Ontology-Based Requirement Management Approach for Distributed Software Environment

Submitted by

Sonia Kaleem

Registration No: 325-FBAS/MSSE/F10

Supervisor

Mr. Yasir Hafeez

UIIT Rawalpindi

Co-Supervisor

Mr. Asim Munir

Department of Computer Science and Software Engineering

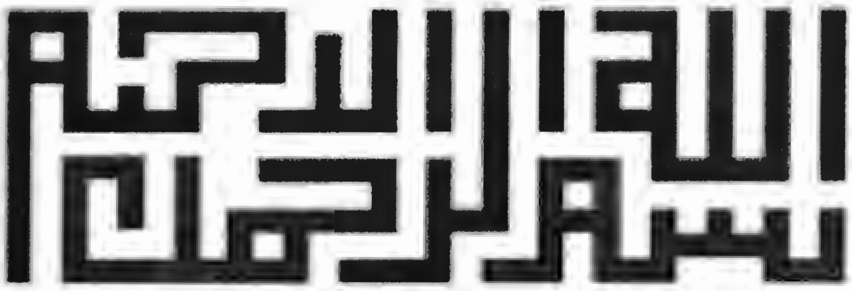
Faculty of Basic & Applied Sciences

International Islamic University, Islamabad.

2016

MS
00633
S00

1. Ontologies (Information retrieval)
2. Intelligent agents (computer software)



In the Name of Almighty Allah, The Most Beneficent, The Most Merciful.

International Islamic University, Islamabad

Faculty of Basic & Applied Sciences

Department of Computer Science & Software Engineering

Dated: 7th Sep, 2016

Final Approval

It is certified that we have read the thesis, entitled **"ONTOLOGY-BASED REQUIREMENT MANAGEMENT APPROACH FOR DISTRIBUTED SOFTWARE ENVIRONMENT"**, submitted by **Ms. Sonia Kaleem Reg No. 325-FBAS/MSSE/F10**. It is our judgement that this thesis is of sufficient standard to warrant its acceptance by the International Islamic University, Islamabad for MS degree in Software Engineering.

Project Evaluation Committee

External Examiner:

Dr. Aamer Nadeem, [illegible]

Associate Professor,Head of Department Bioinformatics,

Capital University of Science & Technology, Islamabad.

Internal Examiner:

Mr. Imran Saeed, [illegible]

Assistant Professor, Department of Computer Science & Software Engineering,

Faculty of Basic & Applied Sciences,

International Islamic University, Islamabad.

Supervisor:

Mr. Yasir Hafeez, [illegible]

Assistant Professor, University Institute of Information Technology,

PMAS-Arid Agriculture University, Rawalpindi.

Co-Supervisor:

Mr. Asim Munir, [illegible]

Assistant Professor, Department of Computer Science & Software Engineering,

Faculty of Basic & Applied Sciences,

International Islamic University, Islamabad.

A Thesis submitted to Department of Computer Science and Software Engineering

International Islamic University, Islamabad.

As a partial fulfilment for the award of the Degree of MSSE

ABSTRACT

Requirements Engineering (RE) is the primitive and most critical unit of the entire software development process that determines the success rate of a software system. It is very well known and well proven by the researchers that requirements are more important than any other component of the system because requirements establish a founding block for any future software project. Requirements Management is a challenging activity in collocated setting but it becomes more complicated and demanding in the environment of Global Software Development (GSD). To deal with the requirement's change, evaluating impact of change on the overall system, then communicating change with the distant GSD group, is such a chaotic mission to the point that it looks, since it produces misunderstandings and ambiguities among the GSD team. Aforementioned issues can be resolved by employing knowledge management in GSD setting that can be achieved through ontology. In this study, an ontology-based requirements management approach for Global Software Development (GSD) domain is presented, for which we have built up the ontology in four phases: Specification, Conceptualization, Formalization and Implementation with a case study and ultimately domain specialists have done the evaluation as well. The evaluation results and feedback from domain experts is an evidence that our presented solution is a better way to deal with the activity of requirements management in Global Software Development (GSD) set-up as it beefs up the communication structure that is an essence of Global Software Development (GSD) projects. This approach will be useful in many GSD organizations for requirements management and knowledge management where the distance factor creates more obstruction among Global Software Development (GSD) stakeholders.

Dedication

This Thesis is Dedicated to both my Beloved Parents for their Prayers, love and support throughout my life.

Acknowledgement

ALL Praises to Almighty Allah, the supreme, the most compassionate and His Prophet Muhammad (P.B.U.H). The Most perfect among all human beings ever born on the surface of earth, who is forever a source of guidance and knowledge for humanity as a whole.

First and Foremost, I would like to express my gratitude to the Chairperson of the department for her moral support and cooperation. I also want to say thanks for Madam Salma Imtiaz & Madam Zakia Jalil for their support and guidance during my research work.

I owe my gratitude to my supervisor Mr. Yasir Hafeez, who kept me motivated during the whole work of thesis. His useful suggestions, advice and ideas were helpful in keeping the work on track.

I am also thankful to my Co-Supervisor Mr. Asim Munir, who supported me every time and gave me useful feedback whenever I needed. Under his guidance I successfully overcame many difficulties.

I am grateful to all the participants of a software house in Islamabad and especially to my friend Asma for her co-operation by sparing her precious time and giving useful feedback during my thesis work.

This journey would not have been possible without the kind support and encouragement of my family, I am especially grateful to my parents, who supported me emotionally and financially. I must acknowledge my husband Sheraz Khurram, my brothers Hamad Kaleem and Hamid Kaleem and my sister Sidra Kaleem, for their love, encouragement and kind concern throughout my thesis work.

Sonia Kaleem

Declaration

I hereby declare that the research presented in this thesis is my own work, excluding where otherwise acknowledged and that the thesis is my own composition. No part of the thesis has been previously presented for any other degree.

Date: - 07-09-2016 - -

Full Name: - - Sonia Kaleem.

Table of Contents

List of Tables

List of Figures

Key Terms Used

RE:	Requirements Engineering
SDLC:	Software Development Life Cycle
SDP:	Software Development Process
GSD:	Global Software Development
SRS:	Software Requirements Specification
RCM:	Requirement Change Management
RT:	Requirement Traceability
RTM:	Requirement Traceability Matrix
DSE:	Distributed Software Environment
CRF:	Change Request Form
PM:	Project Manager
QA:	Quality Assurance

Chapter 1

INTRODUCTION

1.1. Background

Software organizations understand and value the influential factors during developing a software that leads towards the rewarding system. A successful software development process relies upon the quality of requirements engineering (RE) process though RE phase is considered to be an integral part of the entire software development process [1]. Requirements Engineering (RE) can be divided into two major categories, Requirements Development and Requirements Management.

The activity of Requirements Development covers all of those tasks that are concerned with eliciting, collecting, understanding, analyzing and validating requirements for the software system and Requirements Management deals with the requirement change and requirement traceability [2] this whole process is called Requirements Engineering (RE) because a systematic and repeatable process is followed in arriving on a set of requirements that are considered complete and consistent.

Requirements Engineering ensures that all stakeholders have agreed on a set of requirements and have a mutual comprehension about the requirement engineering process as well [3]. Incomplete requirements, unsatisfactory comprehension and imperfect management of requirements are the real reasons of software system's failure.

At the present time, Trends have changed in the world of software development as software companies are moving towards the "Globalization". The speed of global software development is getting faster due to the utmost advantages that global software development offers the stakeholders, and companies distribute their project's work all over the world for having business benefits.

Global Software Development became more prevailing trend because of minimal development cost, skilled professionals, nearness to the market and round the clock development. Apart from that, GSD Projects are performed by distributed stakeholders so GSD teams face issues of communication, coordination and control [4]. In the global software development, disseminated stakeholders from various geographic sites work on a shared project beyond the barriers such as culture, language and time variations and these factors provoked by distance and also give rise to GSD challenges [5].

Hardly any other business is promoting globalization as much as the software and IT industry remains does. It continues the drift to move software development preferably in those countries that have cheap wages; For instance, In India one-hour programming only cost 20-25 percent of one hour in Germany. And it drives the companies all over the world to distribute their work in India, China, etc. [6].

Due to the rapid expansion of distributed projects across the globe, it becomes indispensable to understand the nature of these kinds of software development environment. Distance is the key issue in globally distributed projects that aggravates other challenges such as communication, coordination and control [5] therefore GSD projects face more trouble with requirements change and requirements change management as compared to single site development. Requirements are very important because requirements are the bridge between a real world problem that has to be addressed and software system.

In fact, the Requirements reflect the problems in the real world, but in a way that can be utilized by the software engineers. Requirements have the embedded element of evolution that keeps them alive and more beneficial, although It is very hard to capture all the requirements at the initial stage because stakeholders don't know what they want from a new system. The environment in which business operates evolves at a high speed [7] as a result their requirements for software system are changing, therefore the change becomes unavoidable because human beings learn with the passage of time and the change incorporation becomes imperative for the system.

To accommodate change is not really simple task even in a single site development environment, but it get worse in cross site projects due to difficulty in communication and requirement change management is a communication and collaborative-intensive activity that needs strong communication mechanism, furthermore GSD demands [8] continuous communication and quick response for managing requirement's change that can be achieved through solid communication structure because in-time and frequent communication is a key ingredient of effective requirements change management (RCM) process.

When the change request is submitted from the user's end, at that time software experts start to observe the change impact on the previous set of requirements, therefore there is a need to have complete information about the domain and all the related requirements in the developer's mind. As the time passes, the requirements become more mature and domain knowledge increases in size, thus it is difficult to keep the required information in the brains and archives [7]. Many software companies fail to retain and utilizing the domain knowledge, user's requirements and previous experiences, if organizations keep the important asset of previous knowledge while developing a system, then the progress in development can be swift and more productive.

Improper management of the requirement's knowledge is the flagrant weakness of the existing RE methods. Ontologies are good candidates for handling these deficiencies and concerns of requirements engineering because they represent the knowledge of a particular domain and the relationships between concepts in a more formal manner [9]. Strong requirements management ensures the success of a software project because it not only manages the change in requirements but also assures the relationship among requirements.

Most certainly, Change impact analysis is an effective activity for figuring out the effects of change functionality of the software and different artifacts of the system [10] and traceability information is very important in impact analysis while managing requirement change.

Decisions about cost, impact and schedule are based on traceability information that produces relationships between requirements and other artifacts. Our solution has been proposed for GSD setting, which will helpful in managing requirement change along impact analysis and also in visualizing change impact on the system.

1.2. Problem Description

Requirements Engineering (RE) is the basic and most significant unit of the entire software development process. In this phase, requirements are collected from diversified customers and users as the requirements are the foundation block of the future software system. It is studied that more than 50% of project's requirements will be changed before its deployment in the real environment [11].

The role of requirements engineering can't be denied at any stage of the software development life cycle; in fact, all the software projects strongly rely on requirements engineering. In a survey of twelve companies [12] it was found that 48 % of software problems were due to the problems in requirements.

Inadequate, imperfect and wrong perceived requirements [1] are the major reasons of software disaster. However, change is accepted as an important element of software because it keeps alive the system to be built although change management is a difficult job even in single site environment and gets more challenging in multi-site due to less communication percentage among distant team members [5]. Due to improper management of the change GSD team can deliver a low quality software system that can deteriorate business objectives as well [15].

Managing software requirement's change along impact analysis in GSD settings is still almost a nightmare due to the following reasons. Improper communication structure among GSD sites, knowledge management and knowledge sharing, high level of misunderstandings and confusions among GSD teams and conflict management. All these issues are a consequence of distance because distance is the key issue therefore the activity of requirement management suffers a lot in a distributed environment.

A large number of these issues can be solved by using knowledge management in the GSD set-up that can be achieved through software engineering ontology, because ontology contain an agreed-upon vocabulary that is required for proper knowledge management and minimize the ambiguities and misunderstandings among the distant team members. Moreover, the process must facilitate in assessing and viewing the impact of requirement's change that can be done with the help of ontology.

In the past, many frameworks have been introduced to manage the software requirement's change in GSD settings, however requirements traceability (RT) has been used less with requirements change management (RCM) process in order to analyze the impact of change for the future implication of software practices.

If both the requirements change management process and requirements traceability are added through some mechanism, we can manage requirement change efficiently and timely by keeping in view the impact of requirements and here is no method to glue them together as a whole. Researchers emphasized that robust models and efficient processes are a primary necessity to perform GSD business [8]. Keeping in mind the discussed challenges, a requirements management model is suggested in a globally distributed development environment that is a humble addition in this research field.

1.3. Objective of Study

Research Goal: The key objective of the dissertation was to produce the requirements management model to strengthen the requirement management process with a case study in the GSD environment.

We have carried through the project by developing two ontologies; one for Requirement Change Management in global software development (GSD) and another for Requirement Traceability Matrix (RTM), in order to obtain the desired result, the ontologies are merged into a single ontology. Our proposed approach has been assessed by a real case study that affirms the effectiveness of the model. Following research questions are investigated:

RQ1: What global software development (GSD) challenges affect the activity of requirements management?

RQ2: How ontology can be used in requirements management in global software development (GSD) environment?

RQ3: What challenges could be resolved using the provided solution?

1.4. Research Methodology

This part discusses the research approach that is performed to carry out our research work. Many research methodologies exist in a research setting, but we have chosen a qualitative research method for this project.

1.4.1 Qualitative Research

This research adopted a qualitative methodology, that concentrates the words rather than numbers [57]. we liked to study individuals, process and the process related technology in real life setting therefore qualitative approach was an appropriate choice for this purpose.

Acquiring qualitative information from the individuals in real-life set-up takes the adoption of qualitative methodology which is reasonable for such data gathering, thus this method was picked as our selected approach of investigation [56].

Furthermore, researcher endorsed that qualitative approach is the best option for getting strong comprehension of information gathered from individuals about their encounters alongside their native setting [57].

1.4.2 Case Study

The research methodology selected to implement the requirements management approach in GSD settings is an experimental case study. A Case study research is a pragmatic analysis that investigates a current phenomenon (the case) in a real life setting and its emphasis on to answer the "how" and "why" questions [58].

It is more suitable when studied case is not theorized properly or plenty of actors, procedures and goals have to deal and there is no restriction among phenomenon and real life set up.

We have used a qualitative approach to assess the requirements management issues in the global software development environment. We have picked a real life case for investigation of a software house located in Pakistan that also has a branch in the united states (U.S).

1.5. Contribution

The primary contribution has been prepared to serve the research questions mentioned in Section 1.3. The aim of this research is to investigate the process of requirement change management and requirement traceability in GSD setting, that is designed with the help of ontology for the storage of knowledge of users' requirements and the domain knowledge of its concern. Main contributions are:

1) All the stakeholders come to a consensus easily as all of them see the views with clear vision and understanding and they don't have to make certain decisions on estimation.
2) Stakeholders and specially users can get clear pictures of what kind of product these requirements will take shape, then they can timely decide if any change is to be made.
3) As an ontology is a uniform standard, therefore every project manager can use/reuse and understand it due to its uniformity, no ambiguity remains in anyone's mind.

4) Since all the areas and artifacts are in the view of software engineers, they can do a better analysis of change impact on the rest of the requirements, design and the product.

1.6. Thesis Structure

The structure of the study is illustrated in the figure below.

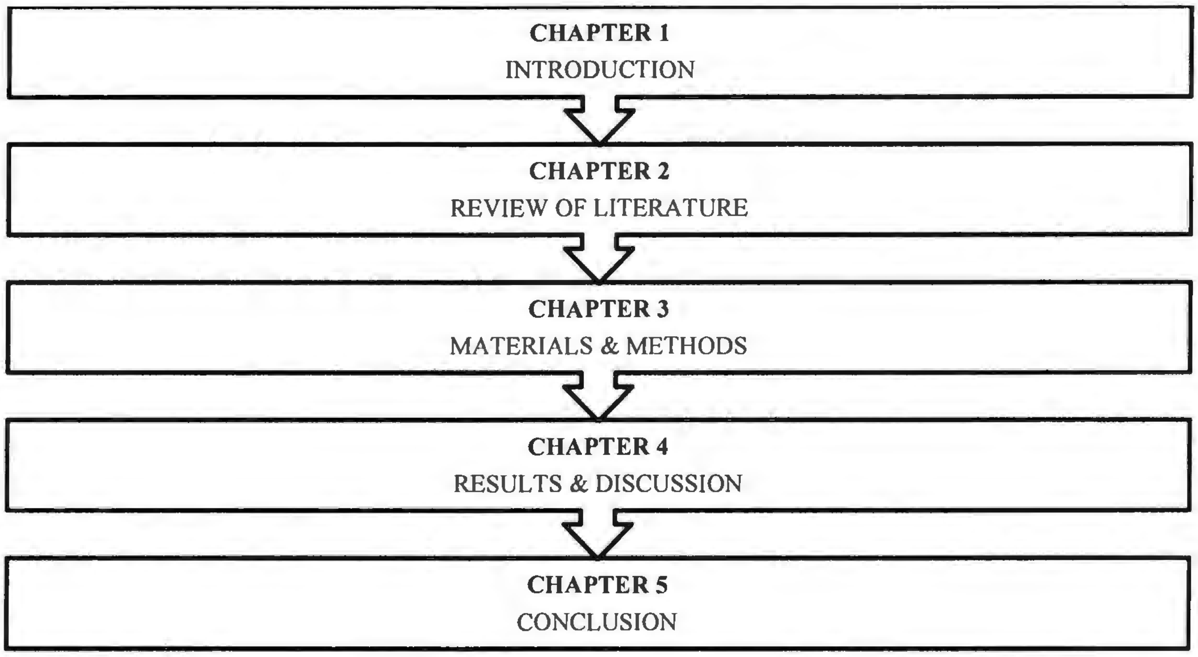


Figure 1: Thesis Structure

Chapter 2

REVIEW OF LITERATURE

2.1. CHAPTER OVERVIEW

In this section the overview of the literature related to requirements engineering, distributed software environment and ontology is presented. In the light of literature, it was studied that procedure for requirements change management must be efficient and well timed for attaining software project's success.

2.2. Requirements Engineering & Requirement Change Management

The following section describes the concept of requirement engineering and informs how to integrate changes.

Keeping in mind the divergent stages of the software development life cycle, it is realized that Requirements engineering (RE) is one of the key phases and fragile as well. The rest of the software development phases depend on this particular phase. Requirements engineering is performed by a procedure which has distinctive stages with various activities to perform. It is apparent from research that the software's quality is equally proportional to the caliber of its process and requirements engineering [35,36,37]. Inappropriate procedure for requirement change management leads towards software system failure.

If errors or changes are not handled in timely then they can cost up to 200 times at later phases of software development. In a study [38] it is revealed that if a change is managed or an error is corrected at requirements definition phase and it costs 1 US$, the same activity will cost 5 US$ at the design phase and may cost up to 200 US$ after the system is delivered. Different studies indicate that most of the softwares fail to finish in time, within budget and with real planned features. Inadequate requirements [1] are the major reason of software failure. In a survey of twelve companies [12] it was found that 48 % of software problems were due to the problems in requirements.

The primary concern of requirements engineering is to identify the goals of the organization to build and It is the part of RE to convert these goals into services and constraints.

In this chapter Requirements engineering and its background are explained accompanying requirements engineering activities.

