Requirement Engineering Challenges in Development of Software Applications and Selection of Customer-off-the-Shelf (COTS) Components

Dr. Sohail Asghar Sohail.Asghar@jinnah.edu.pk
Center of Research in Data Engineering (CORDE)
Mohammad Ali Jinnah University (MAJU)
Islamabad, Pakistan, 44000.

Mahrukh Umar Mahrukhumar@yahoo.com
Department of Computer Science,
Shaheed Zulfikar Ali Institute of Science and Technology (SZABIST),
Islamabad, 44000,Pakistan.

Abstract

Requirement Engineering acts as foundation for any software and is one of the most important tasks. Entire software is supported by four pillars of requirement engineering processes. Functional and non-functional requirements work as bricks to support software edifice. Finally, design, implementation and testing add stories to construct entire software tower on top of this foundation. Thus, the base needs to be well-built to support rest of software tower. For this purpose, requirement engineers come across with numerous challenges to develop successful software. The paper has highlighted requirement engineering challenges encountered in development of software applications and selection of right customer-off-the-shelf components (COTS). Comprehending stakeholder's needs; incomplete and inconsistent process description; verification and validation of requirements; classification and modeling of extensive data; selection of COTS product with minimum requirement modifications are foremost challenges faced during requirement engineering. Moreover, the paper has discussed and critically evaluated challenges highlighted by various researchers. Besides, the paper presents a model that encapsulates seven major challenges that recur during requirement engineering phase. These challenges have been further categorized into problems. Furthermore, the model has been linked with previous research work to elaborate challenges that have not been specified earlier. Anticipating requirement engineering challenges could assist requirement engineers to prevent software tower from any destruction.

Keywords: Requirement Engineering, Customer-off-the-shelf (COTS), Multi-site software development.

1. INTRODUCTION

Software requirements describe the services provided by an application and reflect stakeholder's needs. Requirements are generated from the way people actually work in application domain. The process of eliciting, analyzing, specifying, validating and maintaining requirements is known as *Requirement Engineering (RE).* The main goal of requirement engineering is to meet the degree of end user's satisfaction in minimum cost and time.

Requirement elicitation phase investigates the problems in existing system. However, errors in requirement phase are not identified during application development. Rather they remain concealed until system becomes fully operational and stakeholder's needs are not met [14]. The observation from various researchers [14, 38] illustrate that the cost of fixing an error initially in elicitation process is of little value as compare with other phases of software development. Thus, requirement elicitation plays an imperative role in application development. Requirement engineers have to face myriad problems and difficulties to consult requirements from stakeholders. These problems are then compiled and accumulated into challenges. However, anticipating problems will therefore help requirement engineers to take actions beforehand and prevent software from misfortune.

Additionally, unstructured elicited requirements from operational domain are difficult to manage and model. Requirements need to be concise and well formatted based on any standard requirement specification template [44, 45]. This help stakeholders and maintenance team to understand requirements. Besides, it's a good practice to model requirements so that they can easily be validated by stakeholders. However, poor requirement specifications accelerate the level of ambiguity and requirements become difficult to quantify - resulting in failure of software application.

System requirements explain the detailed description of what software is suppose to do. These requirements are classified as functional requirements which deal with system functionality and non-functional requirements which are software constraints. These requirements are essential for each other and equally critical to achieve. However, decomposition, refinement and validation of these requirements are foremost challenges faced by requirement engineers.

Additionally, most of software applications focus on reusable components for quick development in minimum cost and time frame. Thus, selection of COTS components becomes a major challenge faced by requirement engineers to match stakeholder's requirements with available COTS products [15]. Besides, this introduces new challenges in requirement engineering. Selection of COTS components is often based on subjective judgment. Vendors may take advantage of this and introduce new version for a component, as a result original requirements are modified based on product available in the market. Furthermore, there are no additional specifications provided by vendors for COTS component's internal architecture and descriptions. Thus, requirement engineers have minimum chance to verify whether integrating a particular components with software will meet end user's desire requirements or not. Moreover, some of COTS components are often not tested by real-world users [15].

Prior research studies have often investigated challenges in one particular domain of requirement engineering. However, this paper has merged RE challenges from different domains and accumulated them here. The paper presents and categorized its background study into quadrant that is requirement engineering process, system requirements, applications and product. They are further sub-categorized accordingly. Later, each sub-categorized headings are discussed to identify problems and challenges in that particular area. The paper summarizes different literatures and critically evaluates them. Furthermore, it depicts a framework which elaborates RE challenges that were not highlighted earlier. The framework specifies seven major challenges and classified those challenges into problems. The major factors highlighted in the framework include technological crisis, economic crisis, external events, requirement engineering process difficulties, organizational issues, stakeholder's conflicts and time. Besides, these factors are linked with quadrants of background study to provide a bigger picture of overall RE challenges.

This paper is organized as followed. Section 2 gives an overview of prior research studies in a particular area. In Section 3 challenges highlighted in previous work are critically evaluated. A framework and description of the model is illustrated in Section 4. Finally, section 5 describes the conclusion and future work. References are illustrated in section 6.

2. BACKGROUND STUDY

There are numerous challenges identified by researchers in various requirements engineering domain. Prior studies have usually investigated challenges in only single area of interest such as challenges in

requirement elicitation and analysis [17] or challenges encounter in selection of COTS components [15]. However this section merges those challenges from different literatures. The section categorized background study into four quadrants. These quadrants are further sub-categorized accordingly. Figure1. Shows four major research areas covered in background study. These areas include requirement engineering process, system requirements, applications and product. These areas have been further sub-divided correspondingly. Requirement elicitation, requirement specification and requirement validation have been categorized under requirement engineering process. System Requirement has been sub-divided as functional and non-functional requirements. Application covers challenges in requirement engineering for enterprise application and multi-site software development. Categorically, customer off-the shelf (COTS) have been titled under products. These domains are sum-up in more depth as follow:

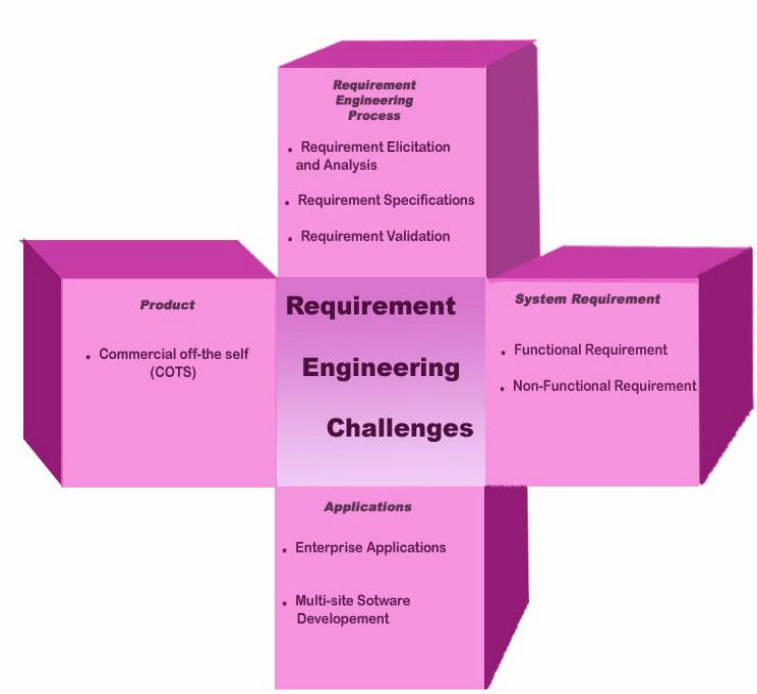

Figure1. Quadrant of research areas for background study

2.1 Requirement Engineering Process

Requirement elicitation, requirement specification and requirement validation have been categorized under requirement engineering process.

