Interoperability, usability and accessibility benefits achievement lead to ambiguity and inclusion of role definition, methodology and user experience diminishes ambiguity.
- Uncertainty is owing to benefits such as conceptual integrity and performance and it may be reduced if size, personnel changes and user experience are reckoned.

7.5 Summary

Software security is a demanding task in case of Ontology Based Projects (*OBPs*) due to the side effects caused by inherent unvisualized states such as complexity, variability, ambiguity and uncertainty. In this view, we have discussed *OBPs* developed using various perspectives. These perspectives include generality, requirement engineering, reusability and reliability. In next Subsections, benefits associated with these perspectives and unvisualized states incurred due to these benefits have been delineated. Next, we have proposed secured software development environment for *OBPs* with various perspectives. Consequently, we present Abstraction Method *(AM)* for developing the secured environment and case study in subsequent Subsection. It has been noticed that the influence of kinds of benefits associated with each perspective leads to different unvisualized state. Thus, the security attributes corresponding to these perspectives have been allocated to ensnare the kinds of unvisualized state accordingly. Finally, *AM* provides analytical scheme to acquire secured environment for different *OBPs*.

Software Projects

1. Hospital Management System
Objective: To computerize the front office management of hospital.
Technology Used: VB 6.0, MS Access
Brief Description: It includes registration of patients, storing their details into the system, and also computerized billing in the pharmacy, and labs. It has the facility to give a unique id for every patient and stores the details of every patient and the staff automatically. It includes a search facility to know the current status of each room. User can search availability of a doctor and the details of a patient using the id.

2. Railway Reservation System
Objective: To provide railway reservation, fare recordings, train and ticket enquiry, and seat details management.
Technology Used: VB 6.0, MS Access
Brief Description: This project is designed to reduce the problem faced while making railway reservations. It helps in the reservation of tickets as per one's choice, wish, simplicity and convenience. It also helps to enable private ticket booking in a secured and authenticated manner. To book tickets through this system, a customer first registers in system. Whenever a user logs-in it is checked whether he is a registered customer in the customer database. A valid user then gives the details of his travel. Our application then determines and displays to the user the list of trains according to his demands from the train, stations and seat databases. A user then books the tickets.

3. Library Management System
Objective: To support the general requirement of the library like acquisition, cataloguing, circulation management.
Technology Used: Servlets, HTML, JavaScript, MS Access, Apache Tomcat Server
Brief Description: This software application manages the student details, employee details, books details. This system deals with the books issue, book return operations, and fine calculations. This also takes care for security options. This application enables you to enter details of new adding books in the institute and also enables you to modify and delete records of books.

4. Bank Automation System
Objective: To manage the Automation of the banking system this will take care of the Accounting information and Transactions.
Technology Used: Microsoft Visual Basic 6.0, Microsoft Access Database

Brief Description: The system is developed for automatic interaction of all bank departments, efficient data collection, processing and management, easy integration into other bank systems / document management systems. It provides global environment for all departments (legal, marketing, logistics, executive) and extensive flexibility to support the localities and specifics of bank operations in different countries.

5. Recycling Machine System
Objective: It controls a recycling machine for returnable bottles, cans and crates
Technology Used: VB. Net, SQL server
Brief Description: There exists different types and sizes of bottle and can, the system has to check for each item that has been returned. The system operator registers number of items returned by each customer and operator give receipt to customer for value of retuned items.

6. Vending Machine System
Objective: To instruct the system to serve the beverage.
Technology Used: VHDL
Brief Description: The system is designed to activate the machine when the user inserts a five rupee coin into coin slot. This coin will be detected by an IR sensor and send a signal to the micro controller. The machine comprises of cylinder controlled by micro controller. A fixed beverage is filled in the main container. The beverage is poured in the glass through tap which opens and closes after fixed time period and only activated when container is filled. Hence, the user gets the beverage demanded by him by fully automated technique.

7. Telecom Management System
Objective: To manage automation of the management of Telecom services, this involves customer applications, entries, and enquiries, queries and complaints.
Technology Used: Java (JDK), Servlets (JSDK), JSP, Tomcat Server, Oracle Server, JDBC driver
Brief Description: It provides automation of the Telecom System and processes all the activities through online. Here the main advantage of this system is to access this database globally for users. The customers see their connection status at any branch. Here dynamically generated the reports like previous details of the customer. The main advantages of this system are to reduce the time and also manpower

8. Web Alumni System
Objective: To make a connection between alumni.
Technology Used: JavaScript, PHP
Brief Description: This project is aimed at developing a repository for alumni of the college. Anyone can access the search engine to know about any alumni of that college but can't be added. Alumni can only update the database when they are in college.

9. Airline Information System
Objective: To reserve seats for its customers, maintain information and also update the database.
Technology Used: VB, MS Access
Brief Description: This project is based on the 2-tier architecture. The Project is developed keeping in mind the security needs of today. The purpose of the Airline Information System Project is to build an application program, which an airline could use to manage the reservation of airline tickets. Passengers make flight reservations through the ticketing staff of the airline, which can access a centralized system to check on flight details. The system able to create flights, delete flights and reserve seats for passengers according to their requested Destination, day and time.

10. Online Shopping System
Objective: To manage purchasing and selling goods online.
Technology Used: HTML, JavaScript, SQL 7.0, JSP, JDK 1.2.2
Brief Description: The objective is to create a system which is used to purchase and sell goods online i.e. through 'internet'. The entire system is developed to meet the requirements of the organization. The whole system is designed in such a way that it contains the entire information required for purchasing and selling goods online. It is developed with a valid login id and password. The entire system is built taking care of user friendliness and security. The above system is Flexible and Efficient and facilities all the users.

11. Online Examination System
Objective: To conduct an examination through internet.
Technology Used: ASP, MS Access
Brief Description: This Web Application provides facility to conduct online examination worldwide. It saves time as it allows number of students to give the exam at a time and displays the results as the test gets over, so no need to wait for the result. It saves time as it allows number of students to give the exam at a time and displays the results as the test gets over, so no need to wait for the result. It is automatically generated by the server. Student can attend exam by entering login id and password.

12. ATM System
Objective: To allow the user to create an account, deposit, withdraw, view his/her account status.
Technology Used: Visual Basic 6.0, MS Access
Brief Description: The ATM is the project which is used to access their bank accounts in order to make cash withdrawals, recharge their mobile phones and booking their air tickets. Whenever the user need to make the cash withdraws, they can enter their PIN number (Personal Identification Number) and it will display the amount to be withdrawn. Once the withdrawal is successful, the amount will be debited in their account.

13. Hotel Automation System
Objective: To maintain the details of customer booking information, trace the details of customer also maintain the information about the hotel room availability.
Technology Used: Microsoft Visual Studio 2005, C#.Net, Microsoft SQL Server 2005
Brief Description: The system provides all the details of availability of rooms and dates for booking in advance. The Hotel Management System provides features like Check In details of customer, customer booking details, Check Out details, searching the details of customer.

14. Result Generation System
Objective: To maintain a database of results of students and produce them.
Technology Used: VB 6.0, MS Access
Brief Description: The system is designed such that it can save all the required data of a student in a database and this database also contain the results of individual students. This system can produce the information of student along with its result.

15. Music School Automation System
Objective: To maintain records of the teachers, students and staff of Music school.
Technology Used: Microsoft Visual Studio 2005, C#.Net, Microsoft SQL Server 2005
Brief Description: The proposed system will automate the current features with an administration database and graphical user interfaces. It will be implemented to cover the requirements of the Principal of the school like maintaining a database for the staff, teachers and students. This product will be used in conjunction with a web-browser.

16. Mass Transit System
Objective: The system manages all the transit mechanism digitally.
Technology Used: VB, MS-Access
Brief Description: It first classify the kinds of transit means such as airways, railways or roadways. Then, it optimizes the routes according to customer's need that is time, route any type of transit means.

17. Power Utility System
Objective: It optimizes the power utility services.
Technology Used: VB, MS-Access
Brief Description: The system includes optimization of power sources such as hydro-power, hydel power, and nuclear power. It manages the power generation, power supply, power control and finally consumer billing.

18. Student Event System
Objective: To manage student events and athletics of a particular institution.

Technology Used: HTML, JavaScript, SQL 7.0, JSP, JDK 1.2.2
Brief Description: The project includes a multi-faceted system for student event ticket distribution. A process for students to login and reserve a ticket for any sporting event, performance, or activity will be developed. This part of the project will have several sections, a web application for the online reservation, a standalone application for information desk and possible kiosk reservations, and an on-site check-in and reservation retrieval system.