2.3. What is a Requirement

Requirements explain what we will have after a software is built. According to [39] Definition of requirement is:

I. A condition or ability that is used for solving a user's problem for accomplishing the project objectives.

II. A condition or ability that is required to fulfill by a system or system component to satisfy an agreement, documentation or other formal reports.

III. A standard representation of a condition or ability that are mentioned in (I) and (II).

Requirements are specifications to be implemented [2] and they also describe the behavior, property or attribute and in some cases constraints of a system. These definitions indicate that requirements are not only the needs of a user, but these also include the ones that originate in an organization and also based on the government's and industrial laws and measures. Thus we can say that requirements are the needs of all the stakeholders and these tell 'what' a system should do. It does not tell 'how' to achieve [1].

Requirements can be classified as [2] Functional Requirements: something that the software system is expected to accomplish, Non-functional Requirements: something that the software system must have e.g. accuracy, reliability, security, scalability, usability etc. Goal Level Requirements: these requirements link to business objectives, Domain Level Requirements: requirements of that particular domain in which the problem lies and a product to be developed, Product Level Requirements: requirements of the product itself, Design Level Requirements: these requirements are about of what to fabricate, key requirements and derived requirements etc.

2.4. Requirements Engineering Process

The journey of software development life cycle (SDLC) begins from Requirement engineering (RE) activities that linked to the software requirements. Requirement Elicitation, Requirement Analysis, Requirement Specification, Requirement Validation & Verification and Requirement Management are key activities in RE process. These activities can be performed by following different approaches. In [49] a theoretical model is offered that shows iterations between different activities, but it's a form of linear model while [40] has proposed a linear model without overlapping or iteration of activities and the Spiral model [42] developed in iterations in each cycle. It was analyzed in [41] that in current practice, normally a linear model with iterations, is adopted.

2.5. Requirements Engineering Activities

As cited earlier, RE process is framed up of two primary activities. Requirement Development and Requirement Management. The first activity includes elicitation, analysis, specification, validation and verification of requirements. The latter one includes requirement change management and requirements traceability.

Requirements Engineering establishes the solid foundation for the future development of a software system and guarantees that all stakeholders have achieved consensus on a set of requirements. Thus, the proper requirement engineering process is necessary in the journey of software development. Hence, poor management and insufficient understanding of requirements are the major reasons of software project's failure [3].

The requirements engineering process consists on the following activities [2]..

2.5.1 Requirements Elicitation:

This activity is the first part of requirements engineering (RE) in which requirement analyst/engineer starts to interact with the customers in identifying their needs, services and constraints that are required to form a new system and domain knowledge where the software system has to be used. Elicitation is made by performing interviews, questionnaires, brainstorming session, document analysis and prototyping, etc. The main purpose of this activity is to achieve consensus among stakeholders on a specific set of requirements.

2.5.2 Requirements Analysis and Negotiation:

In this activity, all the disputes and problems related to requirements are rectified. Detailed analysis is performed to make sure about the impact of change on the project, cost, schedule and customers' satisfaction. Requirements are declared complete, consistent and correct in this activity.

2.5.3 Requirements Specification:

In this activity, the requirements get the official and more formal place in a software artifact that is known as software requirements specification (SRS) that describes the purpose of software and how it will be expected to perform.

2.5.4 Requirements Validation:

This activity is concerned with checking that the system satisfies the client's actual needs. It additionally certifies the completeness, correctness and consistency of requirements and is useful as a real world solution.

2.5.5 Requirements Management:

Requirements management is a critical activity for system development but this activity guarantees that the software organization validates and conforms to the customer's needs. It manages the change in requirements and establishes the interdependencies between requirements. Change management, Traceability, and Version control have a place in this activity.

2.6. LEVELS OF REQUIREMENTS

Understanding the requirements for the purpose of analysis and management is a heavy and complex task. Requirements analysis and management need to distinguish among different levels of requirements for better comprehension so that a project is finished on time and budget. RE helps in realizing this job easier through structured and controlled specification of requirements. Prerequisites can be classed in three levels [43] and each classification is divided along a three floor model for making managerial decisions. Requirements are classified along the layers of organization, product and project. For the purpose of decision making, each level is divided into strategic, tactical and operational management.

- Requirements at Organizational level: Requirements are evaluated against the goals and objectives of the organization.
- Requirements at Product level: Customers' requirements and concerns are balanced with the organizational concerns, goals and strategies.

- Requirements at Project level: Definition of requirements under which a project will be run keeping in view the project planning, scheduling, budget and risk management etc.

2.7. REQUIREMENTS CHANGE MANAGEMENT

Change is only constant and unavoidable; humans learn the things with the passage of time therefore they feel the need of change for getting perfect outcome. In software development, change management is an important activity and tricky as well. And handling change is a hard core job of assuring project's success, therefore change should manage in time, the more it gets later, more difficult is to manage [44]. Software engineering [47] is an activity of managing change in a business setting. Researchers [23] state that changes should be taken up to a certain point, and this level must be stronger for having a useful product.

The following is different cases of change [45]

i. **Adaptive Changes** are those that cope up with environmental change or application change without affecting software functionality.

ii. **Perfective Changes** are those that increase the system's performance by implementing new or modified requirements.

iii. **Corrective Changes** are those that relate to fixing bugs in the software system.

iv. **Preventive Changes** are those proactive adjustments that are done after expected changes that may come along later.

Change impact analysis is the significant activity for identifying the effects of change functionality of the software and different artifacts of the system [10] and this assessment finalize about the acceptance and rejection of change. Whether the change is positive or negative [44] it will have an effect on the system's functionality, its quality, marketing, user's expectations, time line and so on.

The processes of change management are performed by holding a conventional process. Almost all the change management process models incorporate nearly all of primary functions taken during the procedure of handling change.

Following figure shows requirement change management activities [46]:

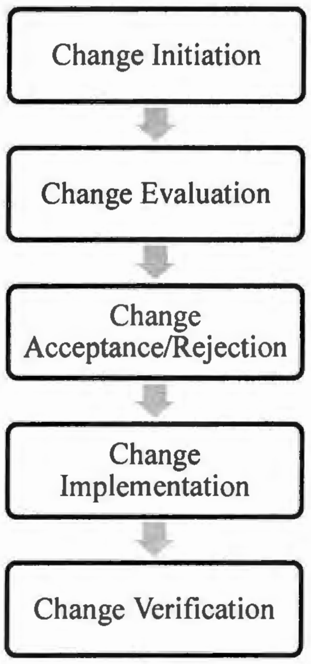


Figure 2: Change Management Activities

Requests for change is initially needed to evaluate for determining the impact of change, afterwards the decision is made to accept or reject the requested change. It is very important to encounter the effect of change on different artifacts of the software system. If the modifications are workable, then these are carried out.

2.8. CHANGE MANAGEMENT PROCESS MODELS

Literature has plentiful process models for managing requirement's change. Different models have their own properties, but all of them serve the same purpose. CHAM, Olsen, V-like, Need-based Requirements Management (NRM) are process models for change management that exist in the literature, some of them are reported briefly.

2.8.1 CHAM

CHAM process [48] is a good example that carries the related change record that is useful for recommending changes in other tasks. Writer emphasized on these change management tasks: change prevention, change prediction, impact analysis, change planning, stability assurance and help change documentation. The CHAM Model contains the 4 units: traceability analysis, change capture, reuse analysis and change management.

2.8.2 OLSEN'S MODEL

The author [47] has used the approach named as *rush hour* for managing change. The change management actions are treated as traffic jam and drivers with COP has to take on the situation. There really is no silver bullet. Each activity of software engineering is alike to change management. This model is selected as the abstraction software engineering process.

2.8.3 V-Like MODEL

Another model [49] V-like model is proposed for change management. The model has a feature of tailoring therefore it can be change according to the system's demands. The models additionally utilized as a support model for change management after the final deliverable as software maintenance activities [35] can be exercised in change management if all activities are part of that process model. V-like model comprises on change identification, understanding, solution analysis and specification, implementation and validation.

2.9. REQUIREMENTS TRACEABILITY

Gotel & Finkelstein defined [26] Requirements Traceability as "Requirements traceability refers to the ability to describe and follow the life of a requirement, in both forwards and backwards.". Pinheiro & Goguen defined RT as "Requirements traceability refers to the ability to define, capture and follow the traces left by requirements on other elements of the software development environment and the trace left by those elements on the requirements." [27].

CMMI explained RT "A discernable association between requirements and related requirements, implementations, and verifications." & Bi-directional Traceability as "An association between two or more logical entities that is discernable in either direction (i.e., to and from entity)." [28].

Study of requirements traceability is the significant activity in software development. Traceability information is most imperative in impact analysis while managing change. Without the knowledge of the relationship between requirements and other artifacts, decisions about cost, impact and schedule are not possible. After the implementation of change, traceability gives the assurance of accomplishment. In [50] authors say that neglecting traceability leads to low quality software of high cost.

2.9.1 TRACEABILITY TYPES

Generally, there are backward and forward traceability e.g. when requirements relate to its domain, it is backward and when related to design it is forward. Similarly, when domain relates to requirements it is forward and when design relates to requirements it is backward. There are two types of traceability: Pre-traceability and Post-traceability.

2.9.2 PRE- TRACEABILITY

It is the status of requirements when they are not yet specified. During this life of requirements, understanding is developed and elicitation is made. During this phase requirement are related to their origin, e.g. stakeholders, standards and rules, etc. In short, pre-traceability concerns with pre-production.

2.9.3 POST- TRACEABILITY

It starts when requirements are specified and they used in design and implementation. It is assured that the implementation is according to the requirements. Specified requirements are also related to test cases. Post-traceability concerns with requirements deployment.

Wieringa considered traceability in two types [29]

- Forward Traceability: A requirement is traced to constituents of a design or implementation.
- Backward Traceability: A requirement is traced to its source, that can be a person, institution, law, and so on.

Lindval, Sandahl split requirement traceability in the two groups [30]

- Horizontal Traceability is tracing the relevant items among various models.
- Vertical Traceability is tracing dependent items within a model.

2.10. DISTRIBUTED SOFTWARE DEVELOPMENT

The term “Distributed Software Development” has also been used [31] describing the act of developing software in geographically distributed settings, but not necessarily on a global scale. Another suitable term which carries with it less ambiguity is “Globally Distributed Software Development (GDSD)” [32] [33]. Distributed Software Development DSD, OR Global

Software Development GSD OR Global Software Engineering GSE OR Distributed Software Engineering DSE OR Multi-Site Software Development, Offshore Software Development; All these terms serve the same purpose of software development among globally dispersed locations [55].

Distributed software projects normally have two or more teams from various locations that got connected to complete the project's objectives. When teams are globally distributed [34] they face temporal, cultural and language barriers that give rise to communication issues while GSD setting gives the opportunity of availing its attractive benefits. Cost effectiveness is the momentous benefit to outsource the projects because software companies outsource their projects to those places which cost relatively low. [14]

Some GSD advantages and issues that faced by GSD stakeholders are listed below.

2.11. BENEFITS OF GSD

The popular researchers of GSD domain such as Conchuir [6], Damian & Zowghi [60], Carmel [32], Herbsleb & Moitra [4] etc. have discussed the noteworthy advantages of GSD settings. These are obviously great advantages and most frequently cited, e.g. Reduced cost of development, Follow the sun development, Access to talented workers, Nearness to market, etc. As the inclination to GSD development is because of its recognized and documented benefits.

2.12. CHALLENGES IN GSD

Regardless of the benefits that GSD offers to the distributed teams, it faces certain issues that are spurred by distance. Distance has considered the key issue in GSD settings because it provokes many other challenges that cause hindrance in the course of software development at multiple locations. The GSD factors such as time variation, geographical distribution, cultural and language barriers are dependent on the distance which produces the issues in communication, coordination and control that become complex and chaotic during the software development process. Moreover, insufficient communication, distant team members, different kind of processes, different kinds of development & testing softwares, different knowledge and expertise levels intensify the issues faced by GSD teams. Therefore, the projects that are being developed in the distributed environment are perceived as troublesome and intricate efforts. [55]

Following figure shows the effects of distance in GSD Setting.

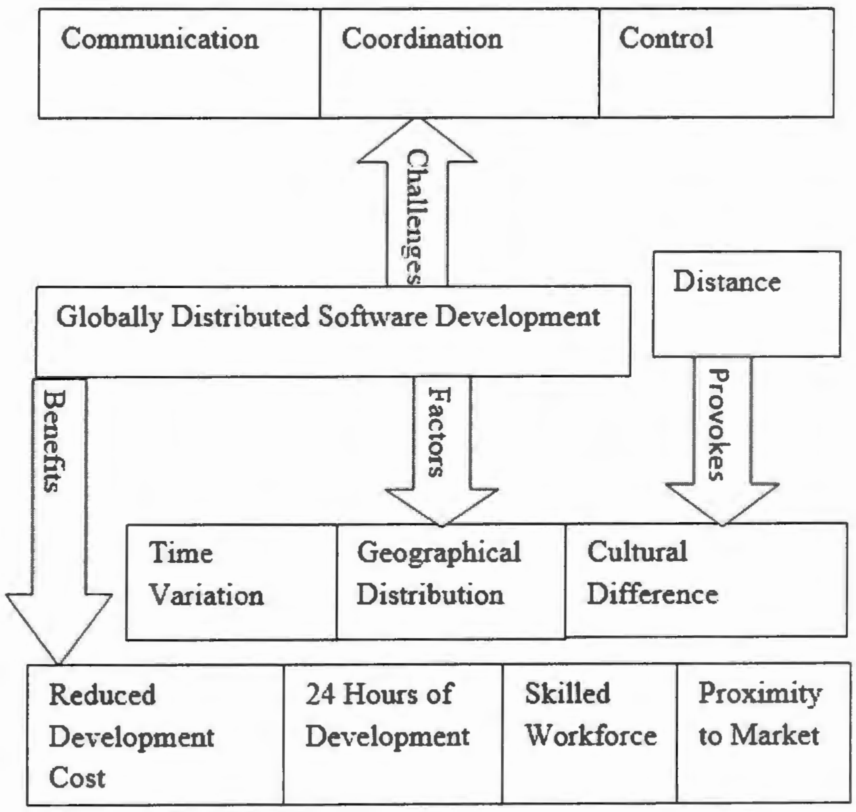


Figure 3: GSD concerns showing effects of distance

2.13. ONTOLOGIES

Thomas R. Gruber stated [59] ontology as "Ontology is an explicit specification of conceptualization". The Concepts are the description of model and knowledge of a certain domain. Ontology captures these concepts and specifies them formally by using some formal language. Conceptualization consists of objects, concepts and entities of any particular domain. There is also a relationship between these entities. No knowledge base of any type can be formed without conceptualization.

2.13.1. COMPONENTS OF ONTOLOGY

The elements of an ontology are the foundation pillar of the ontologies and irrespective of the language type and domain difference they have to use for ontology development.

1) **Classes:** Classes describe the concepts in a domain and a concept is a key component that is used to show different Individuals sharing typical features. These concepts may have instances. Classes may have subclasses that represent concepts that are more specific than the superclass, e.g. PastelColors for the superclass Colors.

2) **Individuals:** Individuals are specific instances (members) of a class. E.g. champagne pink for the class PastelColors.

3) **Attributes/Properties:** Properties describe the instances of a class regarding the parameters they can have. Properties express the relationship between individuals or from individuals to data values, e.g. Brightness level or Wavelength for PastelColors.

4) **Axioms:** Axioms are assertions that also have rules in a logical form. They contain the information described in the ontology for a specific domain.

5) **Taxonomies:** Hierarchical semantic relationship. Among concepts is provided by taxonomies.

6) **Non-taxonomic relations:** Non-hierarchical semantic relationships between concepts (classes or instances) are expressed through non-taxonomic relations.

2.14. BENEFITS OF USING ONTOLOGIES IN GSD ENVIRONMENT

Researchers emphasized on strengthening communication mechanism and minimizing ambiguities for a GSD project to work, because it is really difficult to understand and share the common terminology about the problem domain from distant places, therefore researchers accepted the ontology as a good candidate that can resolve communication and knowledge management issues [21]. The benefits of using ontologies in GSD settings are highlighted, some of them are briefly mentioned.

- Ontologies provide a shared vocabulary that is designed by a consensus of all GSD stakeholders.
- Ontologies facilitate in transferring knowledge and simplifying the development process from project to project.

- Ontologies when as machine–understandable representations, helpful in the development of tools for software engineering activities.
- Ontologies help in conflict resolution between concepts and attributes of a specific domain that can save time and effort throughout software project development.
- Ontologies assist in simple knowledge acquisition process.
- Ontologies support shared comprehension among programming engineers, and also being utilized as domain models.
- Ontologies can help in developing benchmarks of software process by data collection medium on the Internet and the use of the Semantic Web.

2.15. Comparative Analysis of the Related Work

This segment is composed of the existing literature about the requirement change management in global software development environment, requirement traceability and ontologies. It also represents the comparative analysis of the related work. Authors [5] designed a framework for managing requirement change in globally distributed software projects. They stated that to make a change in requirement is an uphill task and it becomes more difficult while dealing with multi-site projects due to lack of communication among distributed stakeholders and this communication barrier makes requirement change management process less efficient. Consequently, they presented such framework that can minimize the communication matters in global distributed setting. The model encompasses the required activities of the requirement change management process and likewise cuts down the challenges occur during Global Software Development. The suggested model is measured by two different methods (i) evaluation in real time environment through case study (ii) evaluation through comparison with other models offered in recent literature. The framework also addressed the issues in GSD setting such as culture and language differentiation and insufficient communication among globally dispersed stakeholders. Although GSD Issues are highlighted, but they don't offer any detail about how their model is addressing these subjects, no detailed flow or process is defined on how to communicate between the different steps of the framework and also control is not defined. They need to show such process or flow that illustrates the matters of GSD.

Another framework [13] is proposed for the requirement management activity in global software development. Their approach assists stakeholders by establishing requirement repository, by generating traceability matrix and establishing a communication plan for the development team which facilitates the stakeholders in a proper manner. They validate their method by applying case study in real time environment However, their requirements change management process lacks some activities, and few of these activities are not discussed, the alternate conditions of change are not present.

A model is presented [14] for managing requirement's change using ontology in multi-site software development. Ontology is more appropriate choice when requirements dynamically change because it minimizes the ambiguities among distributed stakeholders. Moreover, they stated that Communication issues could be resolved by accomplishing knowledge management. In the first step, they performed "Specification" from literature (related to GSD & RCM). The next step called "Conceptualization" in which conceptual model (comprises on concepts and relationships) was made which helps in ontology development. protégé tool is used for ontology development. "Implementation" is the final step in which ontology is developed using protégé. The developed Ontology is evaluated through case study and they proved that ontology RCM-GSD facilitates in knowledge sharing and management.

Another model for requirement change management for the global software environment (GRCM) is presented in [15] and validated the model through case study. The model incorporates the major change activities with a collection of activities, roles and artifacts (ARA) but the model has a shortcoming as for communication nò mediated technology was proposed, Furthermore the model can't be used in the later phases of the software development life cycle (SDLC).

A graph-based model is proposed for managing change in requirements in globally distributed software projects [16]. Their approach accommodates the GSD team members in bringing information about the current and modified requirements, identifying the scope of change that can disturb the project work at various locations and analyzing the key changes in requirements with reference to the development responsibilities at diverse localities, and making decision about change with the agreement of GSD teams. Their method addresses the GSD concerns, especially communication can be improving among dispersed teams and GSD stakeholders require adequate time to implement this approach due to detailed RCM steps, Moreover the model is not validated in real time environment.