2.1.1 Requirement Elicitation and Analysis:

Goldin and Finkelstein study highlighted that it has been a great challenge to comprehend stakeholder's needs and manage unexpected growth of requirements [17]. Quality of the software are contingent to requirement elicitation, requirement analysis and requirement management [18]. The researchers have proposed a method 'abstraction-based requirement management (AbstRM)' to conquer elicitation's challenges in requirement engineering. The information becomes contradictory and incompatible as it has been acquired from different sources. Moreover, manual requirement analysis, discovery of important processes and detection of abstractions (main concept) from scenarios have been foremost challenges for requirement elicitor [19]. The researchers proposed a tool known as AbstFinder [20] which lists important terms known as 'abstraction identifiers.' The meta-concept has been used to classified array of identifiers into different categories such as agents, entities, actions, goals. Explanation for each abstraction identifier is retrieved from scenarios. Furthermore, the identifiers and relationship among them are represented in abstraction network. Omitted information is initially identified by elicitor from AbstRM's network diagram. Besides, impacts of modifications within requirement are also exhibited. Executive

summary and software requirement specification can be written precisely from abstraction identifiers [17]. The researchers have made an empirical assessment of AbstRM method by integration of AbstFinder and DOORS tools. Systematic improvements in requirement engineering process can be made from proposed method.

But however there are certain limitations in proposed tool. A lot of work ought to be done by elicitor to review abstraction identifiers. A situation may occur where noun and verb are not distinguished by AbstRM. For instance a sentence says “book a flight.” Humans can understand that it has been referred to flight reservation [20]. Unfortunately, tool may consider word ‘book’ as a noun. Elicitors have to cross-validate words from source what it really means. Irrelevant or redundant data can also be stated by tool. Additionally, product features and their characteristics have only been specified for requirement engineering tools. They do not explain to what degree the product can be integrated with another requirement tool. Although, the websites like Volere [21] or Requirement tools [22] explained capabilities and integration features, still do not specify those ‘elements’ which can be integrated or which cannot [23]. Hence, a deep analysis of both products is required for integration of requirement engineering tools. Besides, a costly software development life cycle is initiated within requirement engineering process that becomes a challenge.

2.1.2 Requirement Specification:

Firesmith explained the problems in requirement specifications and solutions to prevail over them [26]. Traditional manual based documentation (often used in waterfall development cycle) usually consists of incomplete and vague processes descriptions. Configuration and requirements management are strenuous in manual based specifications. Besides, it is expensive to make copies of specification and distribute to different stakeholders. The paradigm shift from traditional requirement engineering to modern iterative requirement engineering has overcome most of these problems [27]. Iterative approach involves requirement engineering process to be performed repeatedly for identification of bugs in requirements. But substantial time is required for frequent elicitation and specification of software with loads of requirements. Researcher has suggested to structure requirements into models (use- cases) for logical specifications. Object oriented or extended relational databases can be used to store requirements into repository for quick access and verification. Requirement specifications template and requirement engineering tools can also assist in software requirement specifications. The paper has focused on modeling the specifications for minimum traceability issues of requirements.

There are few limitations in specifying the requirements into use-cases [28]. But however the technique is most often used for modeling specifications. However, storing requirements into requirement warehouse can become problematic. Requirement engineers have to enter terabytes of requirements into repository and modify each time when end users change their requirements.

2.1.3 Requirement Validation:

Sequeda has highlighted one of crucial task for requirement engineers are confirmation of requirement specifications. [29] The specifications are usually not guaranteed with completeness and correctness. Requirements are often ambiguous or vague which are difficult to verify. Quality of specifications can be improved from different requirement verification and validation techniques. However, it becomes a challenge for requirement engineers to select among different techniques that best corresponds with requirement specifications. To overcome these problems researcher has proposed a model - taxonomy of requirement specifications. The model divides the specifications into executable and non-executable specifications. Non-executable specifications are written in natural language. These specifications can be verified through using experimental requirement management (ERM) tool. Requirement document is inserted in ERM which saves document in XML format. XSLT is later used to verify document [30, 31]. On the other hand executable specifications are written in declarative languages such as java modeling languages, which are verified through developing prototypes. The paper has explained different requirement verification and validation techniques. Problems in requirements are identified initially which enables to reduce errors in software.

However, expertise in ERM and XSLT is required by requirement engineers in addition to domain knowledge. Besides, building a prototype for user's requirement cannot ensure validation of non-functional requirements.

2.2 System Requirement

System Requirement has been sub-divided as functional and non-functional requirements.

2.2.1 Functional Requirement:

Ya-ning, Shu-jiun, Sum, and Lin investigate various challenges and recommendations to overcome problems in functional requirements [32]. Functional requirements are engaged with comprehensive explanations and complicated structure models which are difficult to reveal. Preliminary unexplored issues for requirement engineers are what functions need to be performed by software and how these requirements should be illustrated. Besides, decomposition of requirements into activities is a complicated task for requirement engineers [33]. Functional requirements gathered by different analysts may become redundant and conflicting. To accomplish the objective of software, researchers have made some recommendations. Gathered requirements need to be categorized and refined. Functional requirements that are gathered by different analysts essentially be coordinated and synchronized. These requirements need to be well understood and expressed systematically by requirement engineers. Furthermore, confirmation of functional requirements needs to be made with stakeholders to reduce future challenges. Concluding, the researchers have advice some recommendations for requirement analysts to overcome challenges in functional requirements.

However, requirement management and traceability of requirements becomes really complex with manually written functional requirements. Therefore, to keep these challenges aside formal methods play imperative role in development of software. 'Formalizing the requirement specification' means specifying the requirement mathematically from set theory and logic. These specifications are verified from set of mathematical based rules to ensure that they meet formal specifications and they are then refined and developed. Besides, formal specifications are concise and often complete which help to understand problem domain and investigate errors. Although implementation of formal methods is costly and gave myriad challenges but they endow with accurate result. Formal specification can also assist to develop test cases easily with minimum human's throughput. Moreover, another approach can be applying Attribute Grammar Rules with Software Process Measurement Application [56]. This approach can assist to determine the decomposition and structure of software processes.

2.2.2 Non- Functional Requirement:

Thomas review specifies that architectural structures are often modified by non-functional requirements [14]. These requirements are poorly specified by stakeholders or they acquired substantial work to be done. Considerably, architectural structure of software is selected among choices based on criteria's such as latency, throughput or high-availability. Therefore, non-functional requirements are not essential to achieve if functional requirements have been fulfilled. Moreover, they are ambiguous to examine. "The system shall be maintainable and robust." Besides, these requirements are not verified by any method [16].The paper illustrated the importance of architectural structure and functional requirements, to achieve desire quality goals.