19. Personal Investment System
Objective: To help the user keep account of his/her money invested in institutions such as Banks and Share Market.
Technology Used: VB 6.0, MS Access
Brief Description: The system is aimed towards a person who has considerable number of investments in stock market and banks, and so needs software assistance for book keeping and computations regarding the investments. Personal Investment System is user-friendly, 'quick to learn' and reliable software for the above purpose

20. Electronic Voting System
Objective: The purpose of this system is to implement the computerization of the details, Polling results, current Polling and candidates.
Technology Used: PHP 5.0, My SQL 5.2
Brief Description: The system is supported through the college Student Records System. This allows students to login and be authenticated as an entitled voter for election purposes. The Electronic Voting System is recognized as secure and accurate in its ability to authenticate students and to tabulate voting results.

21. Disease Analysis System
Objective: To develop a software for doctors and patients for immediate diagnose disease.
Technology Used: Servlets (JSDK), JSP, Tomcat Server, Oracle 9i
Brief Description: The system is developed for doctors and patients for immediate diagnose disease using tools which was usually done through manual process. This system will be used to quickly find out the disease and generate reports on about the patient status which will be useful for further understanding to deal with the case.

22. Credit Ranking System
Objective: It rates the credibility of a customer.
Technology Used: VB .Net, SQL server
Brief Description: The system is works for ranking the customers credibility by assessing his or her track records against the financial liabilities. It manages all related operations that vary with type of customer such as service or business class.

23. Intelligent Air-traffic Control System
Objective: To immediate diagnose disease using tools which was usually done through manual process.
Technology Used: VB 6.0, MS Access
Brief Description: The primary purpose of Air Traffic Control systems worldwide is to separate aircraft to prevent collisions, to organize and expedite the flow of traffic, and to provide information and other support for pilots when able. Air Traffic Control System may also play a security or defense role and be run entirely by the military.

24. Intelligent Home Control System
Objective: To control different types of home appliances by using a device.
Technology Used: Servlets (JSDK), JSP, Tomcat Server, Oracle Server
Brief Description: The system allow the users to access their home appliances by using internet either through PC or mobile phones. In this system, users can use Bluetooth transmitter through which they can build a communication between all the devices at home and can access all the appliances at a time.

25. E-Recruitment System
Objective: To provide platform to job seekers to upload their resumes and apply for jobs they wish to join.
Technology Used: Servlets (JSDK), JSP, Tomcat Server, Oracle Server
Brief Description: The system is designed as an online website wherein jobseekers can register themselves at the site, apply for the job and attend exams. This system can be implemented at a global level. The company can post their staffing requirements and check the profiles of various candidates. These recruitment methods are designed to get the best candidates for the job.

26. E-Medicine System
Objective: To manage purchasing and selling medicines online.
Technology Used: HTML, JavaScript, SQL 7.0, JSP, JDK 1.2.2
Brief Description: The objective is to create a system which is used to purchase and sell medicines online. The entire system is developed to meet the requirements of the organization. The whole system is designed in such a way that it contains the entire information required for purchasing and selling medicines online. Any individual can order any amount of medicines.

27. Inter ware House System
Objective: It manages the operations between the warehouses.
Technology Used: ASP .Net, Oracle 9i
Brief Description: The system is responsible for redistribution between different warehouses. Various people are responsible for carrying out different processes such as foreman is responsible for warehouse management. While, warehouse worker works in a warehouse for loading and unloading. Subsequently, truck driver is accountable for transportation and forklift operator drives a forklift in one warehouse.

28. Hydrology Plant Automation System
Objective: To manage data records of hydrologic plant.
Technology Used: VB 6.0, MS Access
Brief Description: The system is developed to automate the manual working over the records of the hydrologic plant. It provides the data to be saved in a particular database so that it can be used for further studies.

29. Online Student Feedback System
Objective: To maintain the quality of lectures by taking feedbacks of students.
Technology Used: HTML, JavaScript, SQL 7.0, JSP, JDK 1.2.2
Brief Description: The system is developed for the main evaluations done at any institution by getting an anonymous student feedback at the end of the semester and getting an overall summary of the students' viewpoints regarding the lecturer's teaching.

30. Quiz System
Objective: The system is developed to evaluate student's knowledge.
Technology Used: HTML, JavaScript, SQL 7.0, JSP, JDK 1.2.2
Brief Description: To randomize the selection of questions for each level based on the difficulty category. The system shall be smart enough to determine the difficulty of the question based on the user responses, meaning that higher wrong answer percentage increase the question difficulty and hence the question is selected in higher levels.

31. Rail route Optimization System
Objective: To show graphical representation of train route from starting point to ending point.
Technology Used: VB 6.0, MS Access
Brief Description: The railway route optimization system is a product to serve the end users or passengers to know the shortest distance in which they can reach their destination in shortest period of time with minimum amount, if there is one or more route to the station then the optimization system will show the graphical representation of the route. From this end users can access from anywhere in the world.

32. Student Evaluation System
Objective: To evaluate the overall performance of student.
Technology Used: VB 6.0, MS Access
Brief Description: The application deals with the student personal information, college records, such as their subjects and their grades, summary reports for enrollees, curricula, course and management of the said records. This application is based on schools basic student information and record keeping and tracking. The system being developed will aim to automate the whole system as it progresses its development.

33. Insurance Transaction Monitoring System
Objective: This SOA based application can be used by an insurance company to maintain the insurance management, daily transactions, and policy registration.
Technology Used: JAVA, Oracle 9i
Brief Description: It keeps account of agents, policies, premiums. Also, generates the monthly, quarterly, half Yearly and yearly premiums. In addition monitors the Agents commission management and branch transaction details.

34. Health Care Service System
Objective: To provide medical services to the patients around, it can be used for maintaining patient details and their test results
Technology Used: ASP. Net, MS Access
Brief Description: This SOA-based Software is for the automation of health care management. Maintaining patient details, providing prescription, precautions and diet advice, providing and maintaining all kinds of tests for patients, billing and report generation.

35. HR Service Outsource System
Objective: Providing with skilled professionals who are focused specifically on HR to help to reduce and manage operating costs and for improving employee relations.
Technology Used: ASP. Net, MS Access
Brief Description: Overseeing organizational structure and staffing requirements recruiting, training, and development Tracking department objectives, goals, and strategies Employee and manager training Benefits administration Employee orientation programs It keeps account of agents, policies, premiums. Also, generates the monthly, quarterly, half Yearly and yearly premiums. In addition monitors the Agents commission management and branch transaction details.

36. Real Estate Management System
Objective: To manage the residential and commercial real estate development through the complete sales cycle starting from the development phase to the post possession phase.
Technology Used: JAVA, Oracle 9i
Brief Description: The system functionalities start with pre sales management then post sales management to customer Service. Pre sales management has four phases for construction management, inventory management, broker management and customer enquiry management. Post sales management concerned with billing, collection, recovery and advertising. Consequently, customer service has three sections such as customer complains management, customer portal and marketing management.

37. Security Architecture Blueprint Service System
Objective: To carry out the intent of the enterprise risk management, security policy and standards, and security architecture.
Technology Used: JAVA, SQL server

Brief Description: The purpose of the security architecture blueprint is to bring focus to the key areas of concern for the enterprise, highlighting decision criteria and context for each domain. Identity management deals with the creation, communication, recognition, and usage of identity in the enterprise. Identity management includes provisioning services, directories, multi-factor authentication, federation, and so on. All access control is predicated on identity, a central concern to security architecture. Threat management deals with the threats to systems such as virus, Trojans, worms, malicious hackers, force majeure, and intentional and unintentional system misuse by insiders or outsiders.

38. DSpace
Objective: To provide digital library system that captures, stores, indexes, preserves and redistributes the intellectual output of an organization's researchers in digital formats.
Technology Used: Visual Basic, Oracle 9i
Brief Description: The system enables institutions to capture and describe digital works using a custom workflow process such as distribute an institution's digital works over the web, so users can search and retrieve items in the collection and preserve digital works over the long term. DSpace system provides a way to manage these research materials and publications in a professionally maintained repository to give them greater visibility and accessibility over time.

39. Daily Sales Reporting Data ware housing System
Objective: analyze sales of major brands varying with different promotional schemes.
Technology Used: Oracle 9i, J2SE
Brief Description: The system has been developed for a corporate food store company, which is one of the organizations that sell various numbers of products every day. This company has more branches in various locations, which maintains database has lots of previous customer details, company personal information, raw materials details, etc.

40. Internet Banking System
Objective: This Project investigates the entry threshold for providing a new transaction service channel via the real options approach.
Technology Used: J2EE, SQL server
Brief Description: It Provides online banking for Balance Enquiry, Funds Transfer to another account in the same bank, Request for cheque book/change of address/stop payment of cheques and viewing monthly and annual statements. An Internet banking system designed for the use of normal users (individuals), industrialists and entrepreneurs.