A framework for requirement change management is proposed in [17] using an ontology for knowledge management and interchange. The framework was named as an Ontological Requirement Change Management (ORCM) model for GSD. Any software organization can acquire this ORCM model for the improvement of software products. Unified Modeling Language (UML) facilitates the proposed framework as UML minimizes the communication and collaboration issues in globally distributed software development. The framework was carried out in a small organization and expert judgment reviews have been practiced for the framework assessment. They used built-in ontology for their framework and defined the ontology at an abstract level, but not provide the detailed mechanism.

The authors proposed the model for requirement change management in globally dispersed software development [18]. The model comprises of different activities, roles and artifacts [ARA] which are stated in the model and it encompass the central activities of the RCM process. The framework does not highlight the problems in the GSD environment like culture and language differentiation, insufficient communication and some change management activities such as change analysis, validation & verification are not incorporated. Moreover, their model is not evaluated.

An ontology-based software requirement traceability matrix (RTM) is generated which can trace vertical and lateral functional requirements [19]. Firstly, the authors constructed a RTM ontology that contains the classes and their relations and later they constructed instances of the ontology according to concepts and their properties. This approach is most appropriate to view extensive areas of traceability and it involves graph implementation for better visualization. It keeps track of all the business requirement and checks the consistency between requirements document.

A framework is presented which shows the effect of GSD factors on communication during Requirement Change Management process [20]. They stated that these three constraints become serious bottlenecks in communication, coordination and control in GSD. The proposed framework has three independent variables, namely geographical distance, socio-cultural distance and temporal distance and model examine the negative impact of these factors on the communication process. The findings from the study data indicate that these three constraints have a substantial negative effect on communication and the results demonstrated that as the impact of these elements increases the level of communication will be decreased.

Another framework is proposed in Ref [21] for requirement management in global software development. Requirement management metrics are generated for determining the whole set of requirements before and subsequently modifying the beginning requirements, the average rate of requirements and the calendar plan of requirements. Ontology is used to manage the project knowledge and authors suggested some constraints to control the requirement management activity. In spite of that the way of dealing with the requirements repository is not given and the mode of communication is not described.

Authors provided a solution for handling change in requirements [22] specifically in globally distributed projects. They claimed that GSD stakeholders must be given with the shared vocabulary for minimizing the ambiguities in requirements and an ontology is a potential candidate for knowledge sharing and management. For developing ontology, the Concepts and their relationships were separated associated to RCM in GSD and then a conceptual model was made which assists in ontology development. A case study is used for evaluating the developed ontology and to answer the competency questions.

One more framework is presented in Ref [24] for ontology based knowledge management. The approach has two phases; (1) knowledge representation and (2) knowledge management. The framework consists effective components of knowledge management, for example capture, store and deploy to address the issues of current knowledge management. Each component is described in detail. The proposed approach provides a better solution for some of the open issues of ontology based knowledge management.

Authors proposed a knowledge management tool [25] that uses a domain-specific ontology for distributed software environment. The primary aim of this study is the proposal of both the ontology and the tool which will establish a proper mechanism to facilitate the distributed software development operation. They created a common vocabulary for dispersed stakeholders, their ontology consists of 50 classes, but they mentioned the core classes and the tool supports the generation and distribution of knowledge, as well as it aids in decision making in DSD setting.

Another solution is recommended for handling change in requirements for global software development [7]. The method is made up of three steps (i) Change understanding (ii) Change analysis (iii) Change Decision. Their method uses graph for representing existing and changed requirements. They validated their approach by applying a case study and proved that the approach facilitates GSD team to manage requirement change effectively.

The comparative analysis of the related work is given in the tabular form:

Techniques	Issues	Description	Significance	Limitation
Approach for Managing Requirements Change in Global Software Development (Lai et al. 2014)	How to handle requirement change in a distributed setting.	Their graph model assists the GSD stakeholders in gaining knowledge of the existing and modified requirements.	Their method addresses the GSD concerns, especially communication can be improve among dispersed teams.	GSD team requires more time to implement this approach due to detailed RCM steps. The Model is not validated in real time environment.
Requirement Change Management Framework in Global Software Development (Minhas et al. 2014)	To handle change in requirements in a global distributed setting.	A Framework is proposed to manage changing requirements in GSD setting. It also contains the key activities of the RCM process.	Facilitates managing change in globally distributed setting. The framework was evaluated in real time on two GSD projects and the model is validated through a case study.	GSD Issues are highlighted, but they don't provide any detail about how their model is addressing these Issues, there is a need to show such process or flow that illustrates the matters of GSD.
Requirements Management Approach for Global Software Development (Lai et al. 2013)	How to tackle requirement change management activity in globally distributed setting.	A requirement repository for managing the requirement change is presented.	A repository is established for requirement's change management in GSD setting.	The requirements change management process lacks some activities, they didn't consider the substitute scenarios of change. Some RCM activities are not added in their work.
Ontological framework for requirement change management in distributed environment (Asma et al. 2014)	Handling requirement change in distributed setting.	They developed their ontology in three steps; specification, conceptualization and implementation and protégé tool is used.	Model lessen the ambiguities between dispersed team members by using ontology and knowledge management is done nicely. Moreover, the model is validated through case study.	The development of ontology can be bit lengthy because of its detailed mechanism.

Model for Global Requirements Change Management (Hussain et al. 2012)	How to manage requirements change in GSD.	The model incorporates the major change activities with a collection of activities, roles and artifacts (ARA).	The Model helps to refine the RE process by analyzing common RE activities for GSD organization.	The model is not applicable on the later development stages of the software development life cycle (SDLC).
Study the Impact of Requirements Management Characteristics in Global Software Development Project: An Ontology Based Approach (Kumar et al. 2011)	How to control requirement management process in GSD.	Requirement management metrics are generated for determining the complete set of requirements before and after modification.	Ontology is used for knowledge management. Ambiguity can be minimized among team members while using ontology.	In spite of that the way of dealing with the requirements repository is not given and description of communication mode is likewise lacking.
A method of RCM for Global Software Development (Ali et al. 2016)	How to manage change in requirements for GSD projects.	The method is made up of three steps. Understanding, Analyzing & Finalizing the changes in the requirements and requirement is represented by a graph.	Their graph based approach helps the team to build up a clear understanding of the existing and modified requirements. Mathematical measures are defined for analyzing the change.	Their RCM approach is a lengthy process and GSD team needs sufficient time to follow the method.

Table 1: Summary of Comparative Analysis Of Existing Work

In this section, the overview of the requirements engineering, distributed software environment and ontologies are presented. Besides it incorporates the existing literature on requirement change management process in distributed setting, requirement traceability and ontologies that helps in getting better understanding of the domain.

Chapter 3

MATERIALS AND METHODS

3.1.BACKGROUND

As previously stated, software requirements change management is a crucial activity to be practiced at the right time, during a software development life cycle, specifically in a distributed software environment where stakeholders are dispersed at different positions. It is also important to note that software project evolves over time and evolving stakeholder's needs is a genuine reason that can demand change and this requirement change can affect the other system components and business objectives as well. But the matters of a requirement change, transform in different form completely in the case of Global Software Development. The GSD main issue is communication that arises many other challenges. Research indicates that managing change, analyzing impact of change and collaborating change to distant colleagues is a strenuous task that outcomes in misunderstandings and confusions among the GSD group. In our research, we presented the solution to the above mentioned problems in the form of such requirement management process for global software development projects that can handle change in requirements along with an effective impact analysis. Besides process also embeds the common vocabulary that is outlined after accomplishing accord of the GSD group as the common vocabulary is valuable in knowledge management and minimize the perplexities and ambiguities brought up in their minds. There is no standard framework available that deals with the requirement change with the activity of impact analysis in GSD domain. An ontology is a strong candidate for managing knowledge and minimizing misapprehensions and ambiguities by offering an agreed upon vocabulary for requirement management process in Global Software Development.

In this chapter, a framework is offered along with the details of its phases. Foremost of all, we will design a concept model of the framework in which the concepts and relations (related to Requirement change management, Requirement traceability matrix and Global software development are determined. Defining concepts (classes) and their relationships, and objects of a group of requirements will be examined alongside the definition of axioms. Unified modeling language (UML) is used for making conceptual models. The key concepts, properties and axioms of RCM, RT, and GSD are expressed with the help of the conceptual model. For which purpose, Violet UML Editor is used.

Protégé tool and OWL (web ontology language) is utilized for ontology development that expresses the concepts and attributes related to Requirement change management, Requirements traceability matrix and Global software development, while to assess the impact due to change on requirements or a module, Requirements traceability matrix (RTM) is being nominated.

C. Roussey et al. said that ontologies can be of different kinds [61] such as domain ontologies, application or local ontologies, foundational or top level ontologies and task ontologies and this classification is done on the base of scope of the ontology. Domain ontology can be applied in such domain that have a particular view point e.g. electric network management system, medicine, bioinformatics, and software engineering. Foundational ontology is generic ontology that can be applied in many fields. It helps in describing the general concepts e.g. objects, relations, time, events, processes and actions. Task ontology represents those concepts that relate to the common tasks of a specific domain. Lastly, Application ontology is the blend of task and domain ontology. Consequently, we built up an application ontology, since requirement management and distributed software development are associated with the discipline of software engineering.

3.2. ONTOLOGY DEVELOPMENT PROCESS

There are several methods of developing the Ontology that has different sorts of activities, but somehow all of them are used for ontology development and no individual method is considered as a benchmark therefore ontology developer follows these development steps according to their own needs. In my thesis, I have developed the "RM-GSD" Ontology that is based on mainly four activities. These activities are:

1. Specification
2. Conceptualization
3. Formalization
4. Implementation

The following figure illustrates the steps of ontology development process.

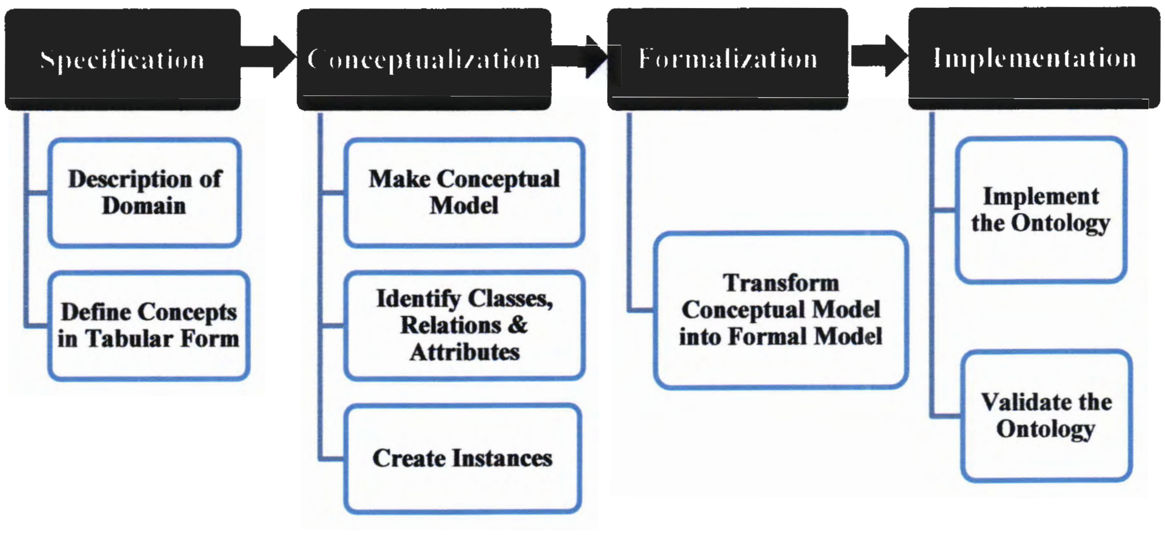


Figure 4: Framework for Ontology Development

3.3. Specification of the RM-GSD Ontology:

The following part describes the tasks of specification.

3.3.1 Description of the Domain

The domain knowledge of Requirement change management (RCM), Requirements traceability (RT) and Global software development (GSD) is presented in this section.

3.3.1.1 Description of Requirement change management (RCM)

We consulted a few papers for setting up better comprehension of Requirement change management (RCM) process. Many authors suggested those RCM processes differently, but following are the normal exercises for change management. The RCM process comprises on different activities and roles such as change request originator, change control board (CCB), change implementer, quality assurance team. The change management process starts when a client or any team member of the project demand for a change and change request form (CRF) is used for submitting the change request by the change request originator.

The change request form (CRF) holds the complete details of the change request and contact details of the change request originator. In the next step, analyst evaluates the change for cost estimation, time frame and resource utilization. The decision about the acceptance or rejection of change depends upon the feedback on evaluation results by CCB, if all individuals have accord on the assessment results, change is acknowledged for acceptance and approved for execution, otherwise change request will be rejected. Change is implemented by the development team as indicated by the calendar plan to meet the deadline. After that, quality assurance team tests the functionality of the implemented change and assures the quality of the project.

3.3.1.2 Description of Requirement Traceability Matrix (RTM)

Requirement Traceability Matrix (RTM) is a vital artifact in requirement management, which summarizes the relationships between requirements and test cases. In actual fact, Traceability information, being able to know from where you come and where you going. It indicates which requirement is implemented by which test case (one or more test cases) and which test case tests which requirement (one or more requirements). RTM is generated by using a table (in excel) that shows the requirements and test cases at (horizontal) x-axis and (vertical) y-axis respectively.

In case of a requirement change, RTM is checked for knowing that which test case cover that requirement and then the affected test case is modified for meeting the new requirement. RTM has some common attributes that is followed in every project. These are: Project name, Project Description, Project Objective, Business Need, Functional Requirements, Nonfunctional Requirements, Requirement Description, Relationship b/w requirements. etc.

3.3.1.3 Description of Global Software Development (GSD)

Software companies outsource their projects to globally distributed sites for having potential benefits such as access to the large skilled workforce, access to cost-effective labor pool, 24 hours of development and nearness to market, but on the other side, GSD projects face numerous difficulties of communication, coordination and control due to geographic, cultural and time zone differences. RCM is a collaborative-intensive activity, it requires strong communication mechanism for managing change in time and GSD demands a quick response to change so there is an intense need to establish a such RCM process that improvises the communication between the dispersed stakeholders.

3.3.2 Specification: Definition of the Concepts in tabular form

In this step, concepts are defined in tabular format which assist in converting the information into a conceptual model. Following tables present the concepts and concept relationships related to Requirement Change Management (RCM), Requirement Traceability Matrix (RTM) and Global Software Development (GSD) respectively.

RCM	**GSD**	**RTM**
Client/Customer	GSD Project	Project Name
Change Request Form	GSD Benefits o Skilled workforce o Low cost labor o 24 hours' development cycle o Latest Technology	Project Description
Change Requester/Initiator o Change Req. Name o Change Req. Location o Change Req. Contact Number o Change Req. Department and Designation	GSD Factors o Cultural differences o Geographical distance o Time zone o Knowledge Management	Project Start Date
Change Request Details o Change Request Date o Change Request Time	GSD Challenges o 3 C's (Communication, Coordination & Control)	Project Completion Date
Change Evaluation o CCB (Change Control Board)	GSD Teams	Requirements Id
Change Implementation o Development Team o Requirement Analyst o Designer o Programmer	Roles o User o Team Members	Requirement Status o Started o In-process o Withheld o Completed
Change Request Log/History o Updated Change /Modified Requirement	Project Activities	Functional Requirements
Allocate Resources for change implementation (time and cost)	Project Artifacts o Change Request Form o Use Cases & Test Cases o Change Review Report	Nonfunctional Requirements
Change Verification o Quality Assurance Team o QA Manager o S/W Tester	Location/Site	Use Cases
	Communication o Synchronous Communication o Asynchronous Communication	Test Cases

Table 2: Concepts of RCM, RT and GSD

The accompanying table illustrates the classes, relationships, domain, range and constraints of our ontology as a whole.

Description of Class	Name of the Relationship	Domain	Constraints	Range	Constraints
Change Originator initiates change request	initiates	Change Originator	1	change request	1..*
Change request submitted through change request form	submitted through	Change request	1	change request form	1
Change request form is received by change manager	receivedBy	Change request form	1	change manager	1
Change request form describes change	describes	Change request form	1	Change	1..*
Change request form need to be evaluated in Decision making activity	isEvaluatedIn	Change request form	1	Decision Making	1..*
Decision making results in impact analysis document	evaluatedChang e Results	Decision Making	1	impact analysis document	1
Impact analysis document provide support for change	support	Impact analysis document	1	Change	1..*
Rejected request is roll backed.	rejected request	Decision Making	1	Rollback Change	1..*
Rollback change notified to the change Originator	notifiedTo	Rollback Change	1..*	Change Originator	1..*
Change requires change in schedule	requires	Change	1..*	Change schedule	1..*
Change schedule output artifact schedule plan	outputArtifacts	Change schedule	1	schedule plan	1..*
Change schedule is scheduled by change manager	scheduledBy	Change schedule	1..*	Change Manager	1..*
Change is implemented in change implementation	isImplementedIn	Change	1..*	Change implementation	1..*

Change implementation is performed by change implementer	isPerformedBy	Change implementation	1..*	change implementer	1..*
Schedule plan is followed by change implementer	isFollowedBy	schedule plan	1..*	change implementer	1
Implemented change is verified in change Verification	isVerifiedIn	Change	1..*	Change verification	1
Change verification is accomplished by quality assurance team	accomplishedBy	Change verification	1..*	quality assurance team	1..*
In change verification Quality assurance team uses test cases for verification	uses	Change Verification	1..*	test cases	1..*
Change verification failure results to review report	failureResultsTo	Chang verification	1	review report	1
Verified change is validated in change validation	isValidatedIn	Change	1..*	Change validation	1..*
Change validation is done by the change request originator	doneBy	Change validation	1..*	change request originator	1..*
Change validation failure outputs in review report	results in	Change validation	1	Review report	1
Review report is re-evaluated in decision making	re-evaluated in	Review report	1..*	Decision Making	1..*
Validated change is closed in change closure.	isClosedIn	Change	1..*	Change closure	1..*
Closed change is updated to version	isUpdatedTo	Change closure	1	Version	1

Table 3: T-Box of RM-GSD Ontology

The Following tables represent the concept description and relationships of RCM, GSD and RT separately.

