# Quality-Oriented Requirements Engineering for Agile Development of RESTful Participation Service

Michael Gebhart
iteratec GmbH
Stuttgart, Germany
michael.gebhart@iteratec.de

Pascal Giessler, Pascal Burkhardt, Sebastian Abeck
Cooperation & Management
Karlsruhe Institute of Technology (KIT)
Karlsruhe, Germany
pascal.giessler@student.kit.edu,
pascal.burkhardt@student.kit.edu, abeck@kit.edu

***Abstract*—Decision-making between humans is a recurring challenge in a society where consensuses for disagreements have to be found. To support such decision-makings, at the Karlsruhe Institute of Technology a Participation Service is developed as part of a service-oriented campus system in an agile manner and based on the Representational State Transfer (REST) paradigm. One of the key success factors of such software projects is the requirements engineering process. Scenarios are an appropriate way to describe a system from the user's point of view. However, existing methodologies do not specify quality requirements for these scenarios. This article presents an enhancement of existing scenario-based requirements engineering techniques to fulfill the quality characteristics of the international standard ISO/IEC/IEEE 29148 and align the quality aspects to the product strategy. We illustrate the approach and the resulting quality improvements by eliciting functional and non-functional requirements for the Participation Service in an agile manner, while considering constraints emerged from the existing RESTful system.**

***Keywords: requirements engineering; agile; scenario; rest; service; participation; iso 29148***

## I. INTRODUCTION

In a society, decision-making is always a recurring and complex challenge. Several stakeholders and participants defend their points of view and try to convince the others of their personal opinion. To overcome these disagreements, consensuses have to be found that satisfy all participants.

To support the process of decision-making in a society by Information Technology (IT), at the Karlsruhe Institute of Technology (KIT) a software solution is to be created that is part of the existing service-oriented KIT Smart Campus System. This system is a collection of functionality for students for supporting their life on the campus of the university. The required software solution consists of a so-called Participation Service that provides the required functionality. The Participation Service is based on the idea of systemic consenting. This approach describes how to find a compromise or consensus that is near to an optimal consensus of the group. For that purpose, compared to usual decision-making processes, possible solutions are not scored with agreement points but with refusing points. This means, after describing the issue and collecting possible solutions, the one is selected that has the fewest refusing points. This solution represents the one with minimum resistance. As the Participation Service is required to be used by different devices, such as smartphones and tablets, it is expected to be developed as a web service based on the Representational State Transfer (REST) paradigm [1] as lightweight alternative to technologies, such as SOAP over Hypertext Transfer Protocol (HTTP), Extensible Markup Language (XML), and Web Services Description Language (WSDL). Furthermore, the service is developed in an agile manner.

One of the key success factors of such software projects is the requirements engineering process [2][3], i.e., the way how functional and non-functional requirements are captured. The usage of scenarios has evolved as an appropriate methodology to describe a system from the user's point of view. As the requirements constitute the basis for the developed software system, the quality of these requirements is very important. For that purpose, the IEEE recommended practice for software requirements specifications [4] and ISO/IEC/IEEE 29148 [5] were created. However, existing requirements engineering methodologies do not consider these quality requirements.

This article enhances existing methodologies in a way that quality characteristics of the international standard ISO/IEC/IEEE 29148 [5] are considered. For that purpose, the quality characteristics in [5] are analyzed and existing requirements engineering methodologies are described step by step and adapted when necessary.

To illustrate the approach, the resulting methodology is directly applied to the Participation Service at the KIT as a real-world project. After identification of stakeholders, the goals are elicited and prioritized. Finally, functional and non-functional requirements are formalized that fulfill the quality characteristics.

The article is organized as follows: Section II examines existing work in the context of requirements engineering methodologies. The Participation Service scenario is described in Section III. In Section IV, our methodology is presented and directly applied to the scenario by considering the quality characteristics and existing constraints. Section V concludes this article and introduces future research work.

## II. BACKGROUND

This section analyzes existing approaches in the context of requirements engineering methodologies that identify the goals of stakeholders and writes them down in a precise way so that they can be used in the following development phases [6].

In [4], the IEEE offers an official recommended practice for software requirements specifications, which was replaced by the new international standard ISO/IEC/IEEE 29148 [5]. Based on them, quality characteristics for high quality requirements can be derived. Furthermore, the new standard provides language criteria for writing textual requirements and requirements attributes to support requirement analysis. It also provides guidance for applying requirements-related processes. These concepts will be used to analyze existing scenario-based requirements engineering methodologies and to design the one introduced in this article.

Sharp et al. [7] present a domain-independent approach for identification of the stakeholders based on four determined groups of so-called baseline stakeholders. They can be further refined in three different groups based on their role. This approach will be used to identify the stakeholders in this article. However in large projects, the resulting network of stakeholders can be huge.

For that reason, Ackermann et al. [8] describe a method with a matrix in which the stakeholders were arranged by their importance and their influence on the project. This method can be used to prioritize the discovered stakeholders for the project.

There are different requirement types, which have to be taken into account when eliciting requirements for a software product. Glinz [9] provides a concern-based taxonomy of requirements, which consists of functional requirements, non-functional requirements, and constraints. These types will be reflected in the introduced requirements engineering methodology, however with one difference: The performance will not be considered as a separate entity since it is already an ingredient of [10].