41. Portfolio Management System
Objective: To keep the security, safety of Principal sum intact both in terms of money as well as its purchasing power.
Technology used: J2EE, SQL server

Brief Description: The major activities involved in portfolio management starts with identification of assets or securities, allocation of investment and also identifying the classes of assets for the purpose of investment. Then, it helps in deciding the major weights, proportion of different assets in the portfolio by taking in to consideration the related risk factors. Finally, selects the security within the asset classes as identify.

42. e Tax portal
Objective: simplify and streamline all the corporate tax processes.
Technology used: JAVA, SQL server
Brief Description: A web-based solutions for document management and task delegation to audit management and enhanced communication, eTaxPortal will provide with valuable tax process automation. It ensures consistent business processes and improves productivity with an organization and facilitates collaboration and information sharing among various groups involved in tax function.

43. Customer Invoicing System
Objective: To calculate all applicable charges and to generate all inclusive invoices to the customers online.
Technology used: ASP .net, Mysql
Brief Description: The purpose of this project is to automate the invoicing process as much as possible, leading to a much quicker invoicing of the charges and a more inclusive invoicing that relies on limited to no manual input.

44. Integrated Benefits Administration System
Objective: To process monthly retirement payments, benefit enrolments, new retirements, refund requests, insurance premiums and retirement contributions.
Technology used: JAVA, SQL server
Brief Description: It is a web-enabled self-service functionality for providing ease of use not only to its users, but also to members, retirees, beneficiaries, other plan participants, 3rd party vendors, and employers. In addition, all functionality will be available to each of the plans administered as appropriate for the plan (e.g., handling of member and employer payments and receipts and refunds of over payments, correspondence generations, imaging, maintenance of address information, etc.).

45. Sales Force Management System
Objective: To help to automate sales and sales force management functions.
Technology used: ASP .net, Oracle 9i
Brief Description: It records all the stages in a sales process. It includes a contact management system which tracks all contact that has been made with a given customer, the purpose of the contact, and any follow up that might be required. It also includes a sales lead tracking system, which lists potential customers through paid phone lists, or customers of related products. Other elements of SFMS include sales forecasting, order management and product knowledge.

46. Online Personnel Rehabilitation system
Objective: To improve the quality of life of people with disability / marginalized persons.
Technology used: J2EE, SQL server
Brief Description: It is an online portal for prevention of cause of disability, provision of care facilities, creating a positive attitude towards people with disabilities, provision of functional rehabilitation services, empowerment, provision of education and training opportunities, creation of micro & macro income –generation opportunities and management / monitoring and evaluation.

47. Student Admission System
Objective: To work for an institute conducting a professional course.
Technology Used: JAVA, SQL server
Brief Description: It supports the student admission and registration process, maintenance of student personal, academic and fee related data. Database maintained by this system usually contains the student's personal, academic and its fee related information. It focuses on storing and processing (insertion, updation etc.) by using web pages. Generate Student's Academic Detail Report, Personal Detail Report. It stores Merit list provided by University.

48. Safety Management System
Objective: To provide a complete solution for all areas of enterprise safety management.
Technology Used: JAVA, SQL server
Brief Description: It record, track, report and respond to safety incidents while proactively identifying potential safety hazards and risks. Also, it Conduct accurate root-cause analysis to generate prevention strategies and implement preventative and corrective actions. Streamline reporting processes, corporate policies and workflow.

49. Attendance Management System
Objective: To assess the eligibility of a student on the basis of attendance.
Technology Used: VB .net, SQL server
Brief Description: System is developed for daily student attendance in schools, colleges and institutes. If facilitates to access the attendance information of a particular student in a particular class. The information is sorted by the operators, which will be provided by the teacher for a particular class. This system will also help in evaluating attendance eligibility criteria of a student.

50. Lab Management System
Objective: To develop effective software for maintaining the information relating to the lab details.
Technology Used: Visual Basic and Ms-Access in Windows 98 operating system
Brief Description: This includes information about all lab items, doctors, patients and purchase of this clinic. This system gives generalize, concise and accurate information regarding billing, purchase details, stock etc. This system provides

any type of enquiry such as patient details, stock, purchase details, purchase return details, billing.

51. Online Auctioning Shop for a campus/organization
Objective: To create an Online Auctioning system which can be used to buy and sell articles.
Technology Used: JAVA, mysql
Brief Description: The users of the system can create an item for sale providing the item name, description, an image of the item, minimum bid prize etc. The buyers can bid by providing a bid amount (which should be greater than the previous bid). The system will have an administration module to administer the categories of the shop as well as to block fraudulent users. The administrator will set up the Categories of the items. A category is a logical subdivision of category of similar products (e.g. Furniture, Electronic Accessories, and Books). Administrator Create Categories, Merge Category etc. There will be a Search by which users can search for items up for sale.

52. Profile verification System
Objective: To provide the employee information of all the registered companies to the central server or the area of registration.
Technology Used: ASP .net, Oracle 9i
Brief Description: In the current proposed application each employee is associated or assigned with an employee id that work within the company and the global unique identity number by using which his identity can be traced in the other companies by using the currently designed software. Each employee information containing their employee id with company id, company name, candidate name with initial salary and joining date, relieving and end salary information has to be specified which can be verified by the other companies while recruiting them as experienced candidates. A search engine is provided in the login page help the candidates to search for the required company information like the company profile containing their name, registered number, address, work mode and their achievements.

53. Online Course Portal
Objective: To allow registered users of the system to join a course available in the site and access the materials published for the course.
Technology Used: PHP, Oracle 9i
Brief Description: People can register themselves as students of a course or Faculty for a course. When a person registers himself as a Faculty, an approval mechanism should be triggered which sends an email to the Administrator for approving the person as a Faculty. There will be an admin approval page where admin can approve the faculty members for the course. The course home page contains the title of the course and a brief description. The discussion board for each course where students can interact, an announcement section, which contains the latest announcements, and a course content section which gives the links for the material available for the course has been provided. For faculty members an

extra link for uploading the course content in a zip file format has been given. In addition, mechanism for the faculty members to create a test for the course specifying the test title and a set of multiple-choice questions and duration of time of the test has been allotted.

54. Dealer Ship Management System
Objective: To trace the petrol bunks at dealers.
Technology Used: ASP.NET, Oracle 9i
Brief Description: It manages employee details, bank transaction, balance sheet, monthly sales details, daily sales details, etc. This project makes the Dealer's work easier then manuals.

55. Cloud Operating System
Objective: To concentrate on virtualization of the applications, rather installing into machine
Technology Used: Modified Linux platform
Brief Description: This enables the Operating System to load in no time and connect to the internet instantly. This also provides to the basic Office applications, Audio and video player, image viewer and file manager that manages the files and provide file sharing facility on the cloud. The authentication provides portability along with security, as all our data are stored onto the cloud and can be accessed anywhere with the OS.

56. e-Post Office
Objective: To facilitate with all post office services online.
Technology Used: ASP.NET, MS Access
Brief Description: This is an online application meant to present an advanced post office where the client can buy postcards, stamps, submit couriers/parcels everything as like the normal post office do.

57. Mobile Billing System
Objective: To automate the mobile bill generation.
Technology Used: JAVA, SQL server
Brief Description: The system calculates the mobile bills automatically. It does almost every work which is related to automatic mobile billing connection system via- new connection , customer record modification, viewing customer records & all works related to rate of bills, meter readings in addition to bill calculation and bill generation.

.

CHAPTER 8

Concluding Remarks

This thesis covered some research investigations towards Ontology Based Software Engineering for improvement in various issues of usual software engineering practices such as generality, requirement engineering, reusability, reliability and security. We summarize the following important conclusions:

1. We have mapped phases of Object Oriented Software development Life Cycle (OOSDLC) and Ontology Development life cycle (ODLC) to develop *Ontology Driven Information System (ODIS)*. We have observed that, it enables the developer to reuse and share application domain knowledge using a common vocabulary across heterogeneous software applications. To establish this, we have developed generalized Use Case Model and Object Model during *Ontolysis* phase, generalized Design Model during *Ontodesign* and generalized Implementation Model at *Ontocontation* phase of *ODIS* development. Our investigation revealed that each phase of OOSDLC has very well derived from *ODIS* Life Cycle while developing any information system such as Transaction Processing System (TPS), Management Information System (MIS), Office Automation System (OAS), Decision Support System (DSS) or Expert System (ES). It is concluded that *ODIS* facilitates the developer to concentrate on structure or the domain and task.