Class Description	**Properties/ Relationships**	**Domain**	**Range**
Customer hasInterest in Project	worksOn/ hasInterest	Customer	Project
Change Request canInitiate from any Site	canInitiate	Change Request	Site
Communication ismandatory for all Sites	ismandatory	Communication	Sites
Change Request Form IsSubmittedBy Change Requester	isSubmittedBy	Change Request Form	Change Requester
Change Requester Has Name Change Requester Has location Change Requester Has Contact No. Change Requester Has Deptt. & Designation	has has has has	Change Requester Change Requester Change Requester Change Requester	Name Location Contact No. Department and Designation
Change isRequested on Date Change isRequested on Time	isRequested isRequested	Change Change	Date Time
Change Evaluation IsPerformedBy CCB	isPerformedBy	Change Evaluation	CCB
Change Implementation isPerformedBy Development Team	isPerformedBy	Change Implementation	Development Team
Updated Change- isReceivedBy Project Manager	isReceivedBy	Updated Change	Project Manager
Resources areAllocated to Change Implementer	areAllocated	Resources	Change Implementer
Change verification isPerformedBy QA Team	isPerformedBy	Change verification	QA Team

Table 4: T-Box for Requirement Change Management

Interrelations b/w Classes	**Properties**	**Domain**	**Range**
Requirement-Req. Id	has	Requirement	Req. Id
Project- Starting Date	has	Project	Starting Date
Project- Completion Date	has	Project	Completion Date
Teams-All sites	workson	Teams	All sites
Teams- Roles & Responsibilities	performs	Teams	Roles & Responsibilities

Table 5: T-Box for Requirement Traceability Matrix

Interrelations b/w Classes	**Properties**	**Domain**	**Range**
GSD Project- Benefits	offers	GSD Project	Benefits
GSD Project- GSD Factors	isAffectedBy	Project	GSD Factors
GSD Factors- GSD Challenges	increases	GSD Factors	GSD Challenges
GSD Teams- Different sites	perform	GSD Teams	Different sites
Roles- Project Activities	perform	Roles	Project Activities
Project Activities- Artifacts	produces	Project Activities	Artifacts

Table 6: T-Box for Global Software Development

3.4. Conceptualization of the RM-GSD Ontology: Definition of Conceptual Model

A conceptual model is produced by using the above table (given in Specification step) which represents the main concepts and the association between concepts. The motivation behind making the conceptual model is to understand and separate the domain concepts and their properties that will assist during the time of implementation. Unified Modeling Language (UML) is used to conceptualize the concepts and relations among those concepts. UML is applied as a graphical representation language that explain the elements of our ontology and conceptual model is all about this graphic depictment. Finally, the conceptual model is worthwhile for building an ontology using a tool as it captures the important entities and our conceptual model is grounded on concepts, properties and cardinalities. The Conceptual Models for Requirement Change Management (RCM), Global Software Development (GSD) and Requirement Traceability Matrix (RTM) are presented below:

3.4.1 Conceptual Model for Requirement Change Management (RCM)

The Conceptual Model for Requirement Change Management (RCM) is composed of concepts, relationships and constraints. The key concepts are defined as super classes and other classes will be set as sub classes. Requirement Change Management (RCM) process starts when stakeholder demands for change. This change is recorded on a paper that is well-known as a change request form (CRF), this CRF asks for complete details about change initiator and the change itself. Afterwards that, this change request form (CRF) is forwarded to the project manager officer (PMO) for fulfilling the new demand. Now the change control board (CCB) takes on board for impact analysis of the change whether the change is suitable for development and it will not cause any harm to the other components of the system. The change control board (CCB) is the representative body of all the stakeholders where all the major changes are discussed, evaluated and impact analysis is done where the future of change is finalized.

Analyzing the effect of change is more significant job [10] during requirement change management (RCM) process and is also useful for managing change evolution. After the change evaluation, change control board (CCB) takes a decision based on the evaluation report. If the evaluation results have a negative impact, then the change request will be discarded and project manager informs the concerned party. If the change evaluation report is positive and this change is considered worthwhile for the system, and so a change schedule is planned for implementing the new change and change implementer follows that timeline given by the project manager. After that, quality assurance team uses the test cases and verifies the functionality of the implemented change and assures the quality of the project.

In Addition, more than one change requests can be requested and a change request form (CRF) will be used for a single change request. Every request has its own individual change request form (CRF) submitted by the change request originator.

3.4.2 Conceptual Model for Global Software Development (GSD)

The Conceptual Model for Global Software Development (GSD) is made up of concepts and their associations in the Global Software Development (GSD) domain. "GSD Project" is the super class that offers some "GSD Benefits" which are defined as subclasses of the "GSD Project".

GSD Project gives the potential benefits like access to skilled professionals, state of the art technology, low labour cost and 24 hours of development. On the other hand, Global Software Development (GSD) projects face many challenges like geographic and cultural distribution, time variation and knowledge management, and these challenges create a hindrance in communication, coordination and control. Distance is the key issue in distributed software development that trigger many other problems and the flow of information delays from different locations due to the 3 C's (communication, coordination and control). The main concepts of the Global Software Development (GSD) environment is captured as classes, and classes are represented as Boxes and the relationships are shown by the arrows. As mentioned before, Unified Modelling Language (UML) is utilized for conceptualizing the entities and relationship among those entities.

3.4.3 Conceptual Model for Requirement Traceability Matrix (RTM)

The Conceptual Model for Requirement Traceability Matrix (RTM) also has the main entities of the Requirement Traceability Matrix (RTM). This is the brief model of the (RTM) that contains the main concepts along with their relationships. This model conceptualizes the concepts like "Requirements", "Use cases", "Test cases", "Requirement Type" and "Design". These concepts will be used while developing ontology using Protégé, for this purpose an RTM will be designed that will define these classes with their properties of RTM. Moreover, functional requirements and non-functional requirements will be mapped in Requirement Traceability Matrix (RTM) and then ontology of Requirement Traceability Matrix (RTM) will be developed by using ontology development tool Protégé. The next chapter contains these implemented details. The following figure illustrates the conceptual model of RM-GSD ontology.

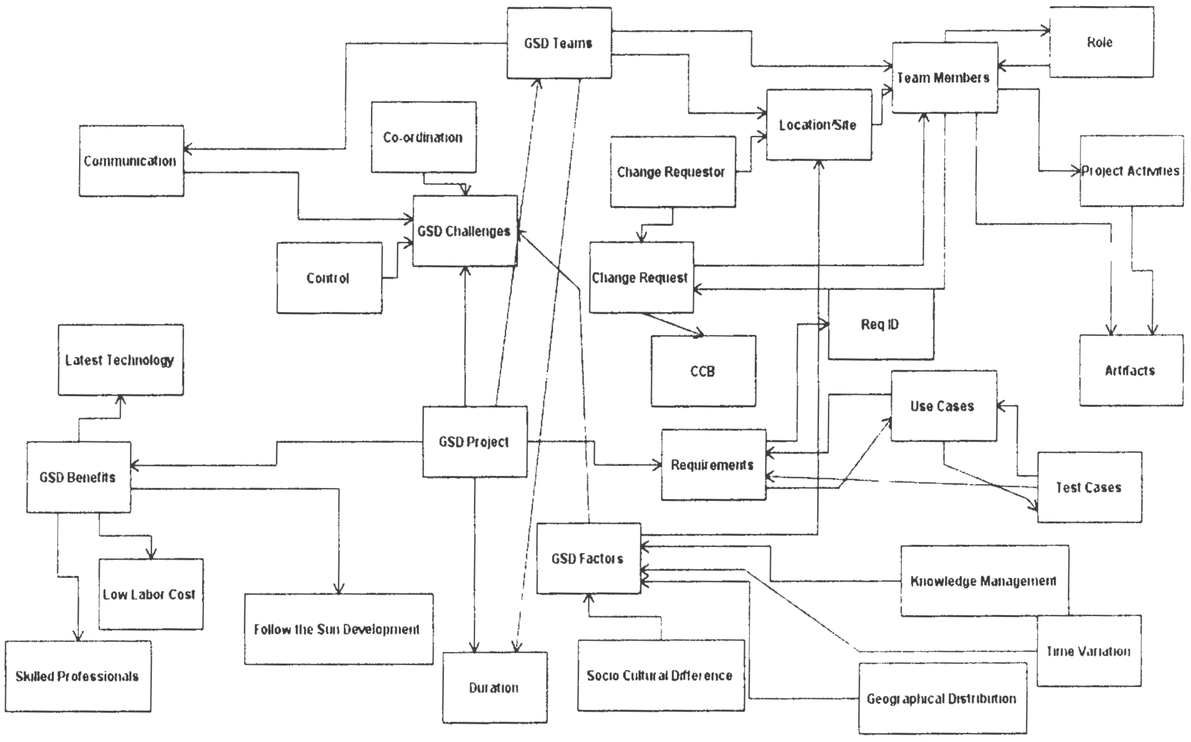


Figure 5: Conceptual Model of RM-GSD Ontology

3.5. Formalization of the RM-GSD Ontology:

The goal of this activity is to formalize the conceptual model into a formal model therefore formally the concepts and relations between concepts can be formalized as C = {GSD Project, GSD Factors, GSD Challenges etc.} where each $c_i \in C$ and R= {offers, follows, has, etc.} where each $r_i \in R$. Entities are shown as rectangles and the relationships are shown using arrows and direction from which entity to another entity. Unified Modeling Language (UML) is used to formalize concepts and relations.

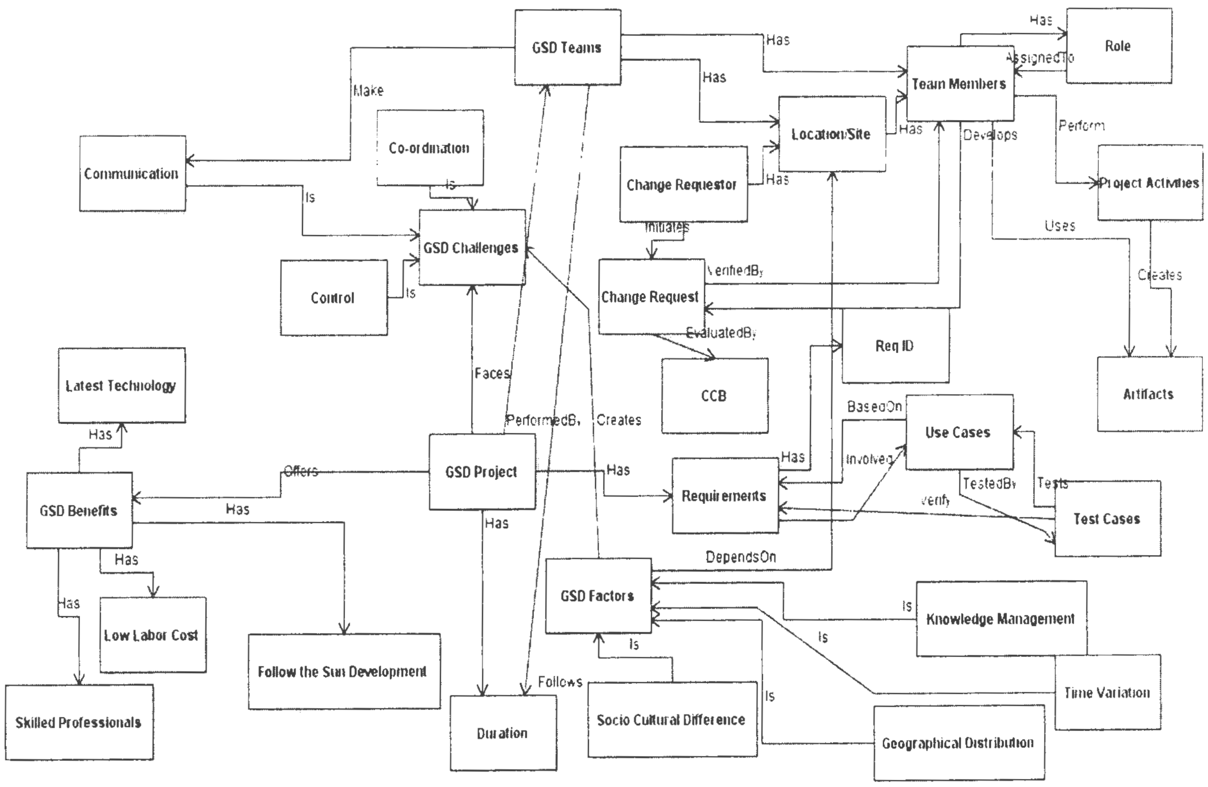


Figure 6: Formalization of RM-GSD Ontology

3.6. Implementation of RM-GSD Ontology:

Following section contains the implementation details of our Requirement Management for Global Software Development (RM-GSD) ontology.

3.6.1 Ontology Development Language:

Requirement Management for Global Software Development (RM-GSD) ontology is implemented by selecting most suitable ontology development language, development tool. There are many ontology development languages that supports differently, for example; Web Ontology Language (OWL), Resource Description Framework (RDF), Agent Markup Language (DAML) and Interchange Language (OIL). We have selected OWL for ontology implementation as it provides a perfect set of expressions for eliciting concepts and relations. Subsequently ontology development tool will be chosen after selecting ontology development language.

3.6.2 Ontology Development Tool: Protégé

Protégé is a versatile and configurable platform for the development of arbitrary model-driven applications and components. It has an extensible and customizable tool set for building ontologies and for developing applications that use these ontologies [52,53]. Protégé has some prominent features:

- Automatic generation of graphical-user interfaces that focused on user-specified models.
- Empowering the scalability of ontologies to the huge knowledge bases.
- Extensible knowledge model and architecture.

The Protégé OWL editor is a free and open source that represents different types of classes, properties, individuals and rules, as well as it provides the features of editing and browsing for OWL models, and so supports the rapid application development. [54]

The Protégé platform allows to generate and edit (Resource Description Framework) RDF schema and Web Ontology Language (OWL). Requirements Management for Global Software Development (RM-GSD) ontology in OWL is developed by using the above standards and consistency checking is performed using a HermiT reasoner. Hermit helps in finding out the taxonomic relationships, hence it will help in removing ambiguities, sharing and managing domain knowledge about requirement's change in global software distributed environment. The Graphical User Interface (GUI) of Protégé 4.3 is composed of some tabs that are utilized for holding concepts and properties. These tabs are entities, classes, object properties, data properties, individuals, Owl Viz, DL Query, Onto Graf, SPARQL Query for class hierarchy and consistency checking. In the figure, highlighted part is showing all the tabs in which well suited data need to be identified.

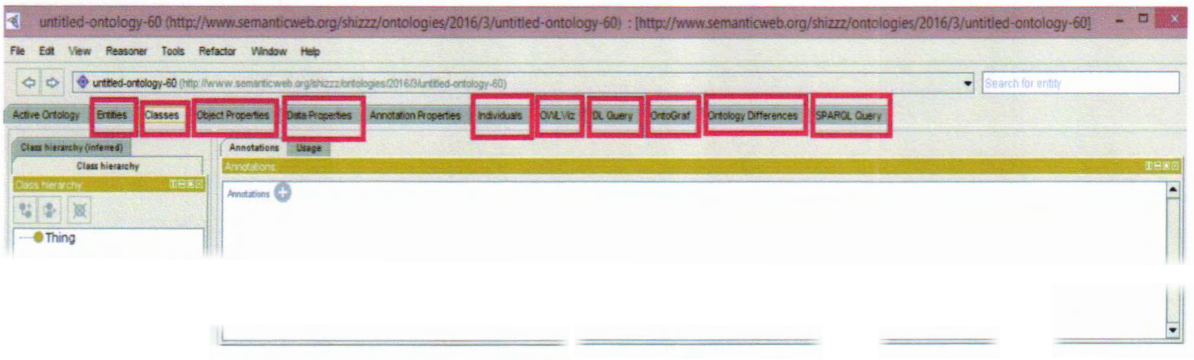

untitled-ontology-60 (http://www.semanticweb.org/shizzz/ontologies/2016/3/untitled-ontology-60) : [http://www.semanticweb.org/shizzz/ontologies/2016/3/untitled-ontology-60]
File Edit View Reasoner Tools Refactor Window Help
untitled-ontology-60
Search for entity
Active Ontology
Entities
Classes
Object Properties
Data Properties
Annotation Properties
Individuals
OWLViz
DL Query
OntoGraf
Ontology Differences
SPARQL Query
Class hierarchy (inferred)
Class hierarchy
Annotations
Usage
Thing

Figure 7: Protégé Interface

3.6.3 Use of Reasoner:

A Reasoner is a software or tool that extracts new facts from the existing ontologies. A semantic reasoner is used to deduce the logical statements from any set of Rules or Axioms. Protégé offers two types of models asserted and inferred. Human beings face difficulties while storing the knowledge in a knowledge base rather knowledge extraction should be implicit. Reasoner assists for reasoning tasks like classification, debugging and querying. [48] Some different kinds of reasoners are available such as Pellet, HermiT, FACT++, RACER, CEL, ELK, TrOWL etc. Some ontology reasoners are described briefly as under.

- **HermiT:**

HermiT is the first open source reasoner and is written in a web ontology language (OWL), It is helpful in consistency checking of ontologies and it also determines the taxonomic relationship among entities. It supports classification of complex ontologies. [24]

- **Pellet:**

Pellet is another open source reasoner software that supports description logics. Pellet also gives the explanation of bugs and supports incremental consistency checking. Furthermore, it provides services for analyzing, debugging and repairing of ontologies [24].

- **FACT++:**

FACT++ (Fast Classification of Terminologies) is an open source reasoner that is implemented in C++. This reasoner implements the procedure of tableaux decision with the support for all data types of strings and integers [24]. It supports the Description logic (DL) and web ontology language (OWL).

3.6.4 Adding Classes in Protégé 4.3:

Requirement change management and requirement traceability in global software development ontology is a collection of information about the project's domain that further transforms in concepts or classes. Now the next step is to select super-classes and sub-classes from among the identified objects and few of them are named in the following table. The root class or parent class of the Protégé is called 'Thing' and all the super classes thus formed by ontology developers are the subclasses of Thing. An individual can have more than one class, consequently they are declared as disjoint classes that cannot hold the common members. This attests that an individual belongs to one or more classes but if a member is from more than one class, these classes must not be disjoint with each other. Main concepts or classes related to requirement change management (RCM), requirement traceability (RT) and global software development (GSD) are listed below. For example; In the ontology of RTM, Use_Cases, Test_Cases, Design are disjoints because these concepts/classes are different from each other. At this point in time we have developed strong enough understanding about the concepts, object properties, data properties and rules that can be mapped in the domains of requirement change management (RCM), requirement traceability (RT) and global software development (GSD).

The concepts of our RM-GSD domain are separated that will direct in the ontology development process. Following section comprises on the rigorous analysis of the concepts of Requirement change management (RCM), Requirement Traceability Matrix (RTM) and Global Software Development (GSD) separately.

Super Classes	**Class Description**	**Sub Classes**
Thing	The Thing is the parent class in Protégé, its union of all classes.	Roles Activities Artifacts Change Version
Activities	The actions that must be done for project completion.	Change Request Change Evaluation CCB Decision Change Schedule Change Implementation Change Verification Change Validation Change Closure
Roles	The Role is assigned to each team member that has some responsibility to perform.	Project Manager Change Request Originator Change Implementer S/W Tester CCB User
Artifacts	Artifacts refer to deliverables that are performed by Roles.	Change Request Form Schedule Plan Use cases & Test Cases Implemented Change Change Review report Change Request Log/History Version

Table 7: Concept section of super classes & sub classes of RCM

Super Classes	Class Description	Sub Classes
Thing	The Thing is the root class in Protégé, its union of all classes.	GSD Project GSD Benefits GSD Factors GSD Challenges GSD Location
GSD Project	Any project that is developed in in multi-site, therefore a project that is being developed in GSD setting is a GSD project and This is the subclass of Thing.	GSD Teams Project Requirements
GSD Benefits	This is the subclass of Thing that have other subclasses.	Skilled Workforce Follow the Sun Development Latest Technology Low Labor Cost
GSD Factors	GSD Factors is another subclass of Thing that have other subclasses.	Time Variation Geographical Distribution Socio Cultural Differences Knowledge Management
GSD Challenges	GSD Challenges is some other subclass of Thing that have other subclasses & faces by GSD teams.	Communication Coordination Control
GSD Location	This is the subclass of Thing that have other subclasses.	Site 1 Site 2

Table 8: Concept section of super classes & sub classes of GSD

Super Classes	**Class Description**	**Sub Classes**
Thing	The thing is rooted class that always exists which deals out all the concepts and individuals.	Requirements Requirement Type Design Use Cases Test Cases
Requirements	Requirements is the subclass of Thing that describe the user's demands.	Business Requirements
Requirement Type	Requirement Type is another subclass of Thing that has information about the Requirement Type.	Functional Requirements Non- Functional Requirements
Design	Design is the subclass of Thing that is divided in two more sub classes.	Architecture Design Detailed Design
Use Cases	A subclass of Thing that has information about the Use Cases.	UC_01 UC_02 UC_03 UC_04
Test Cases	A subclass of Thing that has information about the Test Cases	TC_01 TC_02 TC_03 TC_04

Table 9: Concept section of super classes & sub classes of RTM

3.7. Competency Questions:

Competency Questions play a vital part in the ontology engineering as they define the scope of an ontology. Competency Questions represent the ontology requirements and an ontology must be competent enough to answer the competency questions. For our RM-GSD Ontology, the following are the possible competency questions:

1. Who initiated change request?
2. Which location can initiate change?
3. Which requirement affects the use case?
4. Who is the system admin?
5. Can PCM1 initiate change?
6. Which requirement impact the other requirements?
7. What Change Request was initiated by PCM1?
8. Who validates the implemented change?

