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. *OntoReuseAigo* 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.