Volume 5 No. 2, February 2015 ISSN 2223-4985

International Journal of Information and Communication Technology Research

http://www.esjournals.org

# Requirement Engineering for Non-Functional Requirements

**Abdelkareem M. Alashqar, Ahmad Abo Elfetouh and Hazem M. El-Bakry**

Information Systems Department, Faculty of Computer and Information Sciences, Mansoura University, EGYPT

## ABSTRACT

Both functional and non-functional requirements (NFRs) must be considered when developing software products. Requirements engineering (RE) is the early phase of software development activity in which the system requirements are developed and managed. The RE process includes eliciting, analyzing, documenting, validating and managing requirements. However, most of the work in the RE activities mainly goes to functional requirements, even though NFRs are often more critical than functional requirements. In this paper we will review the state of the art on how NFRs are treated when conducting the activities of the RE process. Hence a criteria list including the main activities of the RE process is defined as a systematic approach to compare different RE process models. The criteria list is applied to compare six established RE process models followed by recommendations for new RE process method that supports NFRs.

**Keywords:** *Requirements Engineering, Non-Functional Requirements, RUP, V-Model, Volere, XP, SCRUM, WSDM.*

## 1. INTRODUCTION

Both functional and NFRs (also called quality attributes) should be considered when developing software applications. Functional requirements [30] are statements of services the system should provide, how the system should react to particular inputs, and how the system should behave in particular situations. NFRs [30] are constraints on the services or functions offered by the system such as usability, flexibility, efficiency, availability and portability, and often apply to the system as a whole, rather than individual system features or services. High qualified software product is achieved by providing the required level of both functional and NFRs. Moreover, the achievement of quality attributes of a system is intimately connected with the software architecture for that system [18]. RE is concerned with eliciting, analyzing, documenting, validating and managing software requirements. Although NFRs are often more critical than individual functional requirements [30], most RE artifacts focus on the functionality of the software product. Real-world problems are more non-functionally oriented than they are functionally oriented, e.g., poor productivity, slow processing, high cost, low quality, and unhappy customer [5].

There are various studies emphasize the definition (e.g. [14] [25]), classification (e.g. [16] [24]) and representation (e.g. [6] [33]) of NFRs without providing a structured approach on how NFRs are dealt in the RE process.

Some research have been carried out to provide an attempt on how NFRs were dealt in the RE process such as [5] which provides a literature review on NFRs in addition to a discussion to open issues in the early treatment of NFRs and its impacts on software construction. Some research focus on particular activity within the RE process as how NFRs are elicited (e.g. [15]), modeled (e.g. [1] [7] [13]), and verified (e.g. [4]). Our work differs from previous work in that it provides preliminary compressive view on how NFRs are dealt throughout the entire activities of the RE process by comparing some well established RE process models.

The rest of this paper is organized as follows: Section 2 introduces the RE process activities and Sections 3 discusses different RE models. Section 4 provides a comparison between six different RE process models. In Section 5 we conclude the paper and give an outlook to future research.

## 2. RE IN THE SOFTWARE DEVELOPMENT PROCESS

Software development approaches can be classified as being either heavyweight (also called plan-driven development) or lightweight (agile methodologies), based on the extent to which they emphasize factors such as planning , documentation, well-defined phases in the development process, composition of the development team, use of well-elaborated modeling and coding techniques. In heavyweight approaches such as waterfall and RUP there is significant overhead involved in planning, designing, and documenting the system. In such approaches, more time is spent on how the system should be developed than on program development and testing. Furthermore as the system requirements change, rework is essential, and hence the specification and design has to change with the program. Agile (lightweight) approaches such as XP and SCRUM universally rely on an incremental approach to software specification, development, and delivery. They are best suited to application development where the system requirements usually change rapidly during the development process. They are intended to deliver working software quickly to customers, who can then propose new and changed requirements to be included in subsequent iterations of the system. They aim to cut down on process bureaucracy by avoiding work that has uncertain long-term value and eliminating documentation that will probably never be used [23] [30].