2. Requirement Engineering (RE) is promising process and especially draws on with the aim of amenable to analysis, communication, and subsequent implementation. We have reviewed conventional REP models and observed that each REP model have certain lacunas over the former hence there exists no ideal REP model. Ontology Aided

Requirement Engineering *(OntoAidedRE)* model has been introduced in order to enable knowledge driven requirement engineering covering requirement type, practices and suitability. Consequently, we have compared conventional REP models with *OntoAidedRE* on the basis of various project parameters such as Project type, Project size, Project team, Project effort, Project quality, Project prioritized element and Project key element. The study reveals that none of the conventional REP models can accomplish each and every project parameter than *OntoAidedRE.* It has been concluded that, it can be put into practice to overcome the problems of conventional REP models and consequently project parameters can be optimally contrived by adapting *OntoAidedRE.*

3. Ontology based reuse is an emerging aspect and specially used for resolving scalability and heterogeneity issues due to elicit practices. We have attempted to present *P4View* approach to ensure the software scalability and heterogeneity. Accordingly, *Ontop4ViewReuse* framework has been developed based on ontology oriented systematic *P4View* approach for reusing. We have found that, it fits in well to make use of the content of ontologies to a maximal extent depending on their particular domain, task and level of application formality. In addition, to build a common conceptual base characterized by knowledge, Ontology based Reuse Algorithm *(OntoReuseAlgo)* for process planning has been recommended. We observed that, it supports the application from three aspects such as *System Element Classification*, *Ontolayering Principle* and *Knowledge Reuse Scheme* for process planning. Lastly, *Ontological Reuse (OnR)* has been devised from Object-Oriented Reuse (OOR*)*. We have explored that *OnR* achieves lucidness of unclear concepts related with software reuse. Besides, the concepts have been linked rigorously. A significant aspect of *OnR* suggests its independence from implementations or

technological aspects and effectiveness of *OnR* has been suggested in terms of software component, architecture, requirement, process, technology and experience reuse subclasses.

4. Software reliability achievement is a challenging task due to its dependency on users' perspective. We have introduced ontological approach for reliability achievement over object-oriented approach. Then, a comparative analysis has been presented and scope of Ontology Oriented Reliability (*OnO-Reliability*) has been outlined. In addition, ontological specifications have been developed using *OntoReliability* protocol. We have presented some case studies to understand the application of *OntoReliability* protocol for software specification development. Subsequently, the benefits have been discussed. Lastly, we have attempted to quantify the reliability using various project parameters. For the same reason, we have introduced Ontological Reliability Quantification Method (*ORQM*). These project parameters vary in their number and type as per the category of project. Therefore, we have considered the project category as a prerequisite for computing the reliability. We conducted a study of different project case as per the category with varying number and type of parameters and establish the fact that *ORQM* generates direct empirical value for software reliability. Finally, we conclude that *ORQM* is not a informal method but found to be a highly useful in absence of reliability experts and historical failure data.

5. Software security is an important issue that needs to be resolved in case of Ontology Based Projects (*OBPs*) due to the side effects caused by inherent unvisualized states such as complexity, variability, ambiguity and uncertainty. In this view, we have discussed *OBPs* developed using various perspectives such as generality, requirement engineering, reusability and reliability. We have attempted to develop a secured

environment by presenting Abstraction Method *(AM)*. Furthermore, performance of *AM* has been examined on the basis of allocation of security attributes with the associated benefits. It has been noticed that the influence of kinds of benefits associated with each perspective leads to different unvisualized state and *AM* provides analytical scheme to acquire secured environment for different OBPs.

Thus, it involved development of Ontology Driven Information System *(ODIS)* by mapping OOSDLC and ODLC. Subsequently, it entailed Ontology Aided Requirement Engineering *(OntoAidedRE)* model for accomplishment of generalized requirement specification set. Then, software reuse well engrossed by building *Ontop4ViewReuse* framework based on ontology oriented *P4View* approach for reusing and followed by Ontology Based Reuse Algorithm *(OntoReuseAlgo)* to aid product redesign. Next, it leaded to software reliability with Ontology-Oriented Reliability (*OnO-Reliability*) and *OntoReliability* Protocol. In addition, Ontological Reliability Quantification Method *(ORQM)* quantified software reliability. Lastly, it included Abstraction Method *(AM)* for developing secured environment for ontology Based Projects (*OBP*s).

References

[ABH+99] Althoff K., Birk A., Hartkopf S. and Mulle W., “Managing Software Engineering Experience for Comprehensive Reuse”, *Eleventh International Conference on Software Engineering and Knowledge Engineering*, Germany, 1999.

[ALB93] Alberts M., “An ontology for engineering design” *Ph. D. thesis, University of Twent., YMIR,* 1993.

[APM+04] Ashri R., Payne T., Marvin D., Surridge M. and Taylor S., "Towards a Semantic Web Security Infrastructure" In *Semantic Web Services 2004 Spring Symposium Series.* Stanford University, Stanford California, 2004.

[AW06] Assmann U. and Wagner G., “Ontologies, Metamodels, and Model-Driven Paradigm”, In *Ontologies for Software Engineering and Technology*, Springer-Verlag Chapter 9, 2006.

[BAB+05] Biffl S., Aurum A., Boehm B., Erdogmus H. and Grünbacher P. (eds.), “Value-Based Software Engineering”, Springer Verlag, 2005.

[BAR06] Barrasa J., “Ontologies for Software Engineering and Technology”, Springer-Verlag Chapter 11, 2006.

[BAT97] Borst W. N., Akkermans J. M. and J. L. Top, “Engineering ontologies”, *International Journal of Human-Computer Studies, Special Issue on Using Explicit Ontologies in KBS Development*, 1997, Volume 46, pp 365-406.

[BBW96b] Borst, W. N., Benjamin J., Wielinga B. J. and J. M. Akkermans, “An application of ontology construction”, 1996.

[BDP06] Broy M., Deißenböck F. and Pizka M., “Demystifying Maitainability”, *Proceedings of the 4th Workshop on Software Quality. ACM Press. Shanghai*, China, 2006.

[BKK+02] Baclawski K., Koker M. K., Kogut P. A., Hart L., Smith J., Holmes W.S., Letkowski J., Arosen M. L. and Emery P., “Extending the Unified Modelling Language for Ontology

Development", *International Journal of Software and Systems Modelling*, 2002, Volume 1, No. 2, pp 142-156.

[BL03] Breitman K. K., and Leite J. C. S. P., "Ontology as a Requirement Engineering Product," *In Proceedings of the 11th IEEE International Requirements Engineering Conference*, Monterey Bay, California, USA, 2003, pp 309-319.

[BMR94] Bateman, J., Magnini B. and Rinaldi F., "The generalized upper model". *In N. J. I. Mars (Ed.), Working Papers European Conference on Artificial Intelligence ECAI'94 Workshop on Implemented Ontologies*, Amsterdam, ECCAI, 1994, pp 35-45.

[BOE96] Boehm B., "Identifying Quality Requirements Conflicts", *IEEE Software*, 1996.

[BOS95] Bose P., "Conceptual Design Model based Requirements Analysis in Win –Win Framework for Concurrent Requirement Engineering", *IEEE Workshop on Software Specification and Design*, 1995.

[CAC01] Castano S., Antonellies V. and Capoitani Vemercati S., "Global Viewing of Heterogeneous Data Sources", *IEEE Transaction on Knowledge and Data engineering,* 2001, Volume 13, No. 2, pp 277-297.

[CG07] Cortellessa V. and Grassi V., "A modelling approach to analyze the impact of errorpropagation on reliability of component-based systems", *In Heinz Schmidt, Ivica Crnkovic, George Heineman, and Judith Stafford, editors, Proceedings of the 10th International Symposium, CBSE 2007, Lecture Notes in Computer Science*, Springer, 2007, Volume 46, No. 8, pp 140-156.

[CHK+07] Cuenca Grau B., Horrocks I., Kazakov Y. and Sattler U., "A Logical Framework for Modularity of Ontologies", *In: Proceeding of the International Joint Conference on Artificial Intelligence*, IJCAI, 2007.

[CJB98] Chandrasekaran B., Josephson J. R. and Benjamins V., "Ontologies of Tasks and Methods", *In proceedings of KAW'98*, Canada, 1998.

[CJB99] Chandrasekaran B., Josephson J. R. and Benjamins V., "What are Ontologies and Why do we need them?", *IEEE Intelligent Systems*, 1999, Volume 14, No. 1, pp 20-26.

[CLA81] Clarke B. L., "A calculus of individuals based on `connection'", *NotreDame Journal of Formal Logic*, 1981, Volume 22, No. 3, pp 204-218.

[CLV+08] Carlos B., Lasheras J., Valencia-Garcia R., E. Fernandez-Medina, Toval A. and Piattini M., "A Systematic Review and Comparison of Security Ontologies", *In Proceedings of Third International Conference on Availability, Reliability and Security, IEEE Computer Society*, 2008.