The paper has got various drawbacks. Functional requirements are what need to be done by system? While non-functional requirements states 'how' the system should achieved that 'what'? Consequently, both requirements are equally critical to achieve [4]. Non-functional requirements are concerned with emergent properties, for instance: reliability, performances or reparability etc [3]. These are constraints and boundaries which are essential to be acknowledged in software development. The importance of non-functional requirement has been grown-up with increased complexity of software and high demand of quality products [1]. However, non-functional requirements to be elicited correctly and completely gave a challenge; interactions with the knowledgeable stakeholders are needed. Researchers have found strategy used in language extended lexicon (LEL) to elicit non-functional requirements [6]. LEL is used to capture terms (phrases or words) peculiar to application field. The vocabulary system consisted of symbols and each symbol is expressed in terms of notations and behavioral response in the operating environment [5]. Additionally, non-functional requirements can be validated by developing tools and

applying abstract interpretation-based static analysis of source program and choosing abstract domains [2]. Although, non-functional requirements gave challenge to be accomplished but they play 'imperative role' in the system.

2.3 Product

Customer off-the shelf (COTS) have been titled under products.

2.3.1 Commercial off-the shelf (COTS):

Alves explained the challenges faced by requirement engineers in selection of COTS products [15]. Generally, organization specifications are not matched with COTS characteristics and requirements are accommodated according to the features present in product. "Let the available COTS feature determine the requirement [15]." Moreover, new updated strategies in COTS might be introduced by vendors. As a result, an erratic situation occurred at times when customers are forced or misguided by suppliers to have adverse product for their organization [12]. The author has justified goal-oriented approach to achieve optimum balance between the requirements and COTS features [13]. The activities involved in goal-oriented approach are identification of goals or objective of the system. Once the goals are established, possible COTS in the market are identified based on their quality and functional aspects. Evaluations of the COTS are matched with the goals. The balance is achieved when the goals collaborated with the COTS features. At the end, the desired COTS product matched with the goals is selected [15].

The limitation of the paper is that the researcher has not focused on the relationship between the COTS features and technology. The specification of the technology in the goals may eliminate assessment of many products in the market. For instance, we may evaluate a product that works on the client-server architecture, while the organization has been operated in distributed system. In such case, it becomes a challenge to judge the right product for the organization requirements. Besides, the modification of requirements according to COTS product available in market may results in the change of business strategies, which become a great risk. COTS components can be evaluated by using fuzzy logic approach [58]. Fuzzy logic is a mathematical based technique to deal with imprecision, uncertainty and information granularity. The approach takes functionality, reusability, performance, security, and portability as input and gives a crisp value of selection efforts.

2.4 Applications

Application covers challenges in requirement engineering for enterprise application and multi-site software development.

2.4.1 Enterprise Application:

Salim highlighted requirement engineering challenges in the development of an enterprise application [7]. The problems encountered by requirement engineers in understanding application domain and business processes are enlightened. Classification of extensive data, providing insufficient information has been a great challenge. Besides, stakeholders have inadequate knowledge or there are no end users for entirely new system [8]. Furthermore, the documentation of software requirements based on standards gave a vital responsibility. Validation and changes within the requirements are also complex [9] [10]. Furthermore, lack of human resources, technical expertise in quality management, knowledge of formalized systems, inadequate knowledge in internal auditing are the foremost challenges faced in small and medium-sized enterprises (SMEs) [57]. The paper can be a helpful source of the material. Requirement engineers can broader their vision to focus on major problems what exist today and how they can better control these challenges to make effective decisions in future.

However, enterprise applications are developed from coalition of business and IT strategies. But unfortunately, there are extensive communication gaps between functional departments. Therefore, it becomes a crucial task for requirement engineers to understand and synchronize the strategies initiated by business departments.

2.4.2 Multi-site Software Development:
Berenbach emphasized on challenges and issues in distributed requirement engineering process [24]. End user's requirements are gathered by requirement analysts who are geographically dispersed. Collected requirements are integrated later for a single software development. Researcher has explained some of distributed structures in distributed requirement process. The problems emerged in these structures have also been pointed out. Inconsistent processes gathered from remote sites create complexity in the requirements. Besides lack of synchronization among analysts are problematic. Requirements gathered from different sites may diverge in applied techniques. For example, site A have used use cases while site B have flow charts. Consequently, requirements are failed to come up with a conclusion what system actually suppose to do? Moreover, un-cleared responsibilities also become confronting [25]. Task assigned to an analyst may presume the responsibility of other analyst. Solutions to these challenges have also been recommended by the researcher. Project manager needs to inspect, a particular tasks has been performed by analysts. Priorities ought to set initially to avoid ambiguities. Additionally, requirements need to be cross-reviewed regularly from remote sites. To achieve an improve coordination among analysts at different sites a facilitator need to be hire. The study aim to find problems in distributed requirement engineering. Researcher has discussed real world scenario of Siemens Corporate.

However, integrated requirements might not correspond with all site's needs. A system may be successful for one site and a failure for another due to miscellaneous organizational culture. Hence, distributed requirement engineering process is also engaged with significant challenges.

The literature review has been summarized in Table 1. The table shows summary for prior researches, main key points and limitations according to particular requirement engineering domain. The limitations in table have been explored by us.

3. CRITICAL EVALUATION

The following section deals with our contribution to prior work. Each of the themes of literatures in previous section is compare among each other.
There are variety of techniques used to collaborate between requirement analysts and end users to elicit requirements. For instance, interviews, questionnaire, ethnography or even return-on-investment (ROI) analysis can identify end user's current operating environment [38, 39, and 44]. However, there are certain advantages and disadvantages in these processes discovery that depends on organization's environment [40].

According to Goldin and Finkelstein, abstraction-based requirement management (AbstRM) surmounts challenges in requirement elicitation [17]. The technique identifies important terms known as 'abstraction identifiers' from application domain. These abstraction identifiers can overcome the challenges highlighted by Firesmith [26] in requirement specifications by formalizing and structuring requirements. For instance, the identified terms can determine name for a particular use case. In addition, variables or objects declared in a prototype for validation of requirements as suggested by Sequeda [29] could be related to general terms used in operating environment. This would help end users to gain better understanding about software requirements and minimize the consequence of requirement engineering challenges. AbstRM does not only state identifiers but distinguish sub-identifiers as well. For instance, identifier 'name' comprise of first name, middle name and last name. Meta-concept used in AbstRM then categorized these identifiers into agent, goals or entities. Thus, the technique can aid to classify extensive data into categories in development of enterprise application and conquer the challenges highlighted by Salim [7]. Moreover, inconsistent processes gathered from remote areas which becomes a challenge in multisite software development explain by Berenbach [24] can be cross-reviewed through network diagram. Contradictory process description identified from incomplete relationships in network diagram can be piloted to navigate and attain further process description.