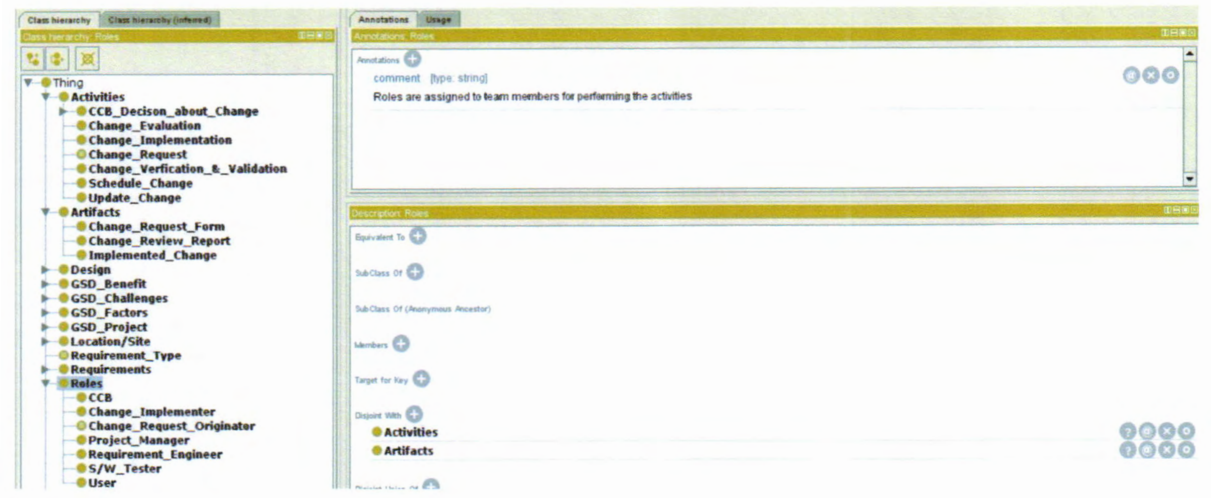
3.8. RM-GSD Ontology:

We have developed a requirement management ontology for a global software development environment that is made up of some important elements that establish a standing ground for an ontology. Following segment represents the implemented screenshots of the RM-GSD ontology that are developed in Protégé 4.3.

1. Classes and class hierarchy
2. Object Properties
3. Data Properties
4. Define Instances
5. Axioms or Rules of ontology

3.8.1 Define the Classes and class hierarchy of RM-GSD Ontology:

The classes (also recognized as concepts) are the founding blocks of an ontology and it captures the background knowledge of a specific domain. The size of an ontology varies; small or large as it is determined by the developer that how much the developer want to cover the domain. The classes and class hierarchy are shown in the following figure. For example, "Activities" and "Artifacts" are subclasses of the root class "Thing" that are declared as disjoint classes and both classes have some subclasses also. The following figure is showing the classes and class hierarchy of RM-GSD ontology by using different tabs of Protégé 4.3.



```
Thing
  Activities
    CCB_Decison_about_Change
    Change_Evaluation
    Change_Implementation
    Change_Request
    Change_Verfication_&_Validation
    Schedule_Change
    Update_Change
  Artifacts
    Change_Request_Form
    Change_Review_Report
    Implemented_Change
  Design
  GSD_Benefit
  GSD_Challenges
  GSD_Factors
  GSD_Project
  Location/Site
  Requirement_Type
  Requirements
  Roles
    CCB
    Change_Implementer
    Change_Request_Originater
    Project_Manager
    Requirement_Engineer
    S/W_Tester
    User
```

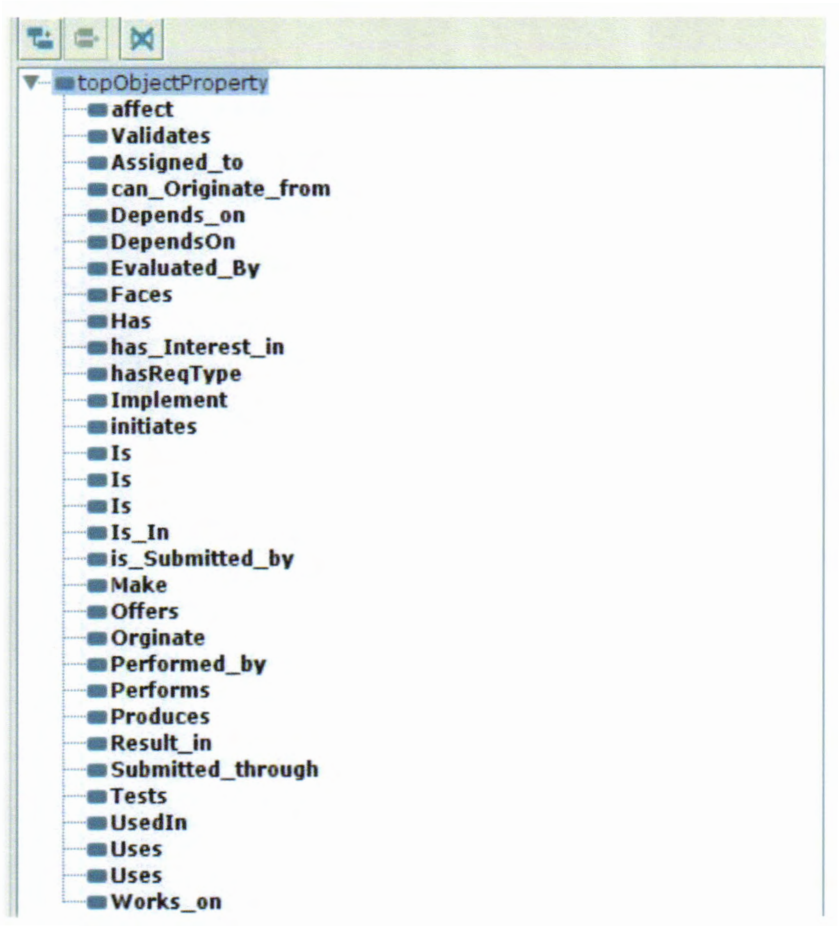
comment [type: string]
Roles are assigned to team members for performing the activities

Disjoint With
Activities
Artifacts

Figure 8: Protégé Interface Showing Entities with taxonomic hierarchy

3.8.2 Define the Object Properties of RM-GSD Ontology:

The object properties consist of domain and range that creates a link between two different Individuals. For example, "GSD Project" and "GSD Challenges" are two separate classes that are associated with an object property "faces". In this example: "GSD Project" is the range and "GSD Challenges" is the domain that are linked by object property "faces", Hence we can describe it as GSD Project faces GSD Challenges. In this work many object properties are used that are presented in the following figure.

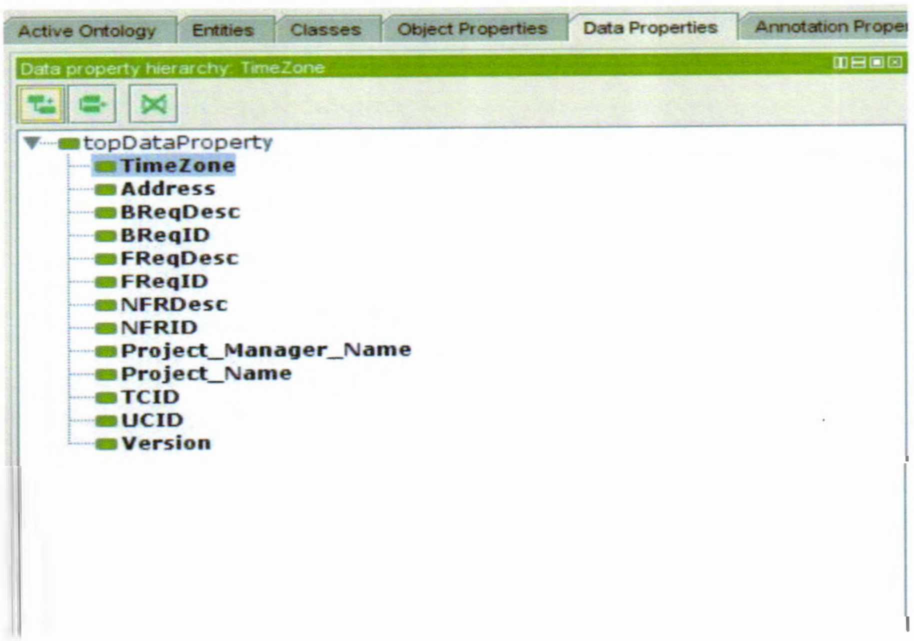


```
topObjectProperty
    affect
    Validates
    Assigned_to
    can_Originate_from
    Depends_on
    DependsOn
    Evaluated_By
    Faces
    Has
    has_Interest_in
    hasReqType
    Implement
    initiates
    Is
    Is
    Is
    Is_In
    is_Submitted_by
    Make
    Offers
    Orginate
    Performed_by
    Performs
    Produces
    Result_in
    Submitted_through
    Tests
    UsedIn
    Uses
    Uses
    Works_on
```

Figure 9: Object Properties of RM-GSD Ontology

3.8.3 Define the Data Type Properties of RM-GSD Ontology:

Data properties are used to describe the association among classes and data values. Some RM-GSD classes can be represented by data values. For example, a requirement or a test case requires to be represented by some ID. The purpose of defining the data properties is to save the data value, For Instace, adding the property name, address and contact # etc. Following figure is showing the data properties that have been used.

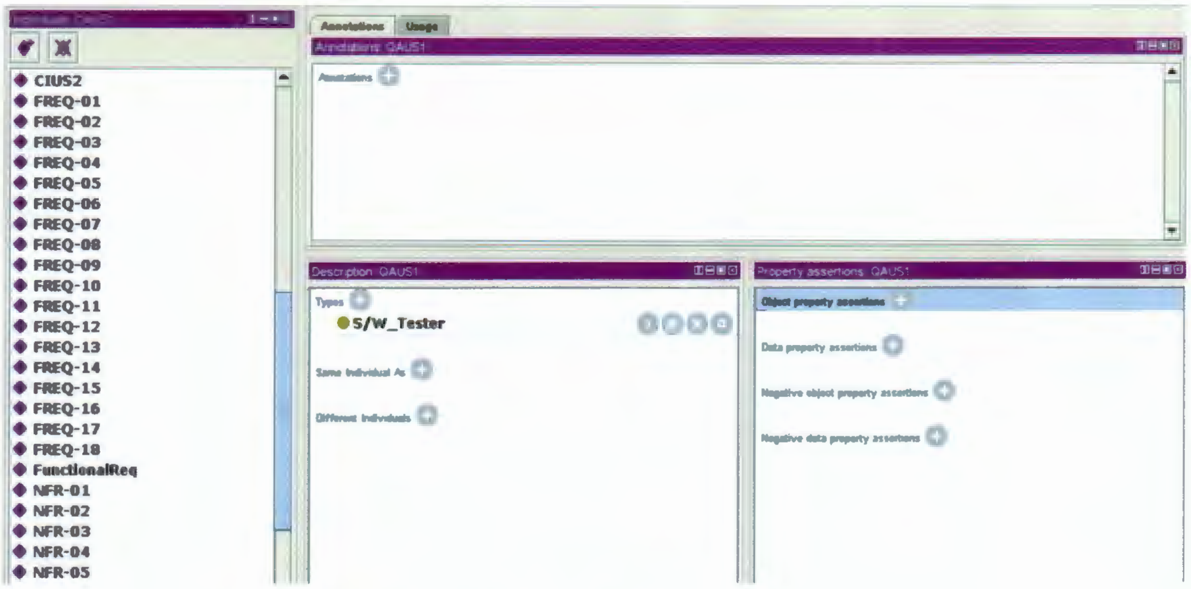


topDataProperty
- TimeZone
- Address
- BReqDesc
- BReqID
- FReqDesc
- FReqID
- NFRDesc
- NFRID
- Project_Manager_Name
- Project_Name
- TCID
- UCID
- Version

Figure 10: Data Properties of RM-GSD Ontology

3.8.4 Define the Instances (Individuals):

Individuals are specific instances (members) of a class. For example, FREQ-01 is the instance of a class "Functional Requirement". Following figure illustrates the instances of RM-GSD Ontology.



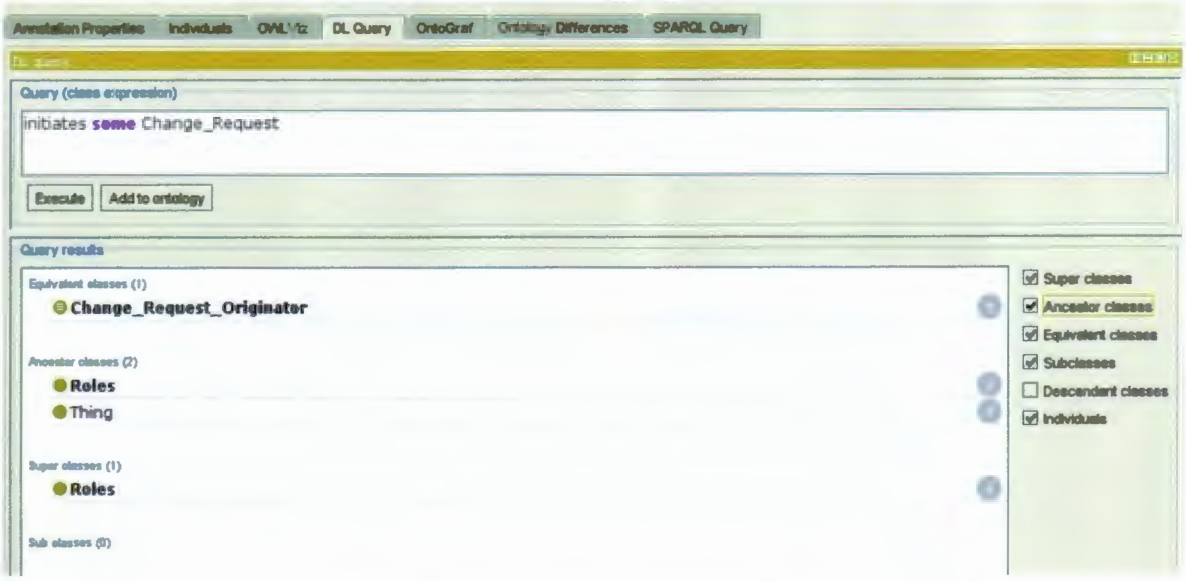
CIUS2
FREQ-01
FREQ-02
FREQ-03
FREQ-04
FREQ-05
FREQ-06
FREQ-07
FREQ-08
FREQ-09
FREQ-10
FREQ-11
FREQ-12
FREQ-13
FREQ-14
FREQ-15
FREQ-16
FREQ-17
FREQ-18
FunctionalReq
NFR-01
NFR-02
NFR-03
NFR-04
NFR-05

S/W_Tester

Figure 11: Instances of RM-GSD Ontology

3.8.5 Axioms of Ontology:

Axioms are assertions that also have rules in a logical form. They contain the information described in the ontology for a specific domain. For example, following axiom is probed in DL query tab: "initiates some Change_Request", in which "initiates" is an object property and "Change_Request" is a subclass of Ӑctivities" that gives the output shown in following figure:



Query (class expression)

initiates some Change_Request

Query results

Equivalent classes (1)
Change_Request_Originator

Ancestor classes (2)
Roles
Thing

Super classes (1)
Roles

Sub classes (0)

☑ Super classes
☑ Ancestor classes
☑ Equivalent classes
☑ Subclasses
☐ Descendant classes
☑ Individuals

Figure 12: Adding Axiom to Change Request Class

Besides, following axioms (rules) can be applied in RM-GSD Ontology.

1. Team Members must exist in min 2 Sites (Due to GSD Setting)
2. Requirement Engineer generates Requirement Traceability Matrix (RTM)
3. Change Request Originator initiates change
4. Change request will be submitted through change request form (CRF)
5. Test Cases tests Requirements
6. GSD Challenges (Communication, Coordination, Control, Knowledge Management) affect GSD Project
7. Change request affect Artifacts.
8. Artifacts affect modules.

Thus from the last two axioms we may infer another rule:

9. Change Request affect modules.

3.9. SELECTION OF THE CASE STUDY

The organization was worked in GSD setting, the software organization's name is not cited due to company privacy policy. The software company has succeeded to complete numerous offshore development projects. This software project is developed at two different sites (Pakistan & U.S (Virginia)). Hence we divided work tasks between distant members on both positions. We have selected E-Health Care Diagnosis System (EHDS) that comprises many more modules discussed ahead. Thus, two modules of user management and doctor management have been chosen to work for the evaluation of our research, also examined it in point of interest to make sense of requirement management process.

3.9.1 A Case Study: E-Health Care Diagnosis System (EHDS)

E-Health Care Diagnosis System (EHDS) was an on-line health care program that gave a complete medical management for the health care professionals and patients. The main purpose of this project is to facilitate the patients by providing easy and fast access for efficient diagnosis and their treatment. In order to facilitate the patients, the diagnose is done on the basis of symptoms and provide treatment according to disease. EHDS system maintains the complete record of patients and doctors, in fact it is a complete package for online health related issues and their remedies. EHDS system had different modules which work successively to complete their activities. Some of them modules have discussed below.

3.9.1.1 User Management Module:

This module is the first module of the system that is used to perform complete registration process of the patients and users. After registration, patients spoke about their symptoms to doctors and then disease is diagnosed for further treatment. This is a pre-requisite of their system to getting health service from them.

3.9.1.2 Doctor Management Module:

This module keeps the information about doctors and every doctor is allowed to make his own profile that holds a doctor's name, email, Specialty, license number, Board, gender, cell number, province, city, country and picture. The Doctor can make his profile after sign in process Further, every doctor's profile is visible to all users. And doctors placed their articles about the diseases on the forums/Stories.

3.9.1.3 Patient Management Module:

This module helps the patient with a diagnosis according to the symptoms. After successful diagnosis patient is asked to get treatment by sending email to doctor. Besides, this module asks for creating a patient profile that maintains the patient's log which can be viewed by system admin and patients themselves. This module is used to update and delete the profile of any patient.