[CMI07] Cortellessa V., Di Marco A. and P. Inverardi, "Integrating performance and reliability analysis in a non-functional mda framework", *In Matthew B. Dwyer and Antonia Lopes, editors, Proceedings of the 10th Fundamental Approaches to Software Engineering, Lecture Notes in Computer Science*, Springer, 2007, Volume 44, No. 22, pp 57-71.

[CRC95] Cohn A. G., RandellD. A. and Z. Cui, "Taxonomies of logically defined qualitative spatial relations", *International Journal of Human-Computer Studies*, 1995, Volume 43, pp 831–846.

[DAV90] Davis E. "Representations of Commonsense Knowledge", *San Mateo, California: Morgan Kaufmann Publishers Inc.*, 1990.

[DDM+98] Darimont, R., Delor E., Massonet P. and van Lamsweerde, A., "GRAIL/KAOS: An Environment for Goal-Driven Requirements Engineering", *Proc. ICSE'98 - 20th Intl. Conf. on Software Engineering*, Kyoto, 1998, Volume 2, pp 58-62.

[DER02] Deridder D., "A Concept Oriented Approach to Support Software Maintenance and Reuse Activities", *5th Joint Conference on Knowledge-Based Software Engineering*, 2002.

[DEV02] Devedzic V., "Understanding Ontological Engineering", *Communications of ACM*, 2002, Volume 45, No. 4, pp 136-144.

[DL07] Darimont, R. and Lemoine, M. "Security Requirements for Civil Aviation with UML and Goal Orientation", *Proc. REFSQ'07 – Intl. Working Conference on Foundations for Software Quality*, Trondheim (Norway), LNCS 4542, Springer-Verlag, 2007.

[DLV04] De Landtsheer R., Letier E. and van Lamsweerde A., "Deriving Tabular Event-Based Specifications from Goal-Oriented Requirements Models", *Requirements Engineering Journal*, 2004, Volume 9, No. 2, pp 104-120.

[DR99] Devedzic V. and Radovic D., "A framework for building intelligent manufacturing systems", *IEEE Transactions on Systems, Man, and Cybernetics, Part C: Applications and Reviews*, 1999, Volume 29, No. 3, pp 422-439.

[DV05] De Landtsheer R. and van Lamsweerde A., "Reasoning About Confidentiality at Requirements Engineering Time", *Proc. ESEC/FSE'05*, Lisbon, Portugal, 2005.

[DV96] Darimont R. and van Lamsweerde A., "Formal Refinement Patterns for Goal-Driven Requirements Elaboration", *Proc. FSE'4 - Fourth ACM SIGSOFT Symp. on the Foundations of Software Engineering*, San Francisco, 1996, pp 179-190.

[DVF93] Dardenne A., van Lamsweerde A. and Fickas S., "Goal-Directed Requirements Acquisition", *Science of Computer Programming*, 1993, Volume 20, pp 3-50.

[DW99] Deridder D. and Wouters B., "The Use of Ontologies as a Backbone of Software Engineering Tools", *Fourth Australian Knowledge Acquisition Workshop*, 1999.

[EMB04] Embley D., "Toward semantic understanding: An approach based on information extraction ontologies", *In Proceedings of the Fifteenth Conference on Australasian Database Conference*, Australian Computer Society, 2004.

[EW05] Evermann J. and Wand Y., "Ontology based object oriented domain modelling: fundamental concepts", *Requirements Engineering, Springer-Verlag*, London Ltd., 2005.

[FCF93] Fox M. S., Chionglo J. and Fadel F., "A common-sensemodel of the enterprise", In *Proceedings of the Industrial Engineering Research Conference*, 1993.

[FCM+03] Falbo R. A., Cruz A. C., Mian P. G., Bertollo G. and Borges F., "ODE: Ontology Based Software Development Environment", *IX Argentine Congress on Computer Science*, 2003.

[FEA+02] Fonseca F., Egenhofer M., Agouris P. and Câmara C., "Using Ontologies for Integrated Geographic Information Systems", *Transactions in GIS*, 2002.

[FEN09] Fenz S. and Ekelhart A., "Formalizing information security knowledge," *Proceedings of the 4th international Symposium on information, Computer, and Communications Security*, 2009, pp 183-194.

[FEN10] Fenz S., "Ontology-based generation of IT-security metrics," *Proceedings of the 2010 ACM Symposium on Applied Computing*, 2010, pp 1833-1839.

[FEN11] Fenz S., Ekelhart A. and Neubauer T., "Information Security Risk Management: In which security solutions is it worth investing?", *Communications of the Association for Information Systems*, 2011, Volume 28, pp 329-356.

[FF91] Falkenhainer B. and Forbus K. D. "Compositional modelling: Finding the right model for the job", *Artificial Intelligence*, 1991, Volume 51, pp 95–143.

[FFB+94] Falkenhainer B., Farquar A., Bobrow D., Fikes R. E., Forbus K. D., Gruber T. R., Iwasaki Y. and Kuipers B., "CML: A compositional modelling language. Technical Report Technical report KSL-94-16", *Stanford Knowledge Systems Laboratory*, 1994.

[FFV+98] Feather M., Fickas S., van Lamsweerde A. and Ponsard C., "Reconciling System Requirements and Runtime Behaviour", *In Proc. IWSSD'98 - 9th Intl. Workshop on Software Specification and Design*, IEEE, 1998.

[FGD92] Falbo R. A., Guizzardi G. and Duarte K. C., "An Ontological Approach to Domain Engineering", *In 14th International conference on Software Engineering and Knowledge Engineering*, 1992, pp 351-358.

[FGJ97] Fernandez-Lopez M., Gomez-Perez A. and Juristo N., "METHONTOLOGY: From Ontological Art Towards Ontological Engineering", *Spring Symposium on Ontological Engineering of AAAI*, 1997, pp 33-40.

[FH97] Fridman-Noy N. and Hafner C. D., "The state of the art in ontology design: a survey and comparative review", *AI* Magazine, 1997, pp 53–74.

[FK05] Frakes W. B. and Kang K., "Software Reuse Research: Status and Future", *IEEE Transactions on Software Engineering*, 2005, Volume 31, No. 7.

[FMR98] Falbo R., Menezes C. and Rocha A., "Using Ontologies to Improve Knowledge Integration in Software Engineering environment", *4th International Conference on Information System Analysis and Synthesis*, 1998.

[FOR84] Forbus K. D. "Qualitative process theory", *Artificial Intelligence* 1984, Volume 24, pp 85-168.

[FS94] Falasconi S. and Stefanelli M., "A library of medical ontologies", *In N. J. I. Mars (Ed.), Working Papers European Conference on Artificial Intelligence ECAI'94 Workshop on Implemented Ontologies*, ECCAI, Amsterdam, 1994, pp 81-91.

[FS97] Fowler M. and Scott K. "UML Distilled: Applying the Standard Object Modelling Language", *Addison-Wesley*, Reading, MA, 1997.

[GF95] Gruninger M. and Fox M. S., "Methodology for Design and Evaluation of ontologies", *In Skuce D (ed) IJCAI'95 Wokshop on Basic Ontlogical Issues in Knowledge Sharing*, 1995, pp 6.1-6.10.

[GFC03] Gomez-Perez A., Fernandez-Lopez M. and Corcho O., "Ontological Engineering", *Springer-Verlag*, London, UK, 2003.

[GFC04] Gomez Perez A., Fernandez Lopez M. and Corcho O., "Ontological Engineering", *Springer-Verlag*, 2004.

[GFV96] Gomez-Perez A., Fernandez-Lopez M. and de Vicente A., "Towards a Method to Conceptualize Domain Ontologies", *In ven der Vet P(ed) ECAI'96 Workshop on ontological engineering*, 1996, pp 41-52.

[GH03] Gonzales Perez C. and Henderson-Sellers B., "An Ontology for Software Development Methodologies and Endeavors", *In Ontologies for Software engineering and Technologies Springer-Verlag*, 2003.

[GH95] Gamma E., Helm R., Johnson R. and Vlissides J., "Design Patterns: Elements of Reusable Object- Oriented Software", *Addison-Wesley*, 1995.

[GHJ+95] Gamma E., Helm R., Johnson R. and Vlissides, J., "Design Patterns: Elements of Reusable Object-Oriented Software" *Addison-Wesley*, Reading, MA, 1995.

[GHW02] Guizzardi G., Herre H. and Wagner G., "On the General Ontological Foundation of Conceptual Modelling", *21st International Conference on Concetual Modelling*, 2002.

[GL02] Gurninger M. and Lee J., "Ontology Applications and Design", *Communications of ACM*, 2002, Volume 45, No. 2, pp 39-41.