RE is an essential phase of the software development process, as errors in this phase will result in problems in the subsequent phases such as design and implementation. The domain of software RE is split into requirements development and requirements management [35]. The requirements development is also subdivided into four main activities namely requirements elicitation, requirements analysis, requirements documentation, and requirements validation. These are the main RE activities proposed by [12] [31] and encompass all the activities involved with gathering, evaluating, and documenting the requirements for a software application. Requirements management activity discussed by [20] [31] and involves: establishing and maintaining an agreement with the customer on the requirements for the software project [22]. Like software process models the activities of RE development activities can be organized into different ways, and it is not necessarily done sequentially or linearly. The linear RE process model was proposed by [22]. Such model is useful when requirements are well-understood from systems users. Iteration is a key to requirements development success [35]. The iterative RE process model was proposed by [21] [30]. In the iterative model the requirements are performed by iterations (releases), each release represents a validated part of the wholes system requirements. Such model is useful when requirements are ill-understood from systems users. The RE process can also be done into spiral [31], each spiral represents a version from the whole requirements of the systems being developed. Each spiral is divvied into phases, and each phase c an activity of RE. The spiral model explicitly handles risk that related to the system requirements.

There is no end to the RE process [20], because when a software product is delivered and users start using it, they discover new needs and uses for it, and they then want it to be extended. This raises new requirements that, in turn, go through the same requirements process.

Despite of the variety of the number, the names and the type of model used in the RE process, we can define seven main activities used in the RE process which will be considered as an evaluation criteria in this study for comparing well-known RE models in terms of their capability to encompass the NFRs. We can summarize seven main activities in the RE process as follows:

1. **Elicitation.** This activity includes identification of stakeholders, capturing, discovering and tracing of business, user, and system requirement. Functional requirements and NFRs are considered during elicitation activity.
2. **Analysis and Negotiation.** This activity aimed at decomposing high-level requirements into details, evaluating feasibility, negotiating priorities with users. Requirements can be prioritized to mandatory, desired or optional. This activity also includes identifying conflicts, determining unclear, incomplete, ambiguous or contradictory requirements and resolving all these issues.
3. **Specification.** This activity focus on documenting the functional requirements that software must provide, and the NFRs that it must respect. All requirements should be specified in a consistent, accessible and reviewable manner.
4. **Modeling.** This activity includes applying certain techniques to produce useful and verifiable requirements models. Models (e.g. UML) include diagrams and their notation.
5. **Verification and Validation (V&V)**. This activity aimed at applying the various tests and means of evaluations used in verifying and validating the requirements. Verification include checking that the software meets its stated functional and NFRs. Validation include checking that requirements actually define the system that the customer really wants.
6. **Management.** This activity includes the monitoring of the changes and the maintenance of the requirements, to ensure that the requirements accurately reflect the product.
7. **Traceability.** This activity focus on documenting the life of a requirement, providing relationship mechanisms between various associated requirements, and tracking changes made to each individual requirement throughout the software development process.

We will consider these seven activities as main criteria to compare six RE models in terms of applying NFRs.

## 3. RE MODELS

A variety of RE models are indentified in the literature and applied by practitioners in the field of software industry. While some of these models used basically for the software development process, other models developed especially for RE purposes. Here is a discussion of six well established RE process models.

### 3.1 Rational Unified Process (RUP)

The Rational Unified Process (RUP) is a modern process model [19] that has been derived from work on the Unified Modeling Language (UML) and it is considered as a good example of a hybrid process model. While conventional process models present a single view of the process, the RUP is described from three perspectives. A dynamic perspective, which shows the phases of the model over time, a static perspective, which shows the process activities that are enacted, and a practice perspective, which suggests good practices to be used during the process [30].

The phases in the dynamic view are: inception phase, elaboration phase, construction phase, and transition phase. The static view of the RUP focuses on the activities (called workflows) that take place during the development process. There are six core process workflows identified in the process and three core supporting workflows. The core workflows include business modeling, requirements, analysis and design, implementation, testing and deployment. The supporting workflows include configuration and management, project management and development environment. The RE discipline pursues the objective of enabling reliable specifications and development, as well as modifications to a system. Really, RE in the RUP consists of the six following principle activities which are: analyzing the problem, understanding the stakeholders' needs, defining the system, managing the scope of the system, changing requirements, refining the system definition. These activities are logically connected to one another and should not be viewed as being purely sequential [28].