Domain	Summary	Key Points	Limitations
Requirement Elicitation and Analysis	Comprehending stakeholder's needs is a great challenge. AbstRM has been proposed to overcome elicitation challenges. [17]	AbstRM has been developed by integrating AbstFinder and Doors tools.	AbstRM may not distinguish between nouns and verbs; integration of tools is a challenge.
Requirement Specification	Requirement specifications need to be structured into models (use-cases). [26]	Requirements need to store in a repository for quick access.	Requirement engineers have to enter terabytes of requirements into repository and modify them.
Requirement Validation	A model 'taxonomy of requirement specifications' has been proposed. The model has divided requirements into executable and non-executable specifications for convenient requirement validation. [29]	Different requirement verification and validation techniques have been discussed to overcome the problems initially.	Expertise in ERM and XSLT is required.
Functional Requirement	Problems lie in identifying what software should do? And how to illustrated the requirements; Decomposition of requirements is complicated; Confirmation of functional requirements is essential. [32]	Recommendations can assist requirement analysts to look into the problem deeply.	Static and dynamic requirements which are correlated with functional requirements have not been focused.
Non-Functional Requirement (NFRs)	Architectural structures are modified by non-functional requirements. These requirements are not important if functional requirements have been fulfilled. They are difficult to elicit and verify. [14]	Architectural structures and functional requirements play important role in software development.	Both the system requirements are critical to achieve; Language extended lexicon can be used to elicit non-functional requirements.
Commercial off-the shelf (COTS)	Selection of COTS products gives a major challenge. Goal-oriented approach can be used to achieve optimum balance between end user's requirement and COTS features. [15]	A model has been proposed for activities involved in COTS selection which also explain how to achieve optimum balance between the goals and COTS	No relationship between COTS features and technology has been identified; change of requirements based on COTS available may change business strategies.
Enterprise Application	Problems in understanding application domain; stakeholder's lack of knowledge; standard based documentation; changes within requirements are some of foremost challenges in enterprise application development. [7]	Requirement engineers can analyze deeply to the problems that exists today and how they can better control these challenges.	Synchronization of the strategies initiated by organization departments is a challenge.
Multi-site Software Development	Inconsistency of processes; lack of synchronization among dispersed analysts; use of different techniques, ambiguity in responsibilities are some of challenges in distributed requirement engineering. [24]	Researcher has discussed real world scenario of Siemens Corporate.	Integrating requirements may not correspond with all sites needs due to diverse organization culture.

Table1. Summary of Literature Review

Software requirement specification illustrates a problem and end user's need. Complete requirement specifications have provided software market with substantial assistance to develop and manage software. Researchers have made conclusive studies for requirement specification quality characteristics such as completeness, correctness, conciseness, validation and verifiable [41]. However, requirements gathered in elicitation process needs to be specified and structured. Internet search on 'requirement engineering tools' will list down thousands of tools to generate requirements or depict diagrams or models (use-case).
Automated Requirement Measurement (ARM) Tool [42] is also one of them which endow quality software requirement specifications, to overcome the challenges highlighted by Firesmith in specifying standard based requirements [26]. AbstRM technique suggested by Goldin and Finkelstein search for identifiers and categorizes them; [17] while ARM discovers indicators and generates reports for rectification in specifying requirements [42]. However, DOORS integration with AbstFinder for development of AbstRM needs knowledge of 'Domino Xml Language' (DXL) scripting language. Whereas, ARM has graphical user interface that is more easy and convenient for analysts to specify requirements. Besides, various existing requirement engineering templates [3, 45] can be selected and refined according to organization's requirements [26, 44]. These templates can assist analysts to write consistent and complete specifications. In addition, complex specifications can lead to implicit requirements. Poorly gathered requirements are often redundant and contradictory as identified by Firesmith [26]. To overcome this problem, Sequeda [29] highlighted requirement specifications need to be validated.

Requirement validation and Requirement verification are often used interchangeably. However at times, these terms become bewildering and problematic in identifying either to validate or verify requirements. Requirement validation ensures "Building the right system" or requirements are compiled with correctness and conciseness. Whereas, Requirement verification certify "Building the system right" guarantees that end user's requirements have been completely fulfilled. [9, 49] Validation entails stakeholder's full involvement in reviewing requirement artifacts. [47, 48] Elicited requirements are usually unrefined as they are haphazardly captured from stakeholders. Therefore, to ensure that gathered requirements also reflect correct functionalities about software, requirements need to be validated. "Have we got the requirements right?" is a key question to be initially answered. Goldin and Finkelstein [17] approach to elicit requirements (AbstRM) provide requirement validation through abstraction network diagram. The links between nodes can be used to navigate and obtain more information about a particular area. [17] However, such manual technique needs number of people to review network diagram and requires a lot of time to check missing requirements. Whereas, testing of requirements through execution of prototypes and XSLT method suggested by Sequeda [29] provides much simpler way to validate requirements. Besides, stakeholders are able to visualize and understand requirements more precisely to recognize omitted requirements. Firesmith [26] investigated challenges in reviewing of requirement specifications which are known to be tedious and at same time one of the vital tasks. However, requirement engineers find it difficult to stay attentive and remember the relevant requirements. Therefore, requirement engineering validation tools such as Requirements Assistant [50], SAT [51] or RavenFlow [52] are often used to review or particularize high-quality requirements. These tools ensure to prevent requirements from errors and omissions. Nevertheless, to operate on such requirement validation tools proficient skills and expertise are required. However, in contrast to requirement engineering validation tools Sequeda [29] proposed a model- 'taxonomy of requirement specifications' for validation of requirements. In addition, the method is more efficient rather than deciding and selecting one tool among thousands of requirement validation tools which becomes a challenge for requirement engineers. Furthermore, use of pre-existing components to develop software not only reduce costs but also provides with quality software in timely means. [53] According to Alves, requirements are accommodated with available products in market. [15] This generates new requirements to software development. Therefore, validation of requirements for COTS components need full analysis for a particular component and matching it with end user's requirements. As Salim [7] and Berenbach, [24] explained requirement validation and inconsistent process in development of enterprise application and multi-site software is one of the major challenges in requirement engineering. Requirements needs to be complete, feasible and unambiguous but very seldom these criteria are fulfilled [50].