[GLT98] Gokhale S., Lyu M. and Trivedi K., "Reliability Simulation of Component BasedSoftware Systems", *In Reliability Simulation of Component Based Sofware Systems, Proc. of the 9th ISSRE*, 1998, pp 192-201.

[GOM01] Gomez-Perez A.," Evaluation of Ontologies", *International Journal of Intelligent Systems*, 2001, Volume 16, No. 3, pp 391-409.

[GOM98] Gomez-Perez A., "Knowledge Sharing and Reuse", *In hand Book of Applied Expert Systems*, 1998.

[GPH+05] Goˇseva-Popstojanova K., Hamill M. and Perugupalli R., "Large empirical case study of architecture-based software reliability", *In ISSRE '05: Proceedings of the 16th IEEE International Symposium on Software Reliability Engineering*, Washington, DC, USA, 2005, pp 43-52.

[GPM+01] Goˇseva-Popstojanova K., Mathur A. P. and Trivedi K. S., "Comparison of Architecture-Based Software Reliability Models", *In Proc. of the 12th ISSRE, Elsevier Science*, 2001.

[GPT01] Goˇseva-Popstojanova K. and Trivedi K. S., "Architecture-Based Approach to Reliability Assessment of Software Systems", *In Proc. of the 12th ISSRE, IEEE Computer Society*, 2001, pp 22-31.

[GRU92] Gruber T. R., "A translation approach to portable ontology specifications, Knowledge Acquisition", 1992, Volume 5, No. 2, pp 199-220.

[GS02] Guarino N. and Schneider L., "Ontology Driven Conceptual Modelling", *ER, Preconference Tutorials*, 2002.

[GSC+07] Gauch S., Speretta M., Chandramouli A. and Micarelli A., "User Profiles for Personalized Information Access", *in "The Adaptative Web"*, 2007, Volume 2, pp 54-89.

[GUA04] Guarino N., "Towards Formal Evaluation of Ontology Quality", *IEEE Intelligence Systems*, 2004, Volume 19, No. 4, pp 78-79.

[GUA98] Guarino N., "Formal Ontology in information systems", *In Guarino N(ed) International conference on Formal Ontology in information systems*, 1998, pp 3-15.

[HAM04] Hamza H. S., "Improving Analysis Patterns Reuse: An Ontological Approach", *In Workshop on Ontologies for Software Engineering Artifacts*, 2004.

[HAY85] Hayes P. J., "Naive physics i: Ontology for liquids", *In J. R. Hobbs and R. C. Moore (Eds.), Formal Theories of the Commonsense World*, Norwood, New Jersey: Ablex Publishing Corporation, 1985.

[HKW+04] Hoffmann M., Kuhn N., Weber M. and Bittner M., "Requirements for requirements management tools, Requirements Engineering", *Proc. 12th IEEE International Conference*, 2004, pp 301-308.

[HL01] Hemer D. and Lindsay P., "Specification-based Retrieval Strategies for Module Reuse", *Proceedings Australian Software Engineering Conference*, 27-28 , Canberra, Australia, IEEE Computer Society, 2001, pp 235-243

[HM01] Huszerl G. and Majzik I., "Modelling and analysis of redundancy management in distributed object–oriented systems by using UML statecharts", *In Proc. of the 27th EuroMicro Conference, Workshop on Software Process and Product Improvement*, Poland, 2001, pp 200-207.

[HOB95] Hobbs J. R., "Sketch of an ontology underlying the waywe talk about the world", *International Journal of Human-Computer Studies*, 1995, Volume 43, pp 819-830.

[HOO98] Hoog R, "Methodologies for Building Knowledge Based Systems: Achievements and Prospects", *Handbook of Expert Systems*, CRC press, 1998.

[HSW97] Heijst Van G., Schereiber A. T. and Wielinga B. J., "Using Explicit Ontology in KBS Development", *International Journal of Human and Computer Studies*, 1997, Volume 46, No. 2, pp 293-310.

[JAC92] Jacobson I., "Object Oriented Software Engineering- A Use case driven approach", *Pearson Education*, 1992.

[JCJ+07] Jacobson I., Christerson M., Jonsson P. and Overgaard G., "Object Oriented Software Engineering – A Usecase driven Approach", 2007.

[JIN00] Jin Z., "Ontology-Based Requirements Elicitation," *Journal of Computers,* 2000, Volume 23, No. 5, pp 486-492.

[JMF08] Jureta I. J., Mylopoulos J. and Faulkner S., "Revisiting the Core Ontology and Problem in Requirements Engineering", *16th IEEE International Requirements Engineering Conference*, 2008.

[JMY99] Jurisica I., Mylopoulos J. and Yu E., "Using Ontologies for Knowledge Management: An Information System Perspective", *In proceedings of ASIS'99*, 1999, pp 482-496.

[JUR01] Jurjens J., "Towards development of secure systems using UMLsec", Lecture Notes in Computer Science, 2001.

[KBG+06] Karyda M., Balopoulos L., Gymnopoulos S., Kokolakis C., Lambrinoudakis S., Gritzalis, and Dritsas S., "An Ontology for Secure E-Government Applications." *In Proceedings of the First International Conference on Availability, Reliability and Security, IEEE Computer Society*, 2006.

[KG02] Khayati O. and Giraudin J. P., "Components retrieval systems: Reuse in Object-Oriented Information Systems Design," *OOIS workshop Montpellier*, France, 2002.

[KL02] Khan L. and Luo F., "Ontology construction for Information Selection", *14th IEEE Conference on Tools with Artificial Intelligence*, 2002, pp 122-127.

[KM97] Krishnamurthy S. and Mathur A. P., "On the estimation of reliability of a software system using reliabilities of its components", *In Proceedings of the 8th International Symposium on Software Reliability Engineering (ISSRE '97)*, Washington, DC, USA,IEEE Computer Society, 1997.

[KMY91] Kelly J. P. J., McVittie T. I. and W. I. Yamamoto, "Implementing design diversity to achieve fault tolerance", IEEE Software,1991, Volume 8, No. 4, pp 61-71.

[KOG+08] Kossmann M., Odeh M., Gillies A. and Wong R., "From Process-Driven to Knowledge Driven Requirements Engineering Using Domain Ontology", *INCOSE*, 2008.

[KS06] Kaiya H. And Saeki M., "Using Domain Ontology as Domain Knowledge for Requirements Elicitation," *in Proceedings of 14th IEEE International Requirements Engineering Conference*, 2006, pp 186-195.

[LAB+96] Laresgoiti I., Anjewierden A., Bernaras A., Corera J., Schreiber A. T. and Wielinga B. J., "Ontologies as vehicles for reuse: A mini-experiment", *In B. R. Gaines and M. A. Musen (Eds.)", Proceedings of the 10th Banff KnowledgeAcquisition for Knowledge-Based SystemsWorkshop KAW'96,* Banff, Alberta, Canada, 1996.

[LEN95] Lenat D. B., "CYC: A large-scale investment in knowledge infrastructure", *ACM 38*, 1995, Volume 11, pp 33-38.

[LG90] Lenat D. B. and Guha R. V., "Building large knowledge-based systems. Representation and inference in the Cyc project", Reading, Massachusetts: Addison-Wesley, 1990.

[LGS+99] Lopez M. F., Gomez-Perez A., Sierra J. P. and Sierra A. P., "Building a chemical ontology using methontology and the ontology design environment", *IEEE Intelligent Systems*, 1999, Volume 14, pp 37–46.

[LHC+05] Lo J. H., Huang C.Y., Chen I. Y., Kuo S. Y. and Lyu M. R., "Reliability assessment and sensitivity analysis of software reliability growth modelling based on software module structure", *Journal of Systems Software*, 2005, Volume 76, No. 1, pp 3-13.

[LINK1] www.bettscomputers.com/fivetypesofinformationsystems.html?

[LINK2] www.idi.ntnu.no/grupper/su/courses/dif8901/presentations

[LINK3] http://www.cbjectiver.com.

[LINK4] http://www.kinetium.com/map/demo/demo_index.html.

[LINK5] www.lifecyclestep.com/open/406.0HomeProjSize.html

[LINK6] OMG, "Reusable Asset Specification", OMG Available Specification Version 2.2, formal 05-11- 02, www.omg.org.

[LINK7] http://msdn.microsoft.com/en-us/library/ee658094.aspx

[LINK8] http://www.microsoft.com/mscorp/execmail/2005/02-03interoperability.mspx

[LINK9] Cyc: OpenCYc.org "Formalized Common Knowledge" in http://www.opencyc.org68

[LIU92] Liu Z. Y., "Integrating two ontologies for electronics. In B. Faltings and P. Struss (Eds.)", *Recent Advances in Qualitative Physics*, Cambridge, Massachusetts: The MIT Press, 1992, pp 153-168.