### 3.2 Volere

The Volere requirements process is a product of the Atlantic Systems Guild that provides templates for structuring requirements specifications [27]. It differs from other

Volume 5 No. 2, February 2015 ISSN 2223-4985

International Journal of Information and Communication Technology Research

©2015 ICT Journal. All rights reserved

http://www.esjournals.org

approaches that it is developed especially for RE process techniques and it provides a template to guide documenting requirements. This template includes a separate section devoted for NFRs. The Volere process model encompasses the following activities:

- Project Blastoff that builds a foundation for the requirements project by establishing its Scope-Stakeholder-Goals and ensures the project is viable and worthwhile.
- Trawling for Requirements to gain an ability to get people to tell you what they really need through using techniques such as use case workshops, interviewing, brainstorming, etc to discover exactly what the customers need and want.
- Functional Requirements for understanding what are things the product must do.
- Non-functional Requirements which are properties the product must have, such as the desired look and feel, usability, performance, cultural aspects and so on.
- Writing Requirements, which addresses the need to communicate requirements, how to formulate them and how to include an unambiguous fit criterion.
- The Quality Gateway to demonstrate how to test requirements before they become part of the requirements specification. It rejects out-of-scope, gold-plated, non-viable, incorrect and incomplete requirements.
- Managing Requirements to look at strategies for requirements project, the requirements knowledge model, how to prioritize requirements, and how to resolve conflicting requirements.

Volere also proposed a way for considering requirements for agile projects and discovering requirements using prototypes and deviations by testing mock-up products for the user's work [27] [34]. The process should also be considered as being iterative [28].

## 3.3 V-Model

V-Model published in the Development Standards for IT Systems of the Federal Republic of Germany in1997 [11]. The activities of V-Model are organized in a sequential manner in V-Shape. V-Model is tailored into a specific project because V-Model is organization and project independent. The V-Model encompasses three levels: the lifecycle process model, the allocation of methods and the functional tool requirements. At all levels the standards are structured according to areas of functionality (called submodels). There are four submodels: project management, system development, quality assurance and configuration management. The software development submodel includes nine main activities [17]:

- Systems requirements analysis. This activity includes specifying the external view of the system and setting up the system requirements from the user's view.
- System design. This activity involves setting up the system architecture and an integration plan for the architecture.
- SW/HW requirements analysis. This activity involves updating the technical requirements and operational information with regard to the software and hardware requirements.
- Preliminary software design. This activity involves designing the software architecture including the interface between components.
- Detailed software design. In this activity, the software architecture and interface description are more detailed.
- Software implementation. The software modules, components and database are realized.
- System integration. The integration of the system is realized.
- Transition and utilization: This activity compromises the tasks needed for putting the system in operation.

The V-Model provides the following steps in the RE process: description of initial situation and objectives, drawing up functional requirements, drawing up non-functional requirements, establishing risk acceptance, drawing up draft of life cycle and overall system architecture, analyzing quality of requirements, drawing up scope of supply and acceptance criteria [28] .

## 3.4 Programming (XP)

XP is originally proposed by Beck [3] and considered one of the best-known and widely used examples of agile approaches. XP is an "extreme" level approach of iterative development process. In XP several new versions of a system may be developed by different programmers, integrated and tested in a day. XP involves a number of practices such as pair programming (programmers work in pairs), on-site customer involvement, refactoring and collective ownership. The requirements in XP are expressed as scenarios (called user stories), which are implemented directly as a series of tasks. The development team working with customers to develop a "story card" that encapsulates the customer needs. The story cards are the main inputs to the XP planning process or "planning game". Once the story cards have been developed, the development team breaks these down into tasks and estimates the effort and resources required for implementing each task. This usually involves discussions with the customer to refine the requirements. The customer then prioritizes the stories for implementation, choosing those stories that can be used immediately to deliver useful business support. The intention is to identify useful functionality that can be implemented in about two weeks, when the next release of the system is made available to the customer [30].

## 3.5 SCRUM

Scrum is an agile, lightweight process that can be used to manage and control software and product development using iterative, incremental practices [29]. Scrum is gradually popular in the recent few years which combines the advantages of XP and RUP to improve the software efficiency [36]. There are three phases in Scrum. The first is an outline planning phase where developers establish the general objectives for the project and design the software architecture. This phase is followed by a series of sprint cycles, where each cycle develops an increment of the system. Finally, the project closure phase wraps up the project, completes required documentation such as system help frames and user manuals, and assesses the lessons learned from the project.