3.9.1.4 Disease Management Module:

This module is used by doctors and admin system, in which doctors add information about new diseases. Further, doctor update/modify the information of existing diseases and delete the information of any disease.

3.9.1.5 Blood Bank Module:

This module is made for those people who want to donate blood to needy patients by creating their profiles. Blood recipients used to check blood donor's profile by locating their address through Google Map that is integrated in the website. It also deals with the donor's record, request for blood donation, available donors; all relevant information is handled in this module.

3.9.1.6 Maps Management Module:

This module was designed for handling donor's address map, by using this users or receivers easily locate the donor address. Admin or donor are able to update or add donor's address map.

3.9.1.7 Language Integration Module:

This is an important module of EHDS system because it facilitates user the most, as by using this function user would be able to change the language of the website. Website language can be switch to default language such as US English.

3.9.1.8 E- Mail Checkup Module:

This is another important module in which patients email their health problems to doctor and doctors reply to their queries and suggest some remedy to treat them. Besides, it shows the profiles of all doctors so patients can choose the physician according to their own choice.

3.9.2 PROBLEM STATEMENT

E-Health Care Diagnosis System (EHDS) was an extensive venture and modules of the system were created at two diverse locales (Pakistan & U.S (Virginia)). According to the need, sometimes a single module was divided among team members at diverse locales.

Both User management and Doctor management module consists of sub modules that were assigned teams of Pakistan and U.S. Sub modules of user management are following: Login User, Update Account, Change Password, Sign Up, Emergency Login, Delete Account, Recover Password, and View all Users. The Doctor management module also comprises sub modules named as Add Doctor and View Doctor. The distribution of the both modules has done at both locations, Module of user management is assigned to team Pakistan and module of doctor management was undertaken by team U.S. It is the obligation of undertaking project manager to track the project and project manager is assigned additional role of change manager in our RM-GSD ontology. Team leads were selected from both teams in order to get any kind of help and assistance of software developers and report back to project/change manager. Requirement engineer is selected to generate a requirement traceability matrix (RTM) on the basis of elicited requirements that shows the relationship between requirements, test cases and any other design artifacts. The role of change implementer was assigned to software engineer or software developer so they implemented the accepted change. Now verification of the change was performed by quality assurance team for figuring out the defects and errors that can disturb the other system components. After ensuring the functionality and worth of implementing change, it became the constituent of the software system.

As discussed in previous chapter, change is the most important element of the software project that keeps a project alive and change can come at any time throughout the software development process, therefore the modules developing in Pakistan and U.S can be demand for change. E-Health care professionals surveyed the other online health related websites and they made changes to their requirements as well, eventually change occurred.

3.9.3 Conceptualization: Identify the Instances

From the above case study, we have separated the instances. Requirement management process was mapped with requirement's change in the module of doctor management of the E-Health Care Diagnosis System (EHDS) in GSD environment (shown in Table # 10).

By using table 10 Instances of doctor management module for Health Care Diagnosis System (EHDS) were taken out that are given in table 11. Subsequently, RM-GSD Ontology (Ontology for requirement management in GSD setting) was developed and for the evaluation/validation purpose, we implemented our ontology on the doctor management module for Health Care Diagnosis System (EHDS). We identified the instances from table 10 to implement them in RM-GSD Ontology. Table 11 demonstrates the Concepts (Classes) and their instances (Individuals). Due to company privacy, names of distant team members are not quoted as we labeled them names abbreviations according to their roles and location for making a difference.

Requirement Management Process	**Requirement Management Process in E-Health Care Diagnosis System (EHDS)**
Requirement engineer generates a requirement traceability matrix on elicited requirements.	In (EHDS), Requirement engineer from Pakistan generates requirement traceability matrix (RTM) that shows the dependency and assess the impact of change that can occur in future.
Change request originator initiated change request.	System admin requested for change and this request will be received by change manager in Pakistan.
Change request is submitted by change request form (CRF)	System admin has used change request form (CRF)
Now change manager make a calendar plan and inform all members at both locations by using asynchronous media.	Change manager taken the updated (CRF) and scheduled the time for change request. Besides, change manager has coordinated with the team leads of Team Pakistan and Team U.S.
Change implementer implements the change by the consensus of CCB (change control board), project/change manager and Change implementer.	Updated change request form that shows the change in any sub module of doctor management is all set for evaluation. The updated (CRF) is sent to the team leads of Pakistan or U.S. Asynchronous communication mode is used for forwarding updated (CRF). And concerned change implementer will implement the change.
After change implementation, verification has to take on board.	Team of quality assurance in Pakistan and U.S. verified the implemented change from their positions.
After verification, the verified change report is combined in change request history/log with the stamp of "Successfully Done". In the meantime Change originator validated the updated change.	Verified change report is combined in change request log and then validated by the system admin.
In the last step, validated change will be incorporated to the system.	Lastly, all the sub modules were integrated to form doctor management module.

Table 10: Mapping requirement management process in GSD environment with E-Health Care Diagnosis System (EHDS) case study

Concepts (Classes)	**Instances (Individuals)**
User	System Admin
GSD Project	E-Health Care Diagnosis System (EHDS)
GSD sub Modules	User Management Doctor Management
Site/Location	Pakistan & U.S (Virginia)
Synchronous Communication Mode	Skype or Video Conference
Asynchronous Communication Mode	Email or Instant Messaging (IM)
GSD Motivations	Skilled professionals, latest technology, less development cost
GSD Factors	Geographical distribution, Time variation, Language and cultural barriers
GSD Challenges	Communication, Coordination & Control
Requirement Engineer	R.E1
Requirement traceability matrix (RTM)	RTM1
Change Request Form (CRF)	CR1
Change Manager/Project Manager (PM has an additional role of CM)	PCM1
Change Implementer	CIP1, CIP2, CIP3, CIP4, CIP5, CIP6 CIUS1, CIUS2
Members from Quality Assurance	QAP1, QAP2, QAP3, QAP4 QAUS1, QAUS2, QAUS3, QAUS4

Table 11: Instances from the E-Health Care Diagnosis System (EHDS) case study

Chapter 4

RESULTS AND DISCUSSION

In this chapter we have evaluated the developed ontology by taking a case study and built a requirement management model introduced and explained in the previous chapter. We built up the application ontology which performed as a knowledge repository. The results were compiled by measuring the effectiveness of ontology to determine that knowledge repository has been made then the knowledge stored will be useful for the members involved in the project for managing requirement change along impact analysis in Global Software Development Setting.

4.1 EVALUATION:

We have executed the developed requirement management model. A questionnaire has also designed and domain experts were the judges.

The worth of our RM-GSD Ontology has been assessed in two stages. Initially, generic ontology of requirement management for Global Software Development has been assessed while in second stage, specific case study of doctor management module (EHDS) instances has been represented to the non-specific ontology and validated the developed ontology with a specific end goal by answering competency questions.

4.1.1 Evaluation & Validation of RM-GSD Generic Ontology by Domain Experts

Domain experts have evaluated our developed ontology named as (RM-GSD Ontology). They assessed the developed ontology by validating its consistency and correctness. The choice of the domain experts is made on the premise of their work experience in GSD as they are aware of issues that arise in GSD settings. They used the Ontology development tool Protégé by selecting a reasoner HermiT 1.3.7 to verify consistency and correctness in an ontology. Domain specialists approved the conceptual model of RM-GSD theoretically and co-ordinated the inferred hierarchy with it by demonstrating the correctness in the developed ontology.

4.1.2 Use of Reasoner in RM-GSD Generic Ontology

Protégé tool is chosen for ontology development that also facilitates in automatic reasoning. It has some different kinds of reasoners. These are HermiT, FACT++ and Pellet. But we have opted HermiT 1.3.7 reasoner for consistency checking. When reasoner was started, then it infers the axioms for showing the inferred hierarchy. Protégé has a tab of OwlViz that converts the concepts and their attributes in a meaningful visualization. Ontology is tested by specialists for investigating the complete taxonomy of concepts. Indeed, even a few concepts are compared with the conceptual model keeping in mind the end goal to ensure consistency and correctness among concepts.

Reasoner is a sort of compiler that checks the consistency and in case of any inconsistency it highlights those concepts that don't match the consistency criteria of the reasoner. Competency questions are answered by showing the right and desired results that measure the calibre of ontology. The tool gives the choice of stop reasoner with start reasoner and reasoner type options, and inferred model shows the concept "Nothing" if reasoner was stopped. Following figures proves that the generic RM-GSD ontology has successfully worked. Our ontology is on its preliminary stage therefore we have applied some rules for taxonomy of concepts and consistency checking.

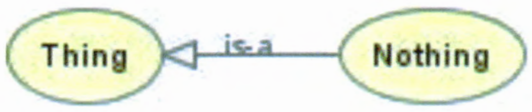


Figure 13: OwlViz Representation of Nothing Class

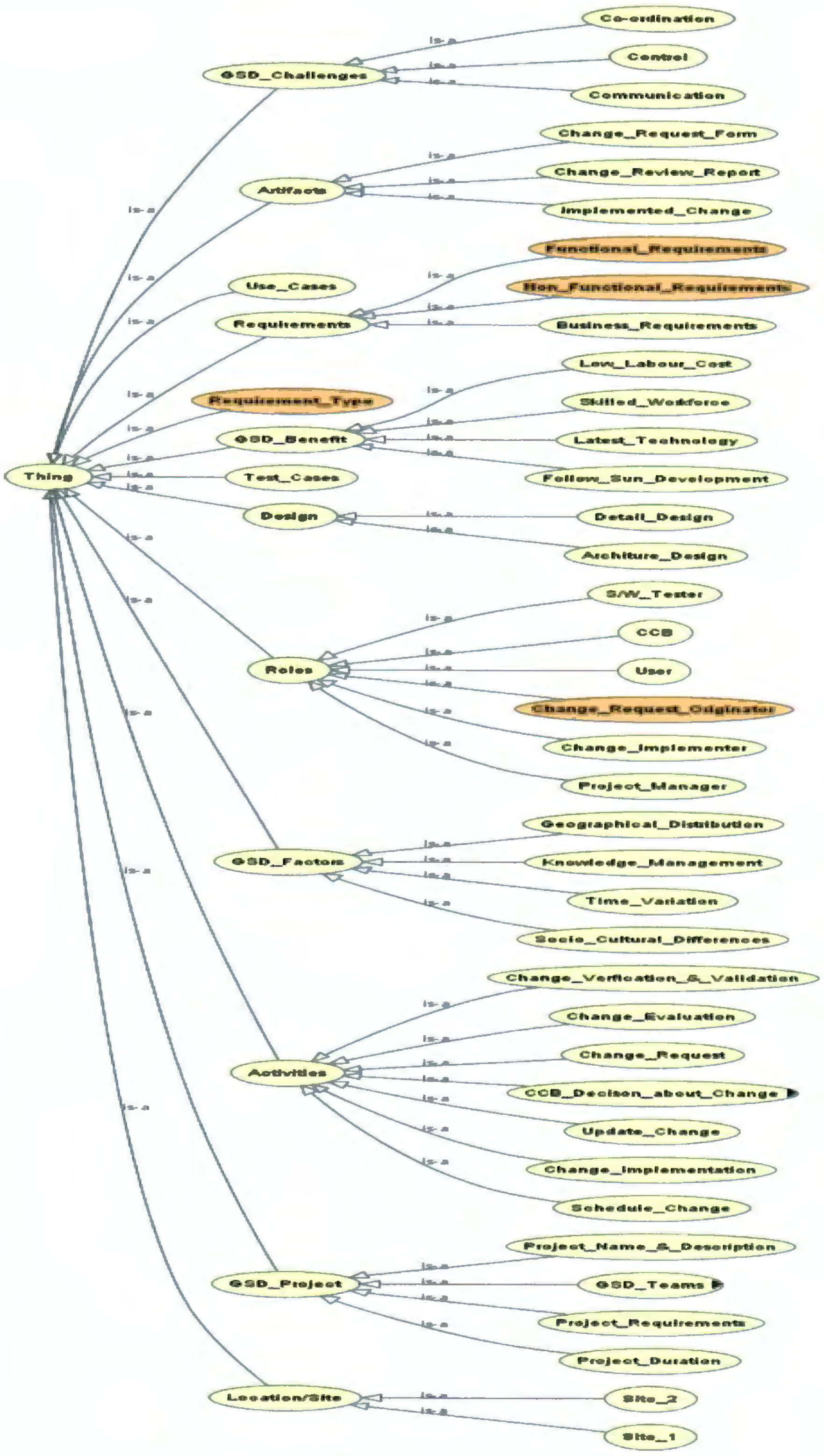


Figure 14: OwlViz Representation of RM-GSD Ontology

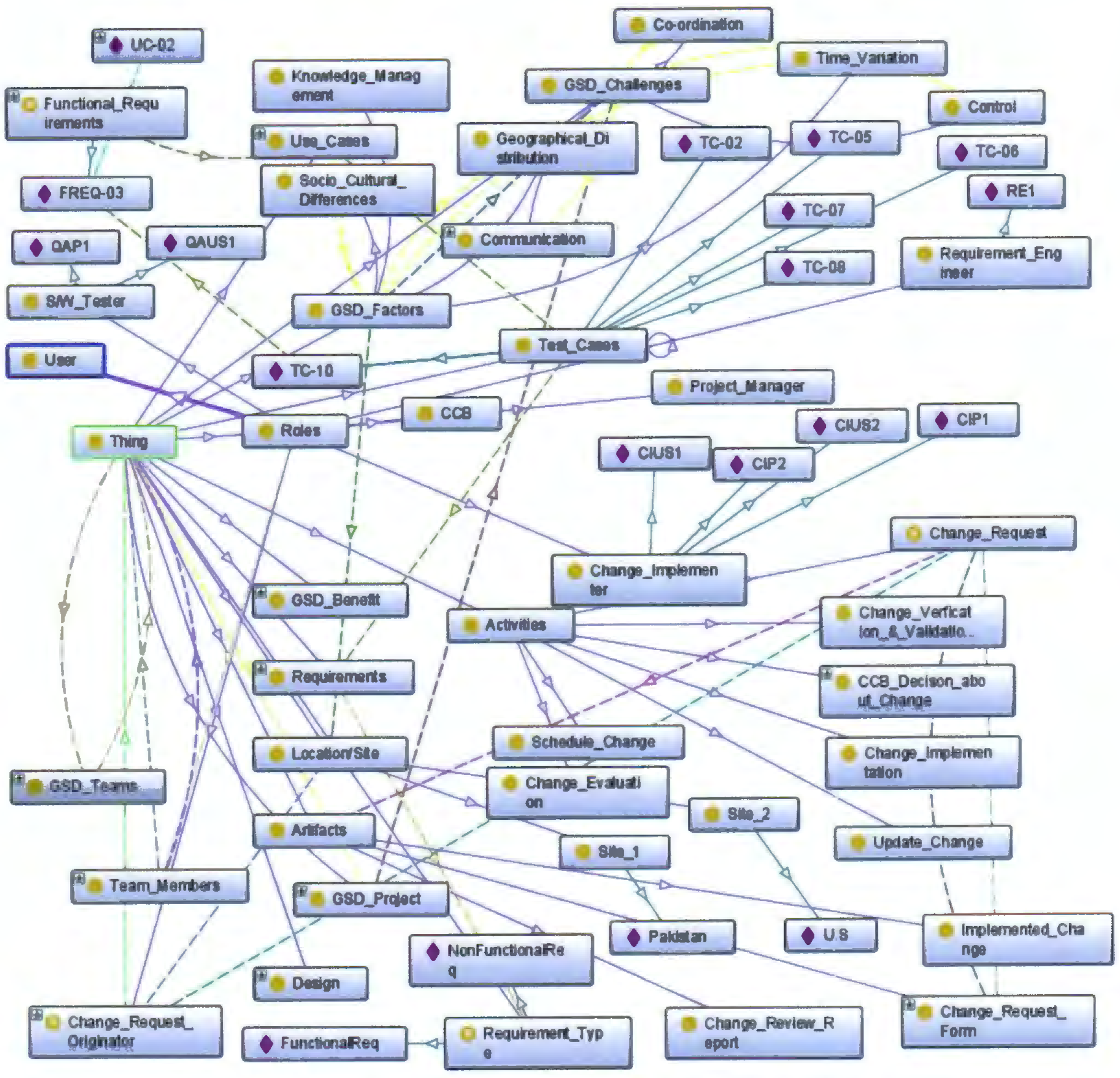


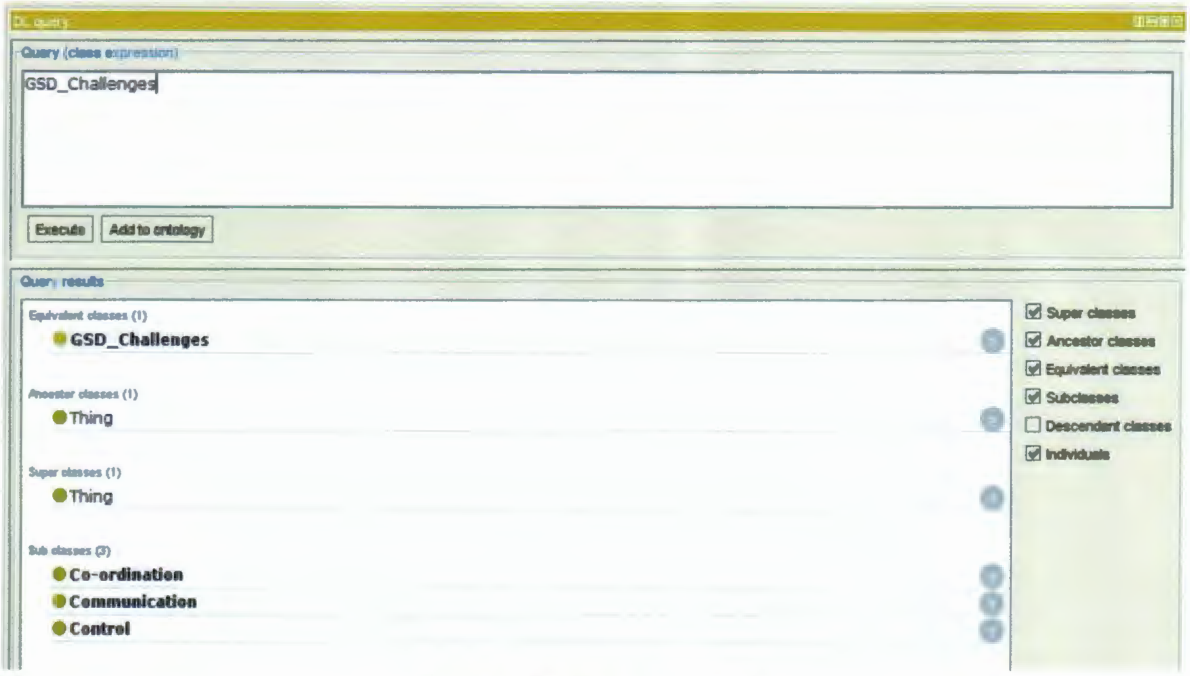
Figure 15: Onto Graf Visualization of RM-GSD Ontology

4.2 Evaluation and Validation of RM-GSD Ontology with a case study E-Health Care Diagnosis System (EHDS)

The developed ontology has been validated with the help of a case study that manifest the system functionality and efficacy of Requirement Management for Global Software Development (RM-GSD) Ontology and additionally the users of ontology are contented to see their endeavours completely operationalized with the feature of visualization. The results obtained after the implementation of Requirement Management for Global Software Development (RM-GSD) Ontology in a software house located in Islamabad. The experts of that house taken up the ontology if there should be an occurrence of change emerging in any requirement. We have already been assessed the generic (non-specific) ontology by giving responses to competency questions and specific ontology of (EHDS) doctor management module also fulfilled the competency questions. The experts handled only one request of change that is requested by the client. One development team and quality assurance team positions in U.S(Virginia) while change manager, development team and client resides in Pakistan (Islamabad). We have extracted all the individuals or instances and mapped them to the generic (non-specific) ontology and ensured that the developed Requirement Management for Global Software Development (RM-GSD) Ontology functioned admirably with the instances.