For eliciting functional requirements, Rolland et al. [11] present a goal modeling approach by using scenarios. A goal represents something that the stakeholders want to have in the future, while a scenario represents the required interactions between two actors to achieve the corresponding goal. Once a scenario has been composed, it is investigated to addict more goals. This approach can be aligned with [5], which is why it will be reused in this article.

However, there are two issues: 1) Goals cannot be regarded separately, because they could be composed of existing goals and 2) the recursive process is repeated until no more subgoals can be derived, but this can lead to a big bunch of subgoals. A solution for 1) is a repository of already analyzed goals, which can be reused by reference. The determination of a threshold in 2) is difficult, because it cannot be set easily by metrics. So the requirements engineer has to decide on its own when the abstraction meets its expectations. For this purpose, some conditions had to be found, which support the decision-making. Furthermore, it is not obvious, how to achieve the initial goals.

At this point, Bruegge and Dutoit [12] introduce some interview questions that can be used for identification of the initial goals. Furthermore, elicitation techniques can be found in [2]. To support agile software engineering, the discovered goals have to be arranged by importance to select the goals with the highest rank similar to iteration.

For that reason, the approach by Karlsson and Ryan [13] will be applied, which uses pairwise comparisons in consideration of cost and value. But, for many goals, this approach will rapidly become impracticable as the number of comparisons increases significantly. For that reason and the statement “Keep the prioritization as simple as possible to help you make the necessary development choices” by Wiegers [14], a simple classification approach with three different scales based on [4] is best suited for the initial prioritization.

When writing scenarios, the quality characteristics by [5] have to be considered. Glinz [15] presents an approach, which respects the quality characteristics by the old recommendation [4]. His findings will be used to improve the quality of requirements.

Also, Terzakis [16] presents techniques for writing higher quality requirements by providing an overview of requirements and pitfalls by using the natural language for their description. Based on this, the quality of requirements will be improved even further.

In [10], the ISO provides a quality model comprising quality characteristics that are further decomposed into sub-characteristics. This model will be used for determining the quality aspects of a software product.

For eliciting non-functional requirements, the approach by Ozkaya et al. [17] will be used. Due to the fact that statements like “The system shall be maintainable” are imprecise and not very helpful, this approach is using so-called quality attribute scenarios. Based on these, the corresponding quality characteristic of ISO 25010 [10] can be derived. However, for many quality characteristics it can be very time-consuming.

To reduce the effort, the decision-making approach by Saaty [18] will be applied by using pairwise comparison of the quality characteristics in [10] with regard to their importance for the product strategy.

With the provided constraints of the architectural style REST in [1], the last requirement type according to the taxonomy in [9] will be considered.

## III. SCENARIO

To illustrate the requirements engineering approach, the KIT Smart Campus System at KIT is to be enhanced by a new service, the Participation Service. The Participation Service is designed to support the process of decision-making between professors, students, and other KIT members according to the principle of systemic consenting.

In the first phase, participants can create and describe their own subjects of debate and share them to a group of participants. In the second phase, the participants rate suggestions by expressing their dislike instead of their like as usually expected. They are able to do that in the form of refusing points from zero to ten. Refusing points indicate

how much a participant dislikes a possible suggestion. Thus, rating a suggestion with zero refusing points means that the participant totally agrees with the suggestion. Rating a suggestion with ten refusing points means that the participant rejects the suggestion. The suggestion with the fewest amount of refusing points represents the one with the highest acceptance of all participants. This suggestion has minimum resistance and is the consensus of the group. Fig. 1 illustrates the described process. For example, the Participation Service can be used for determining new lecture contents in collaboration with students in the context of the Research Group Cooperation & Management (C&M).

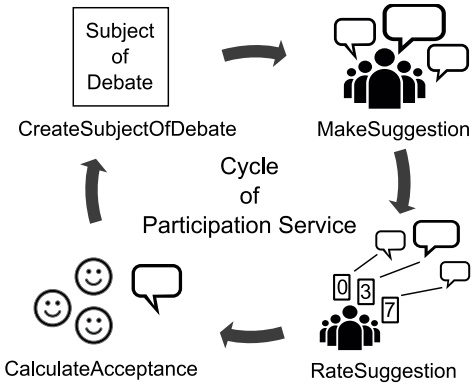


Figure 1. Systemic consenting process.

For illustration of our scenario-based requirements engineering technique, the simple goal "Rate a suggestion" of the Participation Service was chosen: A participant requests the website of the Participation Service and gets to see a login screen. After he logged in correctly, he gets a list of subjects of debate. He selects a subject of debate, which he is interested in. He sees a description of the subject and a list of suggestions sorted descending by acceptance. Once reading all suggestions, the participant rates each suggestion with refusing points from zero to ten to express his dislike against the suggestion. The Participation Service updates the acceptance of each suggestion and rearranges them.

## IV. Quality-Oriented Requirements Engineering for Agile Development of RESTful Participation Service

In this section, our requirements engineering methodology is introduced. This represents our proposed solution for gathering requirements that verifiably fulfill quality attributes introduced in ISO/IEC/IEEE 29148 [5]. This can be proven to the customer. First, the quality characteristics of the standards [4] and [5] are presented. Next, the stakeholders are identified followed by an elicitation of their goals. With the prioritization of the goals, they are selected for the iteration. Afterwards, the functional, non-functional requirements are discovered and documented according to the derived quality characteristics of [5] and the provided taxonomy by Glinz [9]. The entire requirements engineering methodology is shown in Fig. 2.