The requirements are identified in the SCRUM during an innovative central phase called the "sprint cycles". A Scrum sprint is a planning unit in which the work to be done is assessed, features are selected for development, and the software is implemented. At the end of each sprint, the completed functionality is delivered to stakeholders. Key characteristics of this process are as follows [30] [36]:

- Sprints are fixed length, normally 2 to 4 weeks, starts by a product backlog which is a list with all features completed by Scrum team in the current sprint. The customer is closely involved in this process and can introduce new requirements or tasks at the beginning of each sprint. This is reviewed, and priorities and risks are assigned.
- The selection phase involves all of the project team who work with the customer to select the features and functionality to be developed during the sprint.
- Once these are agreed, the team organizes themselves to develop the software. And the team is isolated from the customer and the organization, with all communications achieved by a member called 'Scrum master'. The role of the Scrum master is to protect the development team from external distractions. Unlike XP, Scrum does not make specific suggestions on how to write requirements and test-first development.
- At the end of the sprint, the work done is reviewed and presented to stakeholders. The next sprint cycle then begins.

### 3.6 Web Site Design Method (WSDM)

WSDM was developed by [9] and considered one of the most modern approaches for web application development. It allows web sites and web applications to be developed in a systematic way. WSDM is audience-driven approach (user-centered approach) because it uses the requirements of the intended users to drive the process of web site design. Furthermore WSDM uses the concept "Audience Class" to refer to each intended audience(s) who has his own requirements. WSDM involves the following phases [2]:

- Mission statement: This phase determines, the purpose, subject and intended users are specified, and declares the target audience.
- Audience modeling: It gives the developers a general indication of the audiences involved in the website. Each audience class must determine the functional requirements and information requirements.
- Conceptual Design: let developer to describe the conceptual structure of the web site and model how the members from different audience classes will be able to navigate through the site. In this phase the user requirements, which were informally specified in the previous phase, are elaborated. The conceptual structure of the website is built.
- Implementation Design: the goal of this phase is providing the conceptual design, with necessary implementation aspects. It includes site structure design, presentation design.
- Page Design: in this phase the designer describes where the information on a page should be positioned and how it should be laid out. It is also decided how and where the links (defined in the navigational design) should be presented.
- Implementation: all the information collected so far is taken as input and a website in the chosen and realizing the physical web site, where Implementation environment is automatically generated.

## 4. A COMPARISON BETWEEN RE MODELS CONCERNING NFRS

As a consequence of the discussion provided in the previous section, it can be inferred that the level of achievement of NFRs for each RE activity (criterion) by a certain RE process model is depicted in Table 1. Where "✓" means that process model explicitly support NFRs for specific criterion, "~" means that process model implicitly (partially) support NFRs for specific criterion, and "×" that process model does not support NFRs for specific criterion.

It can be seen from the comparisons that there is no universal and ideal RE model completely support NFRs. Some RE models such as XP and Scum fails to support NFRs in all tasks of RE, while other RE models such as Volere and V-Model implicitly support NFRs in almost all tasks of RE. Although modeling techniques help to clarify requirements for the existing system and to explain new requirements for the proposed system during the RE process, it is seen from the table results that none of the compared models ideally supports modeling techniques for NFRs. For the rest of RE activities on how much NFRs are supported by each compared RE model is disused next.

**Table 1: Comparing six RE process models concerning NFRs**

| RE Activity | RUP | Volere | V-Model | XP | Scrum | WSDM |
|---|---|---|---|---|---|---|
| Elicitation | ~ | ~ | ~ | × | × | ~ |
| Analysis and Negotiation | ~ | ~ | ~ | × | × | ~ |
| Specification | × | ✓ | ~ | × | × | ~ |
| Modeling | × | × | × | × | × | × |
| Verification and Validation | ~ | ~ | ~ | × | × | ~ |
| Management | ~ | ~ | ~ | × | × | ~ |
| Traceability | × | ~ | ~ | × | × | × |

Volume 5 No. 2, February 2015 ISSN 2223-4985

International Journal of Information and Communication Technology Research

©2015 ICT Journal. All rights reserved

http://www.esjournals.org

### 4.1 RUP

The Rational Unified Process is a use-case-driven approach, which means that the use cases defined for the system are the foundation for the rest of development process [19]. Each use case may include scenario(s) which considered as an effective way for capturing system requirements. But scenarios cannot help in eliciting NFRs, since they too focus on user activity and system functionality, rather than NFRs (system qualities) [32]. However NFRs are considered through the elaboration phase of RUP. In this phase the NFRs are captured (elicited) as supplementary requirements [26] with combination of architectural decisions which have to be made with an understanding of the whole system. For this reason, RUP offers partial support for eliciting, analyzing, specifying, validating, and managing NFRs. Nevertheless, RUP lacks in tracking NFRs since it cannot help originally in tracking sources of the main functional requirements.

RUP activities create and maintain models, rather than focusing on the production of large amount of paper documents [26]. For this reason, RUP fails to provide documentation for NFRs as maintaining models focus primarily on functional requirements.