4.2.1 Use of Reasoner in E-Health Care Diagnosis System (EHDS)

We reaffirm our ontology by making a single change in requirements of EHDS, from the case study that has been discussed in the former chapter. We mapped out those instances that were parted in the previous chapter and inferred the results for ensuring that ontology is working according to our aspirations. The participants of the GSD setting have used the ontology on the basis of rules and they utilized the ontology as a part of one module of the E-Health Care Diagnosis System (EHDS).


Query (class expression)

GSD_Challenges

Query results

Equivalent classes (1)
● GSD_Challenges

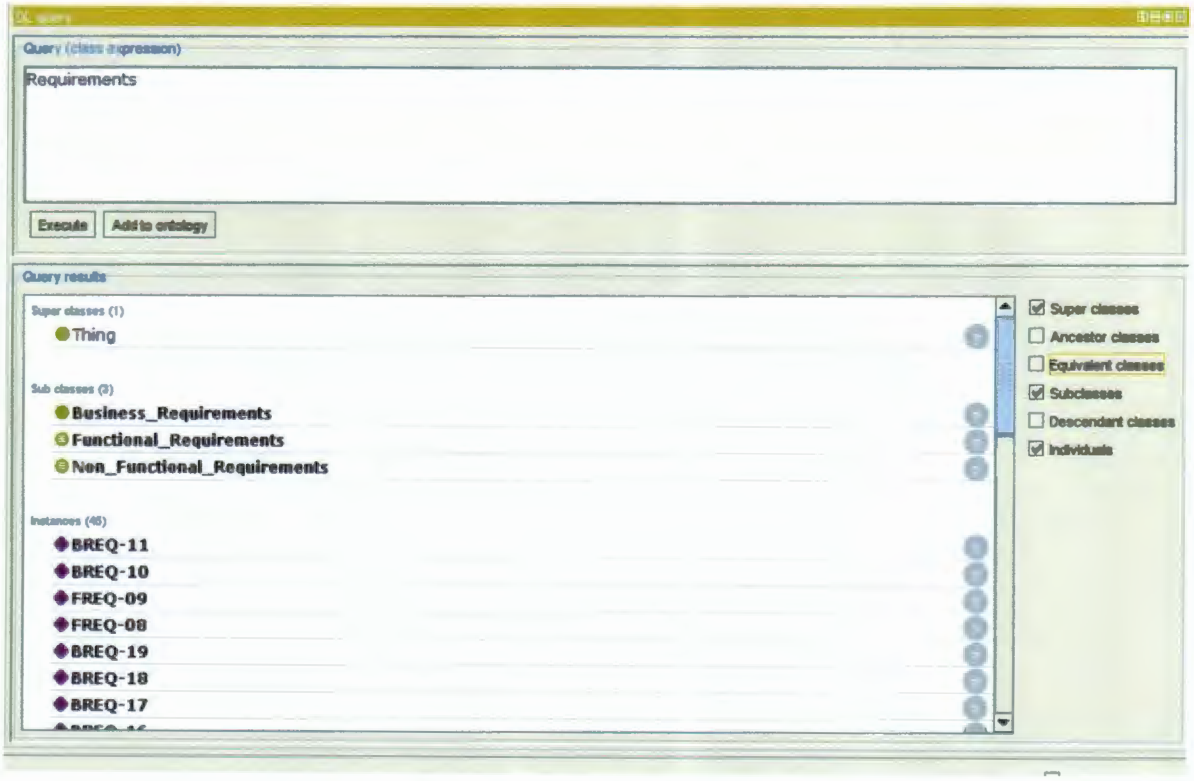
Ancestor classes (1)
● Thing

Super classes (1)
● Thing

Sub classes (3)
● Co-ordination
● Communication
● Control

[✓] Super classes
[✓] Ancestor classes
[✓] Equivalent classes
[✓] Subclasses
[ ] Descendant classes
[✓] Individuals

Figure 16: Results in DL Query1 Tab


Query (class expression)

Requirements

Query results

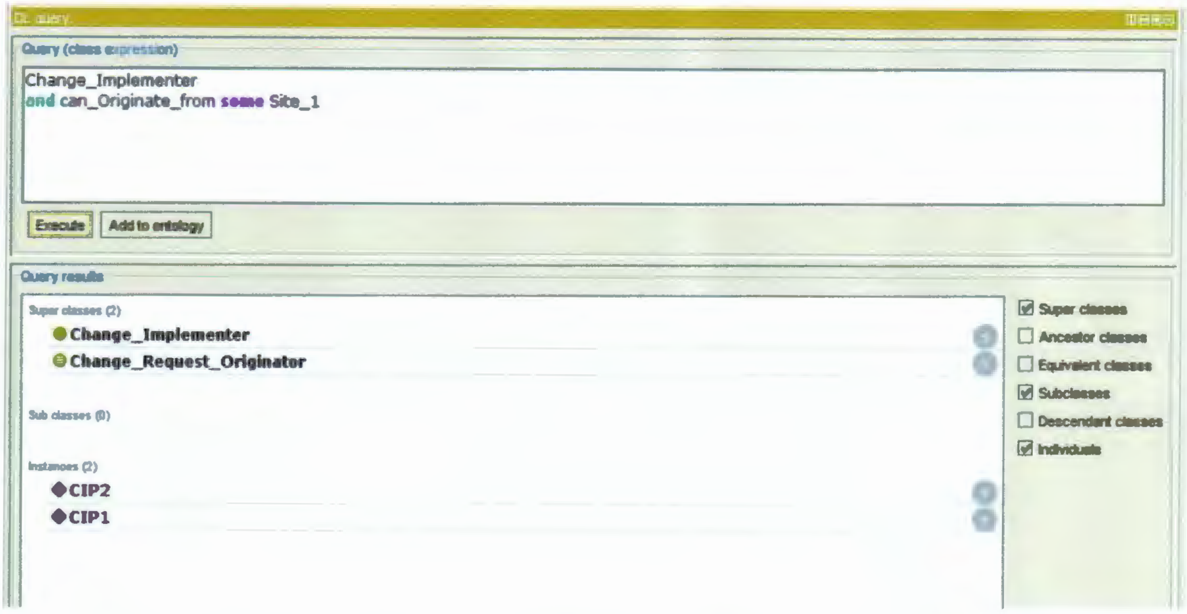
Super classes (1)
● Thing

Sub classes (3)
● Business_Requirements
● Functional_Requirements
● Non_Functional_Requirements

Instances (46)
◆ BREQ-11
◆ BREQ-10
◆ FREQ-09
◆ FREQ-08
◆ BREQ-19
◆ BREQ-18
◆ BREQ-17

[✓] Super classes
[ ] Ancestor classes
[ ] Equivalent classes
[✓] Subclasses
[ ] Descendant classes
[✓] Individuals

Figure 17: Results in DL Query2 Tab


DL query

Query (class expression)

```
Change_Implementer
and can_Originate_from some Site_1
```

Query results

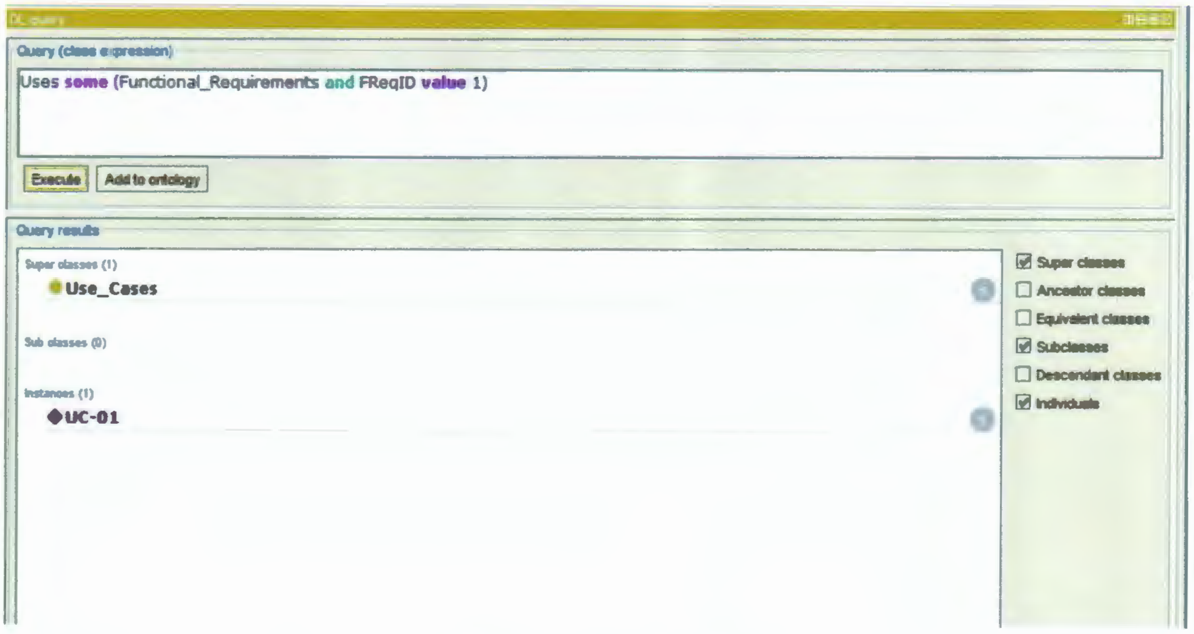
Super classes (2)
- Change_Implementer
- Change_Request_Originator

Sub classes (0)

Instances (2)
- CIP2
- CIP1

[✓] Super classes
[ ] Ancestor classes
[ ] Equivalent classes
[✓] Subclasses
[ ] Descendant classes
[✓] Individuals

Figure 18: Results in DL Query3 Tab


DL query

Query (class expression)

```
Uses some (Functional_Requirements and FReqID value 1)
```

Query results

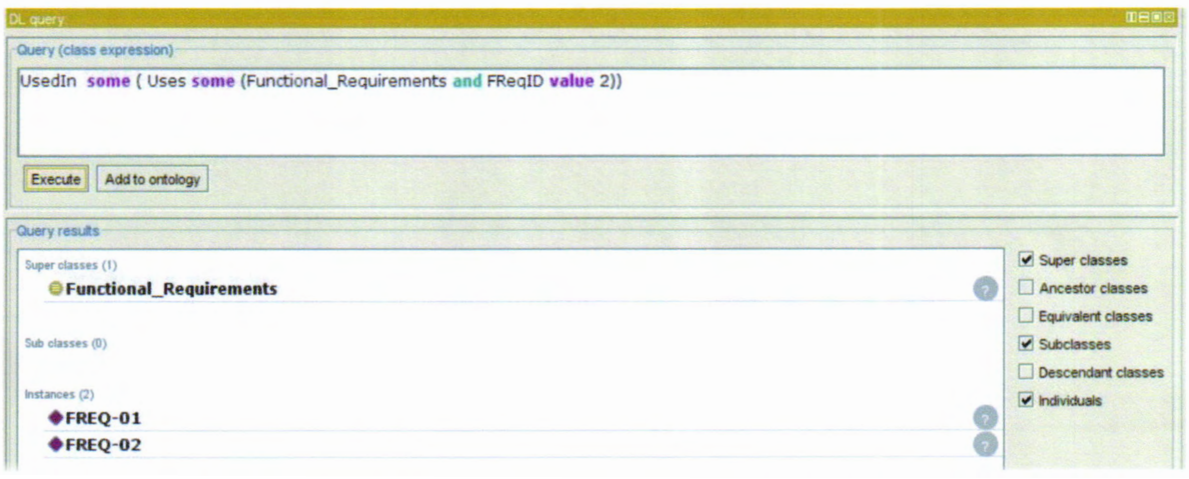
Super classes (1)
- Use_Cases

Sub classes (0)

Instances (1)
- UC-01

[✓] Super classes
[ ] Ancestor classes
[ ] Equivalent classes
[✓] Subclasses
[ ] Descendant classes
[✓] Individuals

Figure 19:Results in DL Query4 Tab


DL query

Query (class expression)

```
UsedIn some ( Uses some (Functional_Requirements and FReqID value 2))
```

Query results

Super classes (1)
- Functional_Requirements

Sub classes (0)

Instances (2)
- FREQ-01
- FREQ-02

- [✓] Super classes
- [ ] Ancestor classes
- [ ] Equivalent classes
- [✓] Subclasses
- [ ] Descendant classes
- [✓] Individuals

Figure 20: Results in DL Query5 Tab

4.2.2 Questionnaires Developed for the Evaluation of our Approach

We have designed a questionnaire for the evaluation of our requirement management approach and the questionnaire is placed as under:

Serial #	QUESTIONS	Strongly Agree (8-10)	Partially Agree (4-7)	Not Agree (0-3)
1.	Does the proposed requirement management approach make stronger the activities of R.E?			
2.	Does the implemented requirement management approach achieve consensus and resolving conflicts among the distant stakeholders?			
3.	Does the proposed requirement management approach provide completeness, consistency and correctness?			
4.	Can our proposed solution reduce the project development time?			
5.	Does the developed Interface of the tool is user friendly?			
6.	Do you think that our proposed approach improves the communication structure?			
7.	Does the provided solution enhance the precision and accuracy in the domain of interest?			
8.	Can our proposed approach achieve customer satisfaction?			

Table 12: Questionnaire for the Evaluation of our Proposed RM Approach

4.3 EVALUATION BY DOMAIN EXPERTS

Our proposed requirement management approach is evaluated by the experts (Members of GSD teams), who used the ontology of E-Health Care Diagnosis System (EHDS). We requested them for knowing the value and benefits of our RM-GSD Ontology and developed knowledge base of EHDS. The questionnaires put forth to the domain experts for their views. Experts are selected for review from both locales such as change manager and team lead CIP1from Pakistan, change implementer and team lead CIUS1 from U.S, one requirement engineer, three members from quality assurance team, and two members from CCB (change control board). Our proposed approach is evaluated by some given options such as Strongly Agreed, Partially Agreed and Not Agreed by ratings (0-3, 4-7, 8-10) separately.

First of all, experts checked the concepts taxonomy for assuring correctness and then they moved towards practical ontology for testing consistency among the classes. Lastly, we have asked them to rate the ontology on the base of some parameters that can be helpful in assuring the validity of our ontology. Following parameters were investigated from the experts e.g. Knowledge Management, Shared concepts & Resolve Conflicts, Consistency & Completeness, Customer Satisfaction, Easier to use, Impact on Cost, Improve Communication and Reduce development Time. Moreover, the mean is applied to all the accompanying values of each parameter for obtaining the result, and presented the solution as shown in following picture.

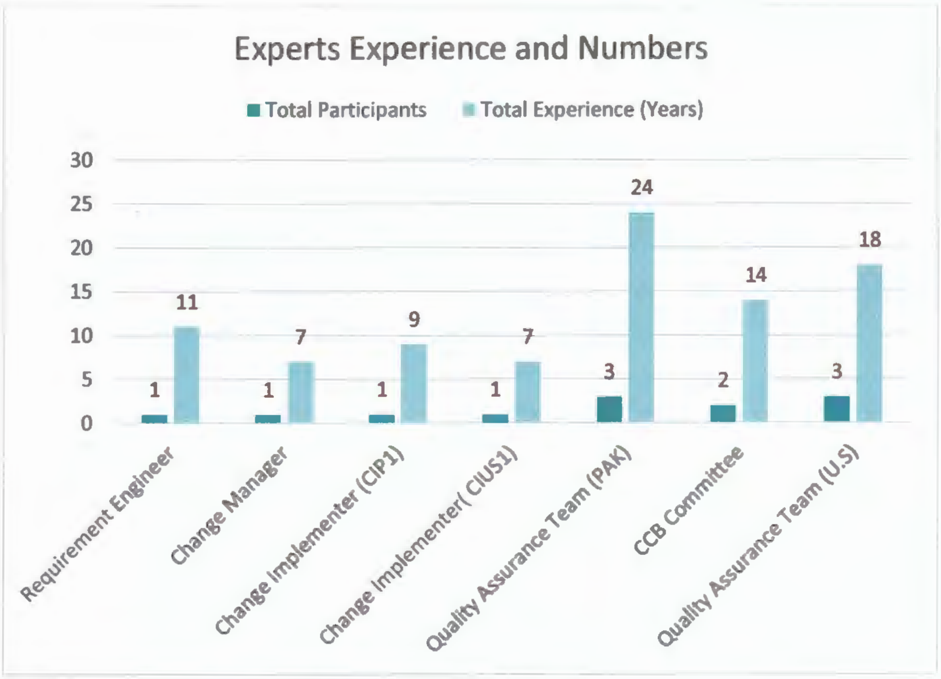


Figure 21: Total participants with Experience (years)

Following picture shows the rating scale between (0-10) by the software specialists.