[LKM+08] Letier E., Kramer J., Magee J. and Uchitel S., "Deriving Event-based Transition Systems from Goal-oriented Requirements Models", *Automated Software Engineering,* 2008, Volume 15, No. 2, pp 175-206.

[LPR93] Lubars M., Potts C. and Richter C, "A Review of the State of the Practice in Requirements Modelling", *Proceedings of the IEEE International Symposium on Requirements Engineering,* San Diego, USA IEEE Computer Society, 1993, pp 2-14.

[LV02] Letier, E. and van Lamsweerde A., "Deriving Operational Software Specifications from System Goals", *Proc. FSE'10:* 10th *ACM Symp. Foundations of Software Engineering*, Charleston, 2002.

[LV04] Letier, E. and van Lamsweerde A., "Reasoning about Partial Goal Satisfaction for Requirements and Design Engineering", *Proc. ACM FSE'04*, 2004, pp 53-62.

[LYU07] Lyu M. R., "Software reliability engineering: A roadmap", *In FOSE '07: 2007 Future of Software Engineering,* Washington, DC, USA, IEEE Computer Society, USA, 2007, pp 153–170.

[LYU96] Lyu M. R., "Handbook of Software Reliability Engineering", *IEEE Computer Society Press and McGraw-Hill*, 1996.

[MA05] Mendes O. and Abran A., "Issues in the Development of an Ontology for an emerging Engineering Discipline", *1st workshop on Ontology, Conceptualization and Epistemology for Software and System engineering*, Spain, 2005.

[MAR03] Marc D., "Towards Security Ontology." *IEEE Security and Privacy 1*, 2003, Volume 3, pp 6-7.

[MC06] McDowell L. and Cafarella M., "Ontology-driven information extraction with OntoSyphon", *In International Semantic Web Conference*, 2006.

[MER+03] Mili H., Ah-Ki E., Godin R. and Mcheick H., "An experiment in software component retrieval", *Journal of Information and Software Technology*, 2003, Volume 45, pp 633-649.

[MF03] Milan P. G. and Falbo R. A., "Building Ontologies in a domain Oriented Software Engineering Environment", *IX Argentine Congress on Computer Science,* La Plata, Argentina, 2003.

[MF11] Montesino R. and Fenz S., "Information security automation: how far can we go?", *Sixth International Conference on Availability, Reliability and Security (ARES),* Vienna, Austria, 2011.

[MGM03] Mouratidis H., Giorgini P., and Manson G., "An Ontology for Modelling Security: The Tropos Project", *in Proceedings of the KES 2003 Invited Session Ontology and Multi-Agent Systems Design (OMASD'03),*University of Oxford, United Kingdom, 2003.

[MHG+05] Mouratidis, Haralambos, Giorgini P. and Gordon Manson J., "When Security Meets Software Engineering: A Case of Modelling Secure Information Systems." *Information Systems 30*, 2005, Volume 8, pp 609-629.

[MIZ98] Mizoguchi R., "A step towards ontological engineering", *Proceedings of 12th National Conference on AI of JSAI,* 1998, pp 24-31.

[MJJ06] Svahnberg M., van Gurp J. and Bosch J. "A taxonomy of variability realization", 2006.

[MJJ06] Svahnberg M., van Gurp J. and Bosch J., "A taxonomy of variability realization techniques", *Software – Practice and Experience*, 2006, Volume 35, No. 8, pp 705-754.

[MMJ+04] Marjo K., Matti V., Jyrki K., Sari K. and Reijo S., "Implementing requirements engineering processes throughout organizations: success factors and challenges", 2004.

[MS00] Maedche A. and Stab S., "Semi-automatic Engineering of Ontologies from Text", *12th international Conference on Software Engineering and Knowledge Engineering*, 2000.

[MVI95] Mizoguchi R., Vanwelkenhuysen J. and Ikeda M., "Task ontology for reuse of problem solving knowledge", *In Building and Knowledge Sharing, 2nd International Conference on Very Large-Scale Knowledge Bases, Enschede, The Netherlands*, 1995, pp 46-59.

[NFF+91] Neches R., Fikes R. E., Finin T., Gruber T.R., Senator T. and Swartout W. R., "Enabling Ontologies for knowledge Sharing", *AI magazine* 1991, Volume 12, No. 3, pp 36-56.

[NHM00] Nour P., Houz H. and Maurer F., "Ontology Based Retrieval of Software Process Experiences", *ICSE workshop on Software Engineering*, 2000.

[NM00] Noy N. F. and McGuinness D. L., "Ontology Development 101: A Guide to Creating Your First Ontology", Stanford University, 2000.

[NM04] Noy N. F. and Musen M. A., "Ontology versioning in Ontology management Framework", *IEEE Intelligent Systems*, 2004, Volume 19, No. 4, pp 6-13.

[OVR+06] Oliveira K. M., Villela K., Rocha A. R. and Horta G., "Use of Ontologies in Software Development Environments", *In Ontologies for Software Engineering and Technology,* Springer-Verlag Volume 10, 2006.

[PM95] Parnas D. L. and Madey, J., "Functional Documents for Computer Systems", *Science of Computer Programming,* 1995, Volume 25, pp 41-61.

[PMM+07] Ponsard C., Massonet P. Molderez J. F., Rifaut A. and van Lamsweerde A. "Early Verification and Validation of Mission-Critical Systems", *Formal Methods in System Design* Springer, June 2007, Volume 30, No. 3, pp 233-247.

[PRE05] Pressman R. S., "Software Engineering: a practitioner's approach", *Mc Graw Hill*, New York, 2005.

[RAM05] Ramachandran, "Software Reuse Guidelines", *ACM SIGSOFT Software Engineering Notes*, 2005, Volume 30, No. 3, pp 1-8.

[REI97] Reifer D. J., *"Practical Software Reuse, Strategies for introducing reuse concepts in your organization"*, John Wiley & Sons Inc, 1997.

[RHT+01] Raskin V., Hempelmann C., Triezenberg K. and Nirenburg S., "Ontology in Information Security: A Useful Theoretical Foundation and Methodological Tool", *In Proceedings of the New Security Paradigms Workshop,* New York, USA, ACM, 2001.

[RK11] Radack S. and Kuhn R., "Managing Security: The Security Content Automation Protocol," *IT Professional*, 2011.

[RL02] Rus I. and Lindvall M., "Knowledge Management in Software Engineering", *IEEE Software*, 2002, Volume 19, No. 3, pp 26-38.

[SAB93] Sablayrolles P., "A two-level semantics for french expression of motion", *DFKI research report*, 1993.

[SBF98] Studer R., Benjamins V. R. and Fensel D., "Knowledge Engineering: Principles and Methods", *IEEE Transaction on Knowledge and Data engineering*, 1998, Volume 25, No.12, pp 161-197.

[SFO03] Sindre G., Firesmith D. G. and Opdahl A. L., "A Reuse-Based Approach to Determining Security Requirements", *In Proceedings of the 9th International Workshop on Requirements Engineering: Foundation for Software Quality (REFSQ'03)*, Klagenfurt/Velden, Austria, 2003.

[SG05] Seok Won Lee and Gandhi R.A., "Ontology-based Active Requirements Engineering Framework", *In the Proceedings of the 12th Asia-Pacific Software Engineering Conference (APSEC'05)*, 2005.

[SHA95] Shaw M., "Patterns for software architectures", *In: J.Coplien, D. Schmidt (eds), Pattern Languages of Program Design*. Addison-Wesley, Reading, MA, 1995, pp 453-462.

[SI10a] Sharma S. and Ingle M., "Developing Ontology for Information Systems Using Object Oriented Engineering Concepts", *National Conference on ICT: Theory, applications and practices*, Sir Padampat Singhania University, Udaipur, 2010.

[SI10b] Sharma S. and Ingle M., "Study of Object Oriented Software Engineering versus Ontology Engineering", *National Conference on Emerging Technologies in Electronics, Mechanical and Computer Engineering*, IIST, Indore, 2010.

[SI11a] Sharma S. and Ingle M., "An Ontology Driven Information System", *International Journal of Computer Technology and Applications*, 2011, Volume 2, No. 1, pp 147-154.

[SI11b] Sharma S. and Ingle M, "An Ontology Aided Requirement Engineering Framework", *International Journal of Advanced Research in Computer Science*,2011, Volume 2, No. 1, pp 279-283.

[SI11c] Sharma S. and Ingle M., "REP Models versus OntoAidedRE - A Parameters Based Study", *International Journal of Latest Trends in Computing,* 2011, Volume 2, No. 1, pp 172-177.