### 4.2 Volere

Although the Volere model was developed especially to handle RE activities, it doesn't fully consider NFRs. However, Volere supports the activity of documenting NFRs well since there is a special section in Volere template dedicated to document NFRs. This not means that Volere explicitly supports NFRs within the remaining RE activities like eliciting, analyzing, etc., because Volere practically uses analysis techniques such as prototyping and user scenarios for elicitation where these techniques don't consider NFRs.

### 4.3 V-Model

The V-Model is not flexible to changes because its phases are rigid and conducted sequentially. As a result, the requirements in V-Model have to be clear and well documented because it is usually expensive to go back and make changes. However, the NFRs in V-Model are determined as a separate process step [28], hence V-Model consider them implicitly during almost all of the RE activities.

### 4.4 XP

In XP, developers work closely with stakeholders to understand requirements (user stories) that must be satisfied by the product being developed. However, in this approach users don't mention NRFs and developers don't push to understand what quality attributes the software should satisfy [8]. Furthermore, user stories can't help in capturing NFRs. Since the XP approach focuses on producing software code via iterative releases, they lack to produce good specification for the both functional and NFRs for the software product being developed. Furthermore, it is implied that XP fails to support managing requirements for both functional and NFRs, this is due to the fact that XP was built around capturing users' refinements after each developed increment which is mainly followed by planning the next one. Also it is hard to manage NFRs in XP since they are related to the entire software product (e.g. efficiency) and typically span multiple user stories rather than individual one.

### 4.5 Scrum

Scrum involves an innovative principle called sprint cycles. The work in each sprint is assessed, the main functionality are captured from user and developed and finally delivered to users, and then the next print cycle begins. The work in each sprint is organized in a to-do-list called backlog which includes different types of items; one type of them is user features. Although the user is closely involved in this process and can introduce new requirements or tasks into backlogs at the beginning of each sprint, such backlogs have a lack in considering NFRs. Scrum does not make specific suggestions on how to write requirements [30]. As in XP, Scrum can't provide support for managing NFRs as backlogs built around considering individual functional services.

### 4.6 WSDM

WSDM does not mention explicitly to NFRs throughout the entire activities. When collecting requirements in WSDM , members of the same audience class have the same information and functional requirements. Followed by a step called audience class characterization, where the characteristics of the different audience classes are given. In WSDM, the result of the audience modeling is a hierarchy of audience classes (groups) together with an informal description of their requirements and characteristics. The requirements here are information, functional and navigational and the usability requirements. Hence, WSDM considers only usability as a NFR in the elicitation process, while it doesn't mention the other NFRs during its process.

## 5. CONCLUSION AND FUTURE WORK

The main objective of this paper was to evaluate well-known RE process models in terms of their ability to fully consider NFRs. A comparison criteria related to the main activities of RE were identified using broad literature review. Using these criteria the six models, RUP, Volere, V-Model, XP, Scrum and WSDM were evaluated by discussing how NFRs are dealt when conducting the main tasks of the RE. From the comparison results it can be concluded that none of these models can fully cover NFRs, although NFRs are considered a critical type of system features. However there is exiting work emphasizes on one individual or some tasks of the RE as stated in the introduction section of this paper. Not considering NFRs will increase the level of software failure risk and reduce the level of software quality, hence we recommend proposing a new model that combines the concepts that stated and reviewed in this paper. The model should be evaluated by experimenting it practically in the real world.

## REFERENCES

[1] Aburub, F., Odeh, M., & Beeson, I. (2007). Modelling non-functional requirements of business processes. Information and Software Technology, 49(11), 1162-1171.

[2] Appelmans, T. (2003). Web Globalization and WSDM Methodology of Web Design. Graduation thesis.

[3] Beck, K. (2000). Extreme programming explained: embrace change. Addison-Wesley Professional.

[4] Chen, M., Tan, T. H., Sun, J., Liu, Y., Pang, J., & Li, X. (2013). Verification of functional and non-functional requirements of web service composition. In Formal Methods and Software Engineering (pp. 313-328). Springer Berlin Heidelberg.

[5] Chung, L., & do Prado Leite, J. C. S. (2009). On non-functional requirements in software engineering. In Conceptual modeling: Foundations and applications (pp. 363-379). Springer Berlin Heidelberg.

[6] Cysneiros, L. M., & do Prado Leite, J. C. S. (2001). Using UML to reflect non-functional requirements. In Proceedings of the 2001 conference of the Centre for Advanced Studies on Collaborative research (p. 2). IBM Press.