Parameters	Requirement Engineer	Change Manager	Change Implementer (CIP1)	Change Implementer (CIUS1)	Quality Assurance Team (PAK)	CCB Committee	Quality Assurance Team (U.S)	CCB Committee
Knowledge Management	Y	Y	Y	Y	Y	PA	Y	PA
Shared concepts & Resolve Conflicts	Y	Y	Y	Y	Y	Y	Y	Y
Consistency & Completeness	Y	Y	Y	Y	Y	PA	Y	PA
Customer Satisfaction	PA	Y	Y	Y	Y	PA	Y	PA
Ease of Use	NA	PA	Y	Y	Y	NA	Y	NA
Impact on Cost	Y	Y	PA	PA	NA	Y	NA	Y
Improve Communication	Y	Y	Y	Y	Y	Y	Y	Y
Reduce development Time	PA	PA	Y	Y	PA	NA	PA	NA

Y=Strongly Agreed PA=Partially Agreed NA=Not Agreed

Table 13: Domain Experts Review

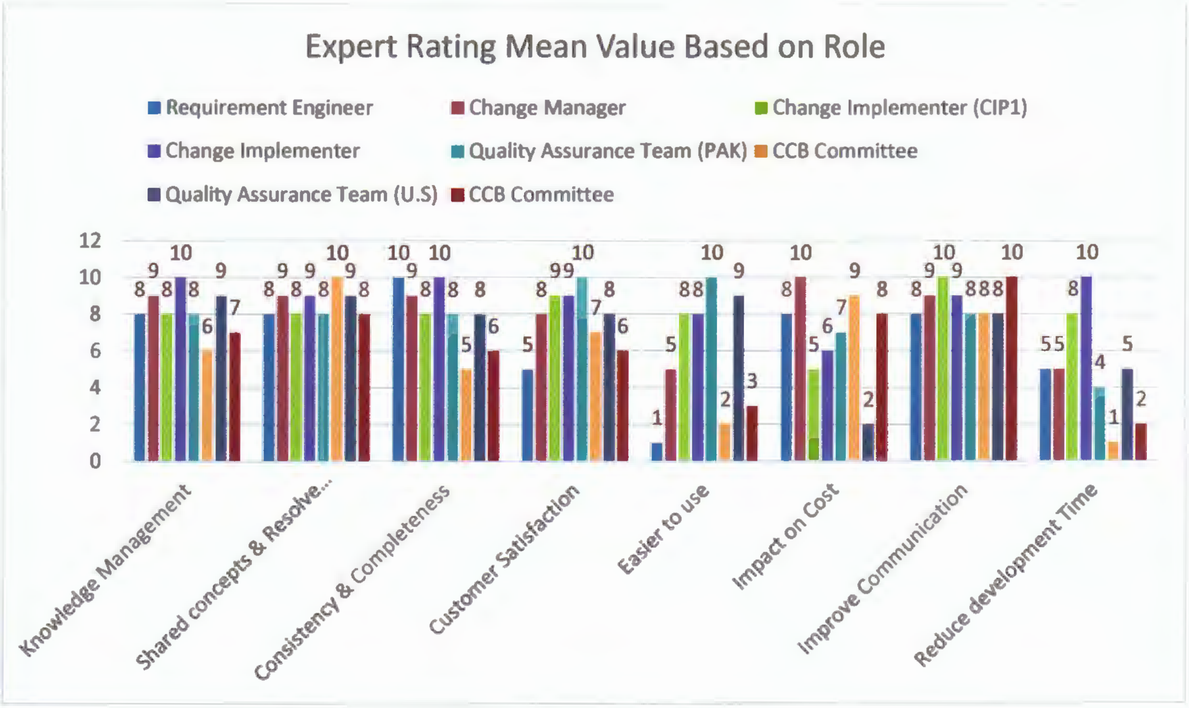


Figure 22: Expert Rating Mean Value Bar Graph

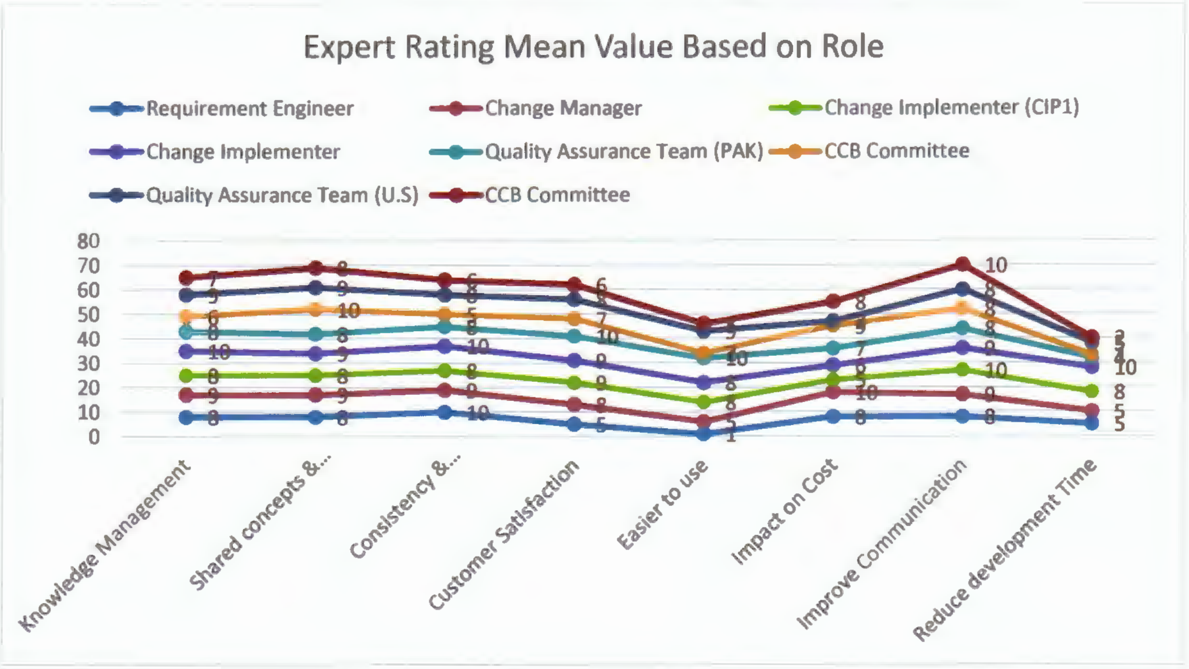


Figure 23: Graphical Representation of Experts Opinion

In the following section, the reasons are explained in the favor or against of the ontology based approach and we also evaluated the reasons for ontology approval or ontology refusal.

4.3.1 Knowledge Management

In Global Software Development environment, teams are located in different places and face communication issues. As we have discussed earlier that knowledge management can minimize the communication issues. Hence, we presented ontology as a solution that is used to store and manage the information/knowledge of a specific domain.

4.3.2 Shared Concepts & Resolve Conflicts

An ontology based repository comprises on agreed upon vocabulary that lessen the conflicts among distant members, since ontology can express the hierarchy of concepts and their instances so it can make simpler for stakeholders to eliminate the conflict of information.

4.3.3 Consistency & Completeness

Consistency indicates about the knowledge contribution must not break any rule. Completeness checks whether all fundamental meet the present and future business data interest are presented in the information resource. In this thesis, ontology based requirement management approach was developed in ontology development tool Protégé that have multiple reasoners (FACT++, Pellet, HermiT) for checking consistency and completeness.

4.3.4 Customer Satisfaction

The parameter identifies the level of customer satisfaction whether accomplished or not by using ontology. The inference rules in ontology gives proper understanding about the domain that raises the customer satisfaction level as well.

4.3.5 Ease of Use

According to experts Opinion, the interface of our ontology is quite simple and easy to understand, though it demands technical expertise in the query process. Change implementers and members of the quality assurance team (Pakistan & U.S), were strongly agreed about the simple use of ontology especially at the level of super classes and sub classes.

4.3.6 Impact on Cost

Software companies operate in an unforeseen atmosphere as anything can be evolve or change for the software system e.g. Government laws, company policy, technology setup, etc. all have some impact regarding to the situation. We have built up an evolving ontology that can be affected by the new laws or patterns of the software company, Hence, it can affect the cost to a certain extent.

4.3.7 Improve Communication

It is evident from the views of experts that ontology based approach can improve the communication among the distant team members.

4.3.8 Reduce development Time

If development time can be reduced by using ontology, then it will definitely have an impact on cost and budget. The underlying development time of ontology is more than traditional development in view of expert opinions.

In the above part, we have examined the above clarification about the parameters and condensed the level of acknowledgment or dismissal of ontology, and figured result in tabular and also in graphical structure.

Chapter 5

CONCLUSION

This thesis presents an ontology-based requirements management approach for the globally distributed software development projects. Requirements management is a vital activity that continues throughout the software development life cycle (SDLC) therefore the success rate of the distributed projects affects the most because of the requirements management (RM) process. Previously, numerous frameworks have been presented to handle the software requirement's change in Global Software Development (GSD) environment, though requirement traceability (RT) has been used less with requirements change management (RCM) process in order to analyze the impact of change.

In this study, we have developed an ontology based process for the activity of requirements management for Global Software Development set-up. The main objective was to develop a solution made for managing requirement change and improve the communication mechanism between distant GSD group and the developed ontology will facilitate in knowledge sharing and knowledge management. Ontologies are constantly evaluated by individuals, for which our ontology has been tested by a group of people, they have assessed the ontology theoretically and by applying it in a genuine case study. We used a case study from a vast project for validation, yet it was confined to the single module to lessen the intricacy. The feedback from experts shows that our ontology based requirement management approach offers better benefits. Indeed, it's suitable for handling change, resolving conflicts and confusions, sharing and managing knowledge as well as it establishes an appropriate communication mechanism that is a crux of Global Software Development (GSD) projects.

In this research, the results have fetched from the small ontology, but if already engineered ontology of any specific domain is combined and then the requirements and user's knowledge are included in the same vocabulary, it will require less energy to perform the task with multiple advantages.

5.1 Recommendations for Future Study

In the future work it is recommended that the approach should be extended to the activity of requirements development for the global software development projects. Another ontology can be developed according to the format of our approach, that covers the requirements development activity, afterwards it could be merged to our provided framework thus the new ontology will resolve the issues of both activities of Requirements Engineering.

LITERATURE CITED

1. Kotonya, G. and I, Sommerville 1998 Requirements engineering – processes and techniques, John Wiley & Sons UK.
2. Sommerville, I, and P. Sawyer. 1997. Requirements Engineering: A Good Practice Guide. Chichester, UK: John Wiley & Sons.
3. Farfeleder, S., T. Moser, A. Krall, I. Omoronyia, and H. Zojer. 2011. Ontology-Driven Guidance for Requirements Elicitation. ESWC:212–226.
4. Herbsleb, J. D., & Moitra, "Global software development" Software, IEEE, 2002.
5. Nasir Mehmood Minhas, Qurat-ul-Ain, Zafar-ul-Islam, Atika Zulfiqar, "An Improved Framework for Requirement Change Management in Global Software Development", Journal of Software Engineering and Applications, Published Online in Sci-Res August 2014.
6. Eoin O Conchuir, Par J. Agerfalk, Helena Olsson, and Brian Fitzgerald, "Global software development: where are the benefits?", Communications of the ACM - A Blind Person's Interaction with Technology, Volume 52 Issue 8, August 2009.
7. Naveed Ali and Richard Lai, "A method of requirements change management for global software development", Information and Software Technology, Elsevier, 2016.
8. Daniela Damian, "Stakeholders in Global Requirements Engineering: Lessons Learned from Practice", IEEE Software Engineering Journal, 2007.
9. Siegemund, K., E. J. Thomas, Y. Zhao, and J. Pan. 2011. Towards Ontology-driven Requirements Engineering:15.
10. N. Nurmuliani, D. Zowghi and S. Fowell "analysis of requirements volatility during software development life cycle", Proceedings of the Australian Software Engineering Conference, Melbourne, Australia, 2004.
11. Saffena Ramzan, Naveed Ikram, "Making Decision in Requirement Change Management", 2005 IEEE.
12. Hall, T., S. Beecham, and A. Rainer, Requirements problems in twelve companies: an empirical analysis. IEEE proceedings software, 149 (5): 153-160. 2002.
13. Richard Lai. And Naveed Ali, "Requirements Management Method for Global Software Development", Advances in Information Sciences (AIS), 2013.

14. Asma Khatoon, Yasir Hafeez, S.Asghar & T. Ali, "An ontological framework for requirement change management in distributed environment", The Nucleus 51, No. 2 (2014).

15. Waqar Hussain and Tony Clear, "GRCM: A Model for Global Requirements Change Management", 2nd International Requirements Engineering Efficiency Workshop held in Essen, Germany, 2012.

16. Richard Lai. And Naveed Ali, "Managing Requirements Change in Global Software Development", IEEE Data and Software Engineering (ICODSE), International Conference in November. 2014.

17. Yasir Hafeez, Muhammad Riaz, Sohail Asghar, Hummera Naz, Syed Mushhad Mustuzhar Gilani, Asma Batool, Mehmood Ahmed, M. Shabbir Hassan, "A Requirement Change Management Framework for Distributed Software Environment", Computing and Convergence Technology (ICCCT), 2012 7th International Conference, December 2012, IEEE.

18. Arif Ali Khan, ShuibBasri, P.D.D.Dominic "A Propose Framework for Requirement Change Management in Global Software Development", 2012 International Conference on Computer & Information Science (ICCIS), 2012 IEEE.

19. Ying Guo, Meihong Yang, Jun Wang, Ping Yang, Feng L, "An Ontology-based Improved Software Requirement Traceability Matrix", 2009 Second International Symposium on Knowledge Acquisition and Modeling, 2009 IEEE.

20. Arif Ali Khan, Jacky Keung, Shahid Hussain and Kwabena Ebo Bennin, "Effects of Geographical, Socio-cultural and Temporal Distances on Communication in Global Software Development during Requirements Change Management: A Pilot Study", ENASE, 10th International Conference on Evaluation of Novel Software Approaches to Software Engineering, July 2015.

21. Kumar, S.A. And Kumar, T.A. "Study the Impact of Requirements Management Characteristics in Global Software Development Project: An Ontology Based Approach". International Journal of Software Engineering and Application, October 2011.

22. Asma Khatoon, Yasir Hafeez Motla, Madiha Azeem, Humera Naz, Sana Nazir, "Requirement Change Management for Global Software Development using Ontology",

Emerging Technologies (ICET), 2013 IEEE 9th International Conference on December 2013.

23. Antunes, B., N. Seco, and P. Gomes, "Using Ontologies for Software Development Knowledge Reuse", in Proc. Of the 13th Portuguese Conference on Artificial Intelligence (EPIA 2007), LNCS, Springer, Vol. 4874, Guimarães, Portugal, December, 2007.
24. Sunitha Abburu, G Suresh Babu, "A Framework for Ontology Based Knowledge Management", International Journal of Soft Computing and Engineering (IJSCE) ISSN: 2231-2307, July 2013.
25. Rodrigo G.C Rocha, Rayan Azevedo, Silvio Meira, "A proposal of an ontology-based system for distributed teams", Software Engineering and Advanced Applications (SEAA), 2014 40th EUROMICRO Conference, Aug 2014.
26. Orlena C. Z. Gotel and Anthony C. W. Finkelstein, "An Analysis of the Requirements Traceability Problem," in Proceedings of 1st International Conference on Requirements Engineering, 1994, pp. 94-101.
27. F. Pinheiro and J. Goguen, "An Object-Oriented Tool for Tracing Requirements," IEEE Software, vol. 13, no. 2, pp. 52-64, March 1996.
28. CMMI for Development, Version 1.2.: Carnegie Mellon Software Engineering Institute, August 2006.
29. R.J. Wieringa, "An Introduction to Requirements Traceability," Faculty of Mathematics and Computer Science, University of Vrije, Amsterdam, September 1995.
30. M. Lindval and K. Sandahl, "Practical Implications of Traceability," Software Practice and Experience, vol. 26, no. 10, pp. 1161-1180, 1996.
31. Damian, D. Dustdar, S. (2005) "International workshop on Distributed Software Development", 13th IEEE Requirement Engineering Conference 2005, 29 August 2005, Paris, France.
32. (Carmel, E., 1999. "Global Software Teams: Collaborating Across Borders and Time Zones", Prentice Hall.
33. Herbsleb, D.J. Mockus, A., 2003. "An Empirical Study of Speed and Communication in Globally-Distributed Software Development", IEEE Trans. Softw. Eng, 29(6), pp. 481-494.

34. Carmel, E. Dubinsky, Y. Espinosa, A., 2009. "Follow the Sun Software Development: New Perspectives, Conceptual Foundation, and Exploratory Field Study", Proceedings of the 42nd Hawaii International Conference on System Sciences.

35. Bennett, K., 1996. Software evolution: past, present and future, Information and software technology, 38:673-680.

36. Emam, K.E. and N.H. Madhavji, 1995. Measuring the success of requirements engineering processes, Proc. 2nd IEEE International Symposium on Requirements Engineering, York, UK. pp.204-211.

37. Kobayashi, M. Maekawa, 2001. Need-Based Requirements Change Management, 8th Annual IEEE International Conference and Workshop on the Engineering of Computer Based Systems (ECBS '01), Washington D.C., 2001, pp.171-178.

38. Boehm BW, PN, Papaccio 1998. Understanding and controlling software costs. IEEE transactions on software engineering, 14 (10): 1462 – 1477.

39. IEEE-STD 610.12-1990, Standard Glossary of Software Engineering Terminology, 1990, Institute of Electrical and Electronics Engineers.

40. Macaulay LA, 1996, Requirements Engineering, Springer-Verlag, New York, London.

41. Martin S, A. Aurum, R. Jeffery, and B. Paech, 2002. Requirements engineering process models in practice. In: Proceedings of 7th Australian workshop on requirements engineering. AWRE02, 2-3 December, Melbourne, pp. 41-47.

42. Boehm BW, 1998. A spiral model of software development and enhancement, Computer, May 21(5): 61-72.

43. Anthony, R. N., 1965. Planning and control systems: a framework for analysis. Harvard University, Boston, USA.

44. Jonsson, P., M. Lindvall, 2005. Impact Analysis, in Engineering and Managing Software Requirements, edited by A. Aurum and C. Wohlin, pp. 117-142, Springer Verlag.

45. Ajila, S.A., 2002. Change Management: Modeling Software Product Lines Evolution, Proc. Of the 6th World Multiconference on Systemics, Cybernetics and Informatics, Orlando, Florida, 2002, pp.492-497.

46. Jonsson, P., 2007. Exploring Process Aspects of Change Impact Analysis. Dissertation Series No 2007:13, Blekinge Institute of Technology.

47. Olsen, N. C., 1993 The Software rush hour, IEEE Software, vol.10 no.5, pp.29-37.

48. Lam, W., V. Shankaraman, S. Jones, J.Hewitt, and C. Britton, 1998. Change Analysis and Management: A Process Model and its Application within a Commercial Setting. Proc. of the IEEE Workshop on Application – Specific Software Engineering and Technology.
49. Harjani, D. R., J. P. Queille, 1992. A process model for the maintenance of large space systems software. Proceedings of Conference on Software Maintenance. Los Alamitos: IEEE Computer press.
50. Asa G.D, P. Anne, 2005. Requirements Interdependencies: State of the Art and Future Challenges, in Engineering and Managing Software Requirements, edited by A. Aurum and C. Wohlin, pp. 95-116, Springer Verlag.
51. Sunitha Abburu, "A Survey on Ontology Reasoners and Comparison", International Journal of Computer Applications (0975 – 8887) Volume 57– No.17, November 2012.
52. Chimaera Ontology Environment , www.ksl.stanford.edu/software/chimaera. 2000.
53. http://www.ksl.stanford.edu/software/ontolingua.
54. Graciela Brusa, Ma. Laura Caliusco, Omar Chiotti, "A Process for Building a Domain Ontology: An Experience in Developing a Government Budgetary Ontology", Australian ontology workshop (AOW 2006) Conference in research and practice in information technology, Hobart, Australia.
55. Filippo Lanubile , "Collaboration in Distributed Software Development" , A. De Lucia and F. Ferrucci (Eds.) ISSSE 2006–2008, pp. 174–193, 2009. © Springer-Verlag Berlin Heidelberg 2009
56. Cresswell, J.W., 2009. Research Design: Qualitative, Quantitative, and Mixed Methods Approaches. 3rd ed. Thousands Oaks, CA. Sage Publications.
57. Miles, M. B. Huberman, M., 1994. Qualitative Data Analysis: an expanded sourcebook, SAGE Publication.
58. Yin R. (2014), Case Study Research: Design and Methods, 5th edition (first edition, 1984), Sage, Los Angeles.
59. Thomas R. Gruber, "A Translation Approach to Portable Ontology Specifications", Appeared in Knowledge Acquisition, 5(2):199-220, KNOWLEDGE SYSTEMS LABORATORY Computer Science Department Stanford University Stanford, California, 1993.

60. Daniela Damian, Didar Zowghi "The impact of stakeholders' geographical distribution on managing requirements in a multi-site organization", Requirements Engineering, Proceedings. IEEE Joint International Conference on Sep. 2002.

61. Catherine Roussey, Francois Pinet, Myoung Ah Kang, and Oscar Corcho, "An Introduction to Ontologies and Ontology Engineering", Ontologies in Urban Development Projects, Volume 1 of the series Advanced Information and Knowledge Processing pp 9-38, June 2011.