[SI11d] Sharma S. and Ingle M., "Developing a Reusable Framework using an Ingenious Approach", *International Journal of Research and Reviews in Computer Science (IJRRCS)* 2011, Volume 2, No. 4, pp 1082-1087.

[SI11e] Sharma S. and Ingle M., "An Ontology Based Reuse Algorithm towards Process Planning in Software Development", *International Journal of Advanced Computer Science and Applications,* 2011, Volume 2, No. 9, pp 133-137.

[SI12a] Sharma S. and Ingle M., "Assessment of Ontological Reuse versus Object Oriented Reuse Anchored in Various Reuse Subclasses", *International Journal of Engineering and Technology (IJET)*, 2012, Volume 2 No. 3, pp 469-474.

[SI12b] Sharma S. and Ingle M., "Ontology Based Specifications for Software Reliability Advancement", *International Journal of Computer Applications*, 2012, Volume 43, No.13, pp 18-26.

[SI12c] Sharma S. and Ingle M., "Object Oriented versus Ontology oriented Software Reliability Development", *International Conference on Software Engineering CONSEG'12*, DAVV, Indore, 2012.

[SK03] Stuckenschmidt H. and Klein M., "Integrity and change in modular ontologies", *In Proceedings of the International Joint Conference on Artificial Intelligence - IJCAI'03*, Acapulco, Mexico, 2003.

[SK04] Stuckenschmidt H. and Klein M., "Structure-based partitioning of large concept hierarchies", *In Proceedings of the 3rd International Semantic Web Conference, Hiroshima*, Japan, 2004.

[SMA05] Kaiya H. and Saeki M., "Ontology Based Requirements Analysis: Lightweight Semantic Processing Approach," *in Fifth International Conference on Quality Software*, 2005, pp 19-20.

[SM98] Sutcliffe A. and Maiden N., "The domain theory for requirements engineering," IEEE Transactions on Software Engineering, 1998, Volume 24, No. 3, pp 174-196.

[SMB07] Shoval P., Maidel V. and Bracha S., "An ontology content-based filtering method", *I.Tech-07– Information Research and Applications,* 2007.

[SMJ02] Spyns P., MeersMan R. and Jarrar M., "Data Modelling Versus Ontology Engineering", *SIGMOD* 2002, Volume 31, No. 4, pp 12-17.

[SOM01] Sommerville I., "Software Engineering", *Addison-Wesley, sixth edition*, 2001.

[SPL06] Shenhar A., Poli M. and Lechler T., "A New Framework for Strategic Project Management", 2006.

[SRK+97] Swartout B., Ramesh P., Knight K. and Russ T., "Towards Distributed Use of Large-Scale Ontologies," *Spring Symposium on Ontological Engineering of AAAI*, 1997, pp 138-148.

[SS03] Sugumaran V. and Storey V., "A semantic-Based Approach to Component Retrieval", *The Data Base for advances in Information Systems*, 2003, Volume 34, No. 3.

[SS06] Schlicht A., Stuckenschmidt H., "Towards Structural Criteria for Ontology Modularization", *In Proceeding of the ISWC 2006 Workshop on Modular Ontologies*, 2006.

[SS99] Sodhi J. and Sodhi P., "*Software Reuse, Domain Analysis and Design Process*", Computing McGraw Hill, 1999.

[SSS+01] Staab S., Schnurr H. P., Studer R., Sure Y., "Knowledge Processes and Ontologies", *IEEE Intelligent Systemds*, 2001, Volume 16, No. 1, pp 26-34.

[ST99] Swartout W. and Tate A., "Ontologies, Guest Editors" Introduction", *IEEE Intelligent Systems,* Special Issue on Ontologies, 1999, Volume 14, No. 1, pp 18-19.

[STA08] Stacy Goff, "Measuring and Managing Project Quality", 2008.

[SZY98] Szyperski C., "Component Software: Beyond Object-Oriented Programming*", ACM Press/Addison-Wesley, New York, NY/Reading, MA*, 1998.

[TA00] Tautz C. and Althoff H., "A Case Study on Engineering Ontologies and Related processes for Sharing Software Engineering process", *In proceeding of 12th International conference on Software Engineering and Knowledge engineering,* 2000, pp 318-327.

[TG06] Tsoumas B. and Gritzalis D., "Towards an Ontology-Based Security Management", *In Proceedings of, 20th International Conference on Advanced Information Networking and Applications*, 2006.

[TRA95] Trammell C., "Quantifying the reliability of software: statistical testing based on a usage model", *In ISESS '95: Proceedings of the 2nd IEEE Software Engineering Standards Symposium*, Washington, DC, USA, 1995.

[TVM+04] Tran Van H., van Lamsweerde A., Massonet P. and Ponsard Ch., "Goal-Oriented Requirements Animation", *Proc. RE'04, 12th IEEE Joint International Requirements Engineering Conference*, Kyoto, 2004, pp 218-222.

[UG96] Uschold M. and Gurninger M., "Ontologies: Principles, Methods and Applications", *Knowledge Engineering Review*, 1996, Volume 11, No. 2, pp 93-155.

[UJ99] Uschold M. and Jasper R., "A Framework for Understanding and Classifying Ontology Applications", *In Proceedings of IJCAI workshop on Ontologies and Problem Solving Methods*, 1999.

[VAN01] Van Lamsweerde A., "Goal-Oriented Requirements Engineering: A Guided Tour", *Invited Minitutorial, Proc.RE'01 - 5th Intl. Symp. Requirements Engineering*, Toronto, 2001, pp 249-263.

[VAN03] Van Lamsweerde A., "From System Goals to Software Architecture", *In Formal Methods for Software Architecture*, LNCS 2804, Springer-Verlag, 2003.

[VAN04a] Van Lamsweerde A.,"Elaborating Security Requirements by Construction of Intentional Anti-Models", *Proc. ICSE'04, 26th International Conference on Software Engineering*, Edinburgh, ACM-IEEE, 2004, pp 148-157.

[VAN04b] Van Lamsweerde A., "Goal-Oriented Requirements Engineering: A Roundtrip from Research to Practice", Keynote paper, *Proc. RE'04, 12th IEEE Joint Intl. Requirements Engineering Conf.*, Kyoto, 2004, pp 4-8.

[VAN08] Van Lamsweerde A., "Requirements Engineering: From System Goals to UML Models to Software Specifications", Wiley, 2008.

[VDL98] Van Lamsweerde A., Darimont R. and Letier E., "Managing Conflicts in Goal-Driven Requirements Engineering", *IEEE Trans. on Software. Engineering*, November 1998, Volume 24, No. 11, pp 908-926.

[VHB08] Vorobiev A., Han Jun N. and Bekmamedova, "An Ontology Framework for Managing Security Attacks and Defenses in Component Based Software Systems", *19th Australian Conference ASWE*, 2008.

[VL00] Van Lamsweerde A. and Letier E., "Handling Obstacles in Goal-Oriented Requirements Engineering", *IEEE Trans. On Software Engineering*, Special Issue on Exception Handling, 2000, Volume 26, No. 10, pp 978-1005.

[VRS99] Valente A., Russ T., MacGregor R. and Swartout W., "Building and (re)using an ontology of air campaign planning", *IEEE Intelligent Systems 14*, 1999, pp 27–36.

[VSM95] Van Lamsweerde A., Darimont R. and Massonet Ph., "Goal-Directed Elaboration of Requirements for a Meeting Scheduler: Problems and Lessons Learnt", *Proc. RE'95 - 2nd Intl. IEEE Symp. on Requirements Engineering*, 1995, pp 194-203.

[VSS+05] Villela K., Santos G., Schnaider L., Rocha A. R. and Travassos G. H., "The Use of Ontologies to Support Knowledge Management in Software Development Environments", *Journal of Brazilian Computer Science*, 2005, Volume 11, No. 2, pp 45-60.

[WAR09] Waralak V. Siricharoen, "Merging Ontologies for Object Oriented Software Engineering", 2009.

[WER97] Werner C., "A Reuse Based O. O. Software Development Environment", *In Proceedings of Tools*, Beijing, China, 1997.

[WZX06] Wang Z., Zhan D. and Xu X., "A Component Retrieval Method based on Feature Tree Matching", *International Journal of Information Technology*, 2006, Volume 12, No. 8.

[YKD+06] Yutao M., Keqing H., Dehui D., Jing L. and Yulan Yan, "A Complexity Metrics Set for Large-Scale Object-Oriented Software Systems", *Proceedings of The Sixth IEEE International Conference on Computer and Information Technology* (CIT'2006), 2006, pp 188-189.

[ZZY+07] Zong-yong L. I., Zhi-xue W., Ying-ying Y. and Ying W., "Towards Multiple Ontology Framework for Requirements Elicitation and Reuse", *In proc. of COMPSAC*, 2007.