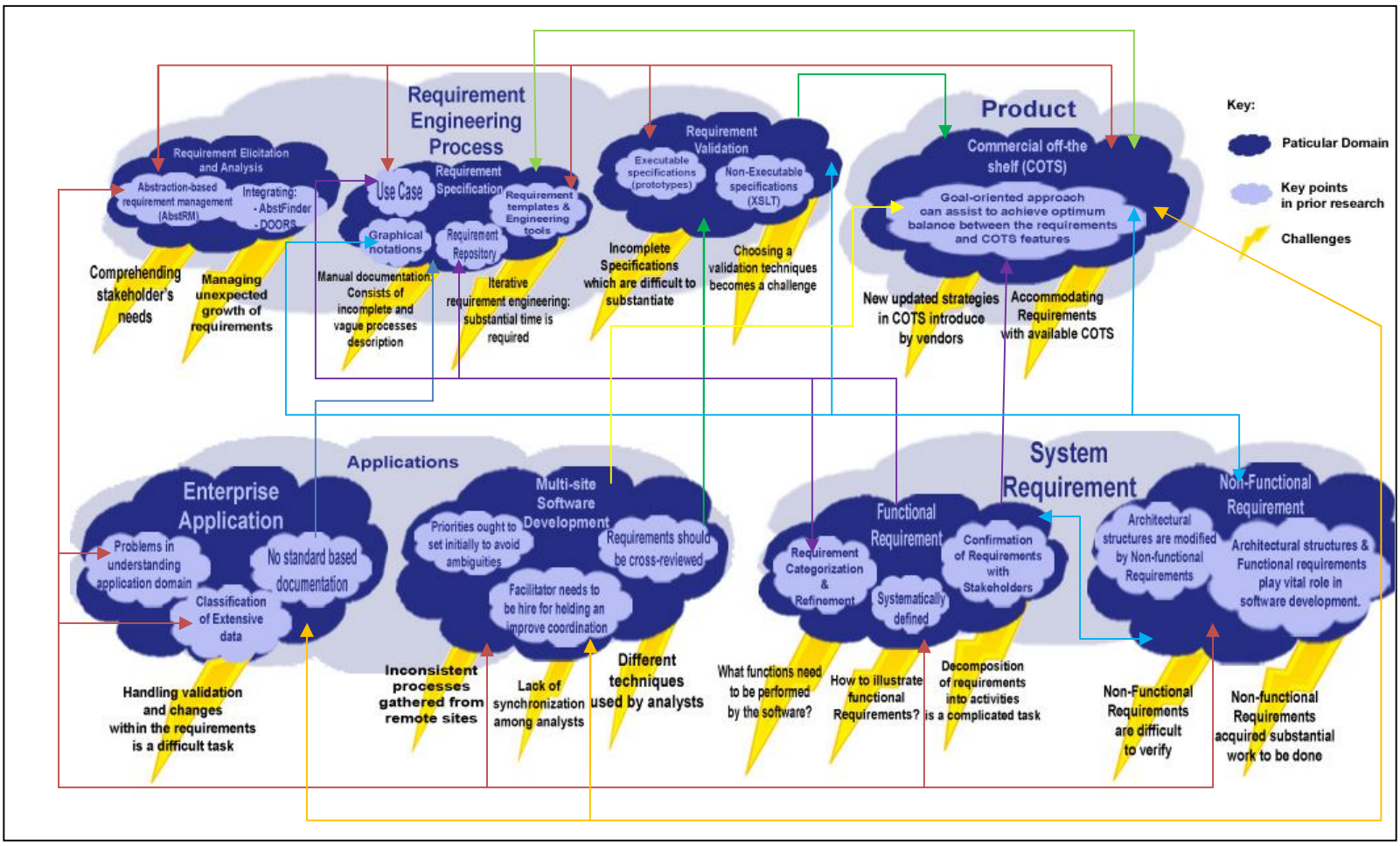

Figure2. Overview of prior research work and their association

Ya-ning, Shu-jiun, Sum and Lin elucidated that functional requirements express a process in terms of relationship between inputs and resulting outputs. [32] These processes are usually missed or undefined and gave major challenges. Hence, omitted process can initially be identified by elicitor from AbstRM's network diagram suggested by Goldin and Finkelstein. [17] Network diagram exhibits association and interdependency among identifiers which can assist to confirm requirements with stakeholders. Furthermore, functional requirements play a vital role in software development and express the behavior of software. These behavior requirements are usually depicted as use-cases in specifications suggested by Firesmith. [26] '*A picture is worth a thousand words*' hence, functional requirement become simple and easy to understand rather intricate explanations. Besides, use-case assists analysts to systematically define and confirm requirements as suggested by Ya-ning, Shu-jiun, Sum and Lin [32]. Moreover, scattered requirements need to be categorized and refined. Thus, Firesmith research gave an idea to compile and store requirements in repository for quick access and verification. [26] Furthermore, functional requirements needs to be validated to ensure that they accept correct data types and are categorized and refined as suggested by Ya-ning, Shu-jiun, Sum and Lin. [32] Thomas review highlighted that functional requirements are important and should only be achieved. [14] However, functional and non-functional requirements are both critically important to achieve. Moreover, requirements are often unclear and vague when elicited from the stakeholders as challenges highlighted by Goldin and Finkelstein. [17] Therefore, introduction of goal oriented approach suggested by Alves [15] in selection of COTS components offers a way to clarify functional requirements through decomposition and refinement of requirement statements. [54]

Non-functional requirements are critical to achieve. However, if these requirements are well elicited, they can reduce the challenges highlighted by Thomas. [14] Therefore, whenever non-

functional requirements are appended or changed; network diagram proposed by Goldin and Finkelstein [17] can trace the impact it produce on other software requirements. In addition, Thomas argues non-functional are ambiguous to examine [14]. However, non-functional requirements are equally important as functional or behavioral requirements. They are concerned with emergent properties that exhibited by software. These requirements are constraints to software such as reliability, performance or maintainability [3]. Unlike behavioral requirements, non-functional requirements are not represented in use-cases. However, these constraints are usually specified as suggested by Firesmith [26] in graphical notations [43] or in mathematical terms. Furthermore, non-functional requirements are not verified by any method [14]. However, Cortesi and Logozzo suggested that non-functional requirements can be validated by developing prototypes or tools and applying abstract interpretation-based static analysis of source program and selecting abstract domain. [2] Moreover, the identification of goals suggested by Alves [15] direct to ask 'what', 'why', 'how' questions. Therefore, goal-oriented approach will provide requirement engineers to understand non-functional requirements and analyzing them with more potential alternatives.

Most of the software application development focuses on reusable components for quick development in minimum cost and time frame. Thus, selection of COTS component becomes a major challenge faced by requirement engineer to match the requirements with available COTS. Therefore, to reduce the challenges as highlighted by Salim [7] and Berenbach [24] in enterprise applications and multi-site software, there need to be a systematic process for selection of COTS components for efficient development of software application. Thus, Alves [15] suggested goal-oriented approach to achieve optimum balance between requirements and COTS features. In addition to select COTS components from goal-oriented approach, abstraction identifiers suggested by Goldin and Finkelstein [17] can also assist requirement engineers to make a checklist in selection of COTS for important terms and ensure that these characteristics have been fulfilled by the evaluated component. Besides, as new updated strategies in COTS are introduced by vendors, COTS-based software requirements are tremendously affecting requirement specifications. As there is cumulative change in requirements corresponding to products evaluated therefore, requirement specifications are also modified resulting in incomplete and out-dated requirements; giving rise to challenges identified by Firesmith [12, 26].

Moreover, Salim [7] explained enterprise applications are complex information systems. They include people, processes, information and technology that interact with each other for accomplishment of goals and objectives. [46] Hence, at times requirement specifications for enterprise applications are complex. Classification of extensive data providing insufficient information; stakeholders inadequate knowledge; no standard based requirement documentation are adding layers to challenges identified by Firesmith [26], Sequeda [29] Ya-ning, Shu-jiun, Sum and Lin [32], Alves [15] and Berenbach [24].

Although different emerging standards like 'IEEE software engineering standards' [3] gave an efficient approach to document specifications, but however there is lack of focus on collecting overall organization's requirements that should be enclosed with development of enterprise application. Consequently, requirement specifications often missed critical and important activity operated in organization environment introducing challenges for requirement engineers and stakeholders. [7]

Berenbach [24] explained emerging collaboration of distinct organizations leads to development of complex multi-site software. [25] Requirements are elicited by analysts at different sites. They may use different techniques and notations for specifying requirements, which becomes difficult to comprehend and cross-review. [24] To prevail over such issue, requirement specifications gather from disperse sites can be stored in distributed requirement repository as suggested by Firesmith [26]. This would help to avoid ambiguities and requirement redundancy in specifications. Furthermore, requirement engineering often directed towards requirement conflicts. For example, analysts at multi-site software have divergent perceptions and directions. Alves suggested [15] identification of goals initially for selection of COTS. However, the approach

can be useful to deal with analyst's conflict as well. Meeting one goal may also interfere with achieving of other goal. [54]

Furthermore, above critical evaluation is depicted in Figure 2. The diagram shows an overview of previous research work. Besides, these research studies have been associated among each other as described in above paragraphs and illustrated in diagram through arrows from different colors.

4. REQUIREMENT ENGINEERING CHALLENGES

Requirement Engineering is a core process for software development life cycle. Bugs in requirements are not identified during development rather they remain concealed until system becomes operational and customer requirements are not met. Poor requirements lead to not only modifications in requirement specifications but require re-designing, re-implementing and re-testing for entire software. Therefore, requirement engineers have to struggle and conquer uncountable numbers of challenges for development of effective and efficient software.