[7] Cysneiros, L. M., & Sampaio do Prado Leite, J. C. (2004). Nonfunctional requirements: From elicitation to conceptual models. Software Engineering, IEEE Transactions on, 30(5), 328-350.

[8] Davies, R. (2010). Non-Functional Requirement Do user stories really help? Methods & Tools, Winter 2010, 18(4).

[9] De Troyer, O. M. F., & Leune, C. J. (1998). WSDM: a user centered design method for Web sites.

[10] Computer Networks and ISDN systems, 30(1), 85-94.

[11] Deutschland, B. (2004). V-Modell® XT.

[12] Dupuis, R., Bourque, P., & Abran, A. (1998). Guide to the Software Engineering Body of Knowledge.

[13] Fei, Y., & Xiaodong, Z. (2007). An XML-Based Software Non-Functional Requirements Modeling Method. In Electronic Measurement and Instruments, 2007. ICEMI'07. 8th International Conference on (pp. 2-375). IEEE.

[14] Glinz, M. (2007). On non-functional requirements. In Requirements Engineering Conference, 2007. RE'07. 15th IEEE International (pp. 21-26). IEEE.

[15] González-Baixauli, B., Sampaio do Prado Leite, J. C., & Laguna, M. A. (2006). Eliciting non-functional requirements interactions using the personal construct theory. In Requirements Engineering, 14th IEEE International Conference (pp. 347-348). IEEE.

[16] ISO/IEC 9126-1:2001(E): Software Engineering - Product Quality - Part 1: Quality Model (2001).

[17] Johansson, C., & Bucanac, C. (1999). The V-Model. IDE, University of Larlskrona/Ronneby.

[18] Kazman, R., & Bass, L. (1994). Toward deriving software architectures from quality attributes (No. CMU/SEI-94-TR-10). CARNEGIE-MELLON UNIV PITTSBURGH PA SOFTWARE ENGINEERING INST.

[19] Kruchten, P. (2004). The rational unified process: an introduction. Addison-Wesley Professional.

[20] Leffingwell, D., & Widrig, D. (2003). Managing software requirements: a use case approach. Pearson Education.

[21] Loucopoulos, P., & Karakostas, V. (1995). System requirements engineering. McGraw-Hill, Inc.

[22] Macaulay, L. A. (1996). Requirements engineering. Springer-Verlag.

[23] Meso, P., & Jain, R. (2006). Agile software development: adaptive systems principles and best practices. Information Systems Management, 23(3), 19-30.

[24] Odeh, Y., & Odeh, M. (2011). A New Classification of Non-Functional Requirements For Service-Oriented Software Engineering.

[25] Paech, B., & Kerkow, D. (2004, June). Non-functional requirements engineering-quality is essential. In 10th International Workshop on Requirments Engineering Foundation for Software Quality.

[26] Rational Unified Process. (1998). Best Practices for Software Development Teams, Rational Software White Paper TP026B, Rev 11/01.

[27] Robertson, S., & Robertson, J. (2012). Mastering the requirements process: Getting requirements right. Addison-Wesley.

[28] Schrödl, H., & Wind, S. (2011). Requirements Engineering for Cloud Computing. Journal of Communication and Computer, 8(9), 707-715.

[29] Schwaber, K. and Beedle, M. (2002). Agile Software Development with Scrum. Prentice-Hall.

[30] Sommerville, I. (2010). Software Engineering, Addison-Wesley, Harlow, England, 9 edition.

[31] Sommerville, I., & Sawyer, P. (1997). Requirements engineering: a good practice guide. John Wiley & Sons, Inc..

[32] Sutcliffe, A. G., & Minocha, S. (1998). Scenario-based Analysis of Non-Functional Requirements. In REFSQ (Vol. 98, pp. 219-234).

[33] Van Lamsweerde, A. (2001). Goal-oriented requirements engineering: A guided tour. In Requirements Engineering, 2001. Proceedings. Fifth IEEE International Symposium on (pp. 249-262). IEEE.

[34] Volere. (Accessed 2015). http://www.volere.co.uk.

[35] Wiegers, K. E. (2003). Software requirements. Microsoft press.

[36] Zhi-gen, H., Quan, Y., & Xi, Z. (2009). Research on agile project management with Scrum method. In Services Science, Management and Engineering, 2009. SSME'09. IITA International Conference on (pp. 26-29). IEEE.