Anticipating requirement engineering challenges will grant opportunities for requirement engineers to enhance software success rate. There have been many investigations conducted to explore different challenges in various domains of requirement engineering. However, these investigations proposed models and gave recommendations to defeat challenges only in a single particular area of requirement engineering (as highlighted in section 2).

In addition to previous research work [17, 26, 29, 32, 14, 15, 7, and 24] and background study, we present a framework for requirement engineering challenges as demonstrated in figure 3. In addition to requirement engineering challenges that are depicted in figure 2 and highlighted in section 3; the model has illustrated more challenges that recur in development of software application and selection of COTS components. Requirement engineering process, System requirements, and Application encounters all these seven major challenges. Whereas, COTS component title under the product only encounters technological, economic crisis and requirement engineering process challenges. The empty spaces in model indicate future problems that can recur in those seven challenges that are highlighted in model.

The model encapsulates overall challenges faced in requirement engineering rather than identifying them in any particular domain. Besides, the model provides with a systematic understanding for requirement engineers to broader their vision and identifies upcoming problems and risks in requirement engineering. Additionally, the model is linked with previous research work to elaborate challenges which were not identified earlier by researchers. Requirement engineering challenges have been categorized into seven components. These components include:

- ✓ Technological crisis
- ✓ Economic crisis
- ✓ External events
- ✓ Requirement engineering process
- ✓ Organizational issues
- ✓ Stakeholder's conflicts
- ✓ Time.

These categorized challenges are further classified into problems that occur during requirement engineering phase. Conclusively, the framework model identifies different problems and later integrates those problems to explore what provoke challenges in requirement engineering.

Requirement engineering problems and challenges presented in the model are explained as follow:

4.1 Technology

Obsolete requirement engineering tool may not provide with accurate functionality for instances, requirement tools for development of prototypes or stimulations. Discarding these requirements engineering tool completely and installing new tool may not be able to convert or emulate the file format. Besides, integrating collaborative features of two requirement engineering tools to obtain functionalities requires deep structural and functional analysis of both available tools, which becomes cumbersome.

In additional, procurement of Customer-off-the shelf (COTS) product is ad-hoc which becomes a challenge later. Selection for COTS products is usually subjective or vague and does not meet customer's needs. The requirements are modified according to available products in market. Besides, configurations in COTS may have major influence on selected product. The new version might not have features that were being evaluated. Thus, underestimating these challenges in selecting accurate component may lead to software failure that does not meet customer's requirements.

4.2 Economic Crisis

IT market is all about new emerging technologies and challenges [35]. Unsolved challenges may increase overall cost of software. For instance unclear software requirements may increase maintenance cost. Besides, there are various other challenges that can come across - Organization developing a system or customers may face financial downfalls during development of software. Increase in accounts payable, out of control spending and poorly planned budgeting strategies can initiate bankruptcy of customer or organization. In addition, variation in depreciation, taxes or stock exchange rates may create difficulties for requirement engineers to manage requirement and select COTS in allocated budget.

4.3 External Events

Targeted effectiveness in software can be achieved if challenging external threats and risk are addressed beforehand [36]. Accidental deletion of valuable data, file corruption, virus-infection or hardware failure may create catastrophe situation for requirement engineers. Besides, external events such as fire, bomb blast or unusual climatic condition may affect requirement engineering process. Consequently, such unpleasant incidents fine an astronomical amount of cost within requirement engineering.

4.4 Requirement Engineering Process

The goal of requirement engineering process is to investigate what tasks need to be performed and what are the boundaries and constraints in software. Acquiring and comprehending requirements for complex domains or critical systems have always been great challenges for requirement engineers. Additionally, stakeholders do not articulate their requirements precisely during requirement discovery process. As a result, requirement specifications are vague, perplexing and ambiguous. Hence, decomposition, modeling of requirements and identification of business processes becomes complicated. Besides, there are over requirement specification which usually defines solutions rather than identifying true problems. Consequently, poor requirement specifications act out as poor process definitions that develop poor software. Validation of requirements improves likelihood of project's success therefore prototypes are developed to ensure requirements and right solution. However, prototype may provide insufficient details due to error occurrence and correcting those errors may allow software to get behind schedule.

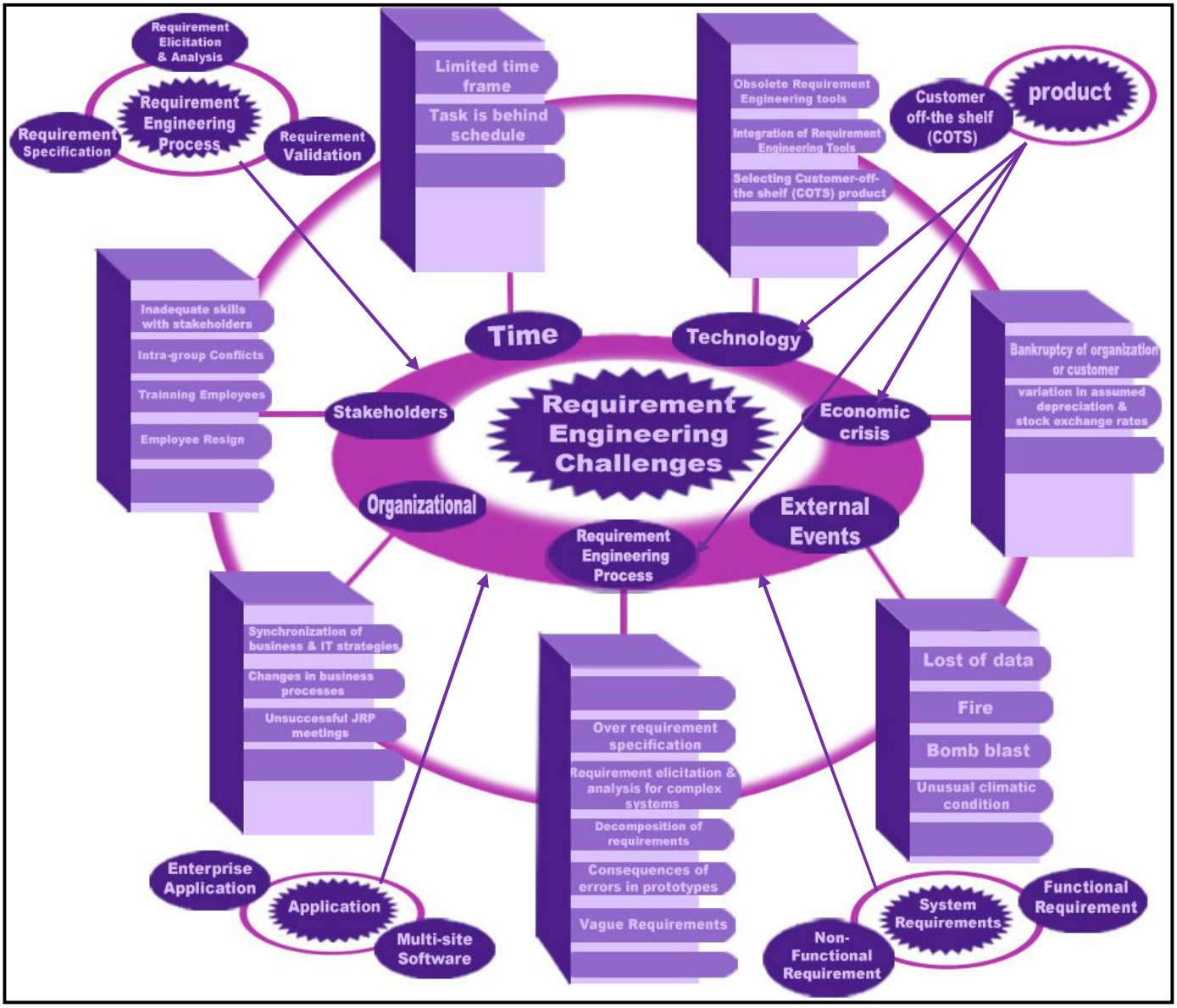

Figure3. Framework for Requirement Engineering Challenges

4.5 Organizational

Software applications are developed from collaboration of business and IT strategies. However, unfortunately there is extensive diversity of perceptions within organizational departments. Hence, aligning and synchronizing strategies recommended by different departments become critical task for requirement engineers.

Additionally, an effective business process represents efficient functioning of an organization. In spite, organizations are rapidly focusing on re-designing of business processes to make substantial changes and improvements in their level of performance. Eventually, changes within business process also transform software requirements. Thus, it acquires substantial efforts to manage these volatile requirements, which set great challenges in requirement engineering.

4.6 Stakeholders

A successful project has a great influence on knowledgeable and experienced stakeholders. Otherwise, software may face significant risks [37]. Inadequate technical skills with requirement engineers and lack of domain knowledge can have a major impact on software. Requirement engineers are unable to adequately address problems and end user's needs. Besides, some pioneer requirement engineers may be ignorant to emergent requirement engineering tools. Therefore, ineffective performance by requirement engineers may results in outdated and error prone requirements.

However, difference in perception or unclear roles and responsibilities leads to confrontations among requirement engineers. These intra-group conflicts may eliminate effective coordination between stakeholders which may have negative impact on performance. Besides, requirement engineer might not be available at critical time or resign from their job. Recruiting and training new employee perhaps not be feasible for successfully completing the development of software within timeframe and budget.

4.7 Time

Scheduling is a process for planning and managing time. Scheduling time is one of the predominantly difficult job and entirely critical to software success. However, usually the time required in completion of tasks during requirement engineering phase is underestimated. As a result, delivery of milestones gets delayed particularly when tasks are on critical path. Great challenges endure for requirement engineers to manage and accomplish seemingly unlimited tasks. Hence, requirement engineers start to take short cuts or sometimes ignore to emphasize and focus on important aspects. Consequently, requirements are poorly established or gets behind schedule. Besides, these futile requirements also lead to downstream failure of entire software.

5. CONSLUSION & FUTURE WORK

Understanding stakeholder's needs; incomplete process description; verification and validation of requirements; selection of COTS products with minimum requirement modifications are foremost challenges faced during requirement engineering. The paper illustrates several problems in requirement engineering domain. These problems have been reviewed from various literatures. Our study is categorized into quadrant of requirement engineering process, system requirements, applications and product. These quadrants are then sub-categorized correspondingly. The challenges and techniques presented by prior literatures have been summarized and critically reviewed. Besides, the paper has made a comparison between different techniques presented in various literatures and had associated those techniques among each other. Moreover, it represents a framework which illustrated those challenges that were not identified by previous research work. The major challenges highlighted in the framework include technological crisis, economic crisis, external events, requirement engineering process difficulties, organizational issues, stakeholder's conflicts and time. These challenges have also been sub-divided into problems. Besides, these challenges are linked with quadrant of background study to provide a bigger picture. Requirement engineering process, system requirements, and application encounter all seven major challenges. Whereas, product only encounters technological, economic crisis and requirement engineering process challenges. There are empty spaces in the framework point to future work in identifying more problems and challenges.
In future, we will be looking forward to prioritize these challenges by calculating the impact of each challenge on development of software applications.

6. REFERENCES

1. G. Maria C. de and J.Brelaz de. *"Improving the Separation of Non-Functional Concerns in Requirements Artifacts."* In proceedings of the 12th IEEE International Conference on Requirements Engineering (RE 2004), 6-10 September 2004, Kyoto, Japan 2004.

2. A. Cortesi and F. Logozzo. ***"****Abstract Interpretation-Based Verification of Non-functional Requirements."* Lecture Notes in Computer Science, Springer Berlin/Heidelberg, 3454: 54-59, 2005.

3. IEEE. *IEEE Recommended Practice for Software Requirement Specification*, 1988.

4. L. Chung, B.A. Nixon, E. Yu, and J. Mylopoulos. *"Non-Functional Requirements in Software Engineering"* Springer Berlin / Heidelberg, pp.363-379 (2009)

5. L. M.Cysneiros and E.Yu. "Perspectives on software engineering", Julio Cesar Sampaio do Prado Leite, Jorge H. Doorn, Kluwer Academic Publishers, pp. 114-138 (2004).

6. L. Marcio and J.C. S. do Prado. "*Using the Language Extended Lexicon to Support Non-Functional Requirements Elicitation."* Anais do WER01 - Workshop em Engenharia de Requisitos, Buenos Aires, Argentina. November 22-23, 2001.

7. J. Salim. "*Requirement Engineering for Enterprise Application Development: Seven Challenges in Higher Education Environment."* World academy of Science, Engineering and Technology, 4:101, 2005.

8. H. d. Vries, H. Verheul and H. Willemse. *"Stakeholder Identification in IT standardization processes."* Standard Making: A Critical Research Frontier for Information Systems MISQ Special Issue Workshop, 2003.

9. A.T. Bahill. and S. J. Henderson. "*Requirements Development, Verification and Validation Exhibited in Famous Failures".* Wiley Periodicals, Inc, Systems Engineering, 8(1): 1-14, 2005.

10. F.T. Sheldon and H. Y. Kim. "*Validation of guidance control software requirements specification for reliability and fault-tolerance."* In IEEE annual proceedings on Reliability and Maintainability Symposium, Washington, DC, USA 2002.

11. M.Oktay, A.B. Gülbağcı, and M.Sarıöz. *"Architectural, Technological and performance issues in enterprise applications."* World Academy of Science, Engineering and Technology, 27: 224-230, 2007.

12. C. Alves, J. B.P. Filho, J.Castro. "*Analyzing the tradeoffs among requirements, architectures and COTS components."* pp. 23-26, 2001.
Website: http://wer.inf.pucrio.br/WERpapers/artigos/artigos_WER01/alves.pdf
Access Date: November 2008.

13. C.Alves and A.Finkelstein. "*Investigating Conflicts in Cots Decision-Making."* International Journal of Software Engineering and Knowledge Engineering, 13(3):1-21, 2003.

14. B.Thomas. *"Meeting the challenges of Requirement Engineering."* News at Software Engineering Institute. 2009. Website: http://www.sei.cmu.edu/library/abstracts/news-at-sei/spotlightmar99pdf.cfm. Access date: January 2010.

15. C.Alves and A.Finkelstein. *"Challenges in COTS Decision-Making: A Goal Driven Requirements Engineering Perspective."* In Proceedings of the 14th international conference on Software engineering and knowledge engineering. Ischia, Italy, 2002.

16. C.J. Fidge and A.M. Lister. *"The challenges of Non-functional computing requirements."* Pp. 6-7.Website: http://sky.fit.qut.edu.au/~fidgec/Publications/fidge93c.pdf.
Access Date: November 2008.

17. L.Goldin and A.Finkelstein. *"Abstraction-based requirements management."* In Proceedings of the international workshop on Role of abstraction in software engineering. Shanghai, China, 2006.

18. A.V.Lamsweerde. *"Requirements engineering in the year 00: a research perspective."* In proceedings of the 22nd international conference on Software engineering, Limerick, Ireland. 2000.

19. Endava: White paper on Requirements gathering and analysis, pp. 7-10, 2007. http://www.endava.com/resources/Endava.com-WhitePaper-RequirementsGathering.pdf Access Date: October 2008.

20. L.Goldin and D.Berry. *"AbstFinder, A Prototype Natural Language Text Abstraction Finder for Use in Requirement Elicitation."* In IEEE Requirements Engineering, 1994. Proceedings of the First International Conference.

21. Volere. Website: http://www.volere.co.uk/tools.htm Access date: November 2008.

22. Requirement tools. Website: http://easyweb.easynet.co.uk/~iany/other/vendors.htm Access date: November 2008.

23. L. K. Meisenbacher. "*The Challenges of Tool Integration for Requirements Engineering."* In Proceedings of SREP'05, Paris, France*,* 2005.

24. B.Berenbach. *"Impact of organizational structure on distributed requirements engineering processes: lessons learned."* In Proceedings of the 2006 international workshop on Global software development for the practitioner. Shanghai, China 2006.

25. S. Timea, H. Andrea and P. Barbara. *"The challenges of Distributed Software Engineering and Requirements Engineering: Results of an Online Survey."* pp.9-13. Website: http://www-swe.informatik.uni-heidelberg.de/research/publications/TR_Distributed_RE_Version1.pdf. Access Date: October 2008

26. D.G. Firesmith. ***"****Modern Requirements Specification."* Journal of Object Technology, 2(2):53-64, March-April 2003.

27. M.Gerdom and Dr. U. Rastofer. *"Rapid requirements engineering – Does a specification Always need to come at the start?"* pp.7.
Website: http://www.compaid.com/caiinternet/ezine/mp-requirements.pdf Access Date: Novemeber 2005.

28. M. Glinz. *"Problems and deficiencies of UML as a requirement specification language."* In Proceedings of IEEE 10th International Workshop on Software Specification and Design. Washington, DC, USA, 2000.

29. F.S. Juan. "*Taxonomy of verification and validation of software requirement and specifications."* Website: http://www.cs.utexas.edu/~jsequeda/pub/Sequeda_VV_req_spec.pdf Access Date: November 2008.

30. Advanced XML validation.
Website: http://www.ibm.com/developerworks/xml/library/x-crsfldvalid/index.html
Access Date: November 2008

31. Schematron: Validating XML using XSLT.
Website: http://www.ldodds.com/papers/schematron_xsltuk.html

Access Date: November 2008.

32. Y.Chen, S.Chen, H.Sum and S.C.Lin. "*Functional requirements of metadata system: from user needs perspective."* In Proceedings of the international conference on Dublin Core and metadata applications: supporting communities of discourse and practice---metadata research & applications. Seattle, Washington 2003.

33. J.H.Hausmann, R.Heckel, G.Taentzer. *"Detection of conflicting functional requirements in a use case-driven approach: a static analysis technique based on graph transformation."* In proceedings of the 24th International Conference on Software Engineering. Orlando, Florida 2002.

34. A.I. Anton, J. H. Dempster and D.F. Siege. "*Managing Use Cases During Goal-Driven Requirements Engineering: Challenges Encountered and Lessons Learned."* Technical Report: TR-99-16, 1999.

35. DK.M: *"Opportunities and Challenges of 21st Century Emerging Technologies."*
Website: http://mi2g.net/cgi/mi2g/reports/speeches/220108.pdf
Access Date: November 2008.

36. K.Thiagarajan. "*Making provisions for external risks."*
Website:http://www.thehindubusinessline.com/iw/2001/03/11/stories/0511e012.htm
Access Date: December 2008.

37. N. Turbit. "*Key Stakeholder Support*"
Website:http://www.projectperfect.com.au/info_key_stakeholder.php Access Date: December 2008.

38. B.Nuseibeh and S.Easterbrook. *"Requirements engineering: a roadmap."* In Proceedings of the Conference on the Future of Software Engineering. Limerick, Ireland 2000.

39. Ian Sommerville: *"Software Engineering"* 7th Edition, Addison.W, pp.168-180 (2004)

40. Search software quality.
Website:http://searchsoftwarequality.techtarget.com/expert/KnowledgebaseAnswer/0,289625,sid92_gci1335438,00.html
Access Date: January 2009.

41. W.M. Wilson, L.H. Rosenberg , L.E. Hyatt. *"Automated analysis of requirement specifications."* In Proceedings of the 19th international conference on Software engineering Boston, Massachusetts, United States 1997.

42. Automated Requirement measurement tool.
Website: http://satc.gsfc.nasa.gov/tools/arm/ Access Date: January 2009.

43. Non Functional Requirements.
Website: http://www.threesl.com/pages/webletter-February06/Non_Functional_Requirements.php. Access Date: January 2009.

44. L. V., J. Donn. *"Writing Software Requirements Specifications."*
Website:http://www.techwrl.com/techwhirl/magazine/writing/softwarerequirementspecs.html
Access date: January 2009.

45. Templates. Website: http://www.stcsig.org/mgt/reference.htm
Access Date: January 2009.

46. Jeffrey L. W., Lonnie D. B. and, Kevin C. D. "*Systems Analysis & Design Methods."* International Edition, Irwin Professional Publishing; pp. 12 (2000)

47. K. Ryan. "*The role of natural language in requirements engineering*." In proceedings of IEEE International Symposium on Requirement Engineering, San Diego, CA, 1993."

48. B.H. C. Cheng and J. M. Atlee. *"Research Direction in Requirement Engineering."* In International Conference on Software Engineering on Future of Software Engineering. Washington, DC, USA, 2007.

49. AOF Requirements and Acceptance.
Webiste:http://www.aof.mod.uk/aofcontent/tactical/randa/content/vandv.htm
Access Date: January 2009.

50. Requirements Assistant. Website: http://www.requirementsassistant.nl/
Access Date: January 2009.

51. SAT. Website: http://www.cassbeth.com/
Access Date: January 2009.

52. RavenFlow. http://www.ravenflow.com/
Access Date: January 2009.

53. W.J.Lloyd. *"A Common Criteria Based Approach for COTS Component Selection."* Journal of Object Technology, 4(3):27-34, 2004.

54. E.Yu and J.Mylopoulos. *"Why Goal-Oriented Requirements Engineering."*
Website: http://www.cs.toronto.edu/pub/eric/REFSQ98.html. Access Date: January 2009.

55. L. Chung. "*Representing and Using Non-Functional Requirements for Information System Development: A Process-Oriented Approach."*
Website: http://www.cs.toronto.edu/~jm/Pub/TSE92.pdf. Access Date: January 2009.

56. R. Atan, A. A. Abd. Ghani, M. H. Selamat, R. Mahmod. *"Automating Measurement for Software Process Models using Attribute Grammar Rules".* International Journal of Engineering, 1 (2): 24-33, 2007.

57. A.S.Poza, M. Altinkilinc, C. Searcy. *"Implementing a Functional ISO 9001 Quality Management System in Small and Medium-Sized Enterprises"*. International Journal of Engineering (IJE), 3(3): 220-228, 2009.

58. Kirti Seth, Arun Sharma, Ashish Seth. "*Component Selection Efforts Estimation– a Fuzzy Logic Based Approach*". International Journal of Computer Science and Security, (IJCSS), 3 (3): 210-215, 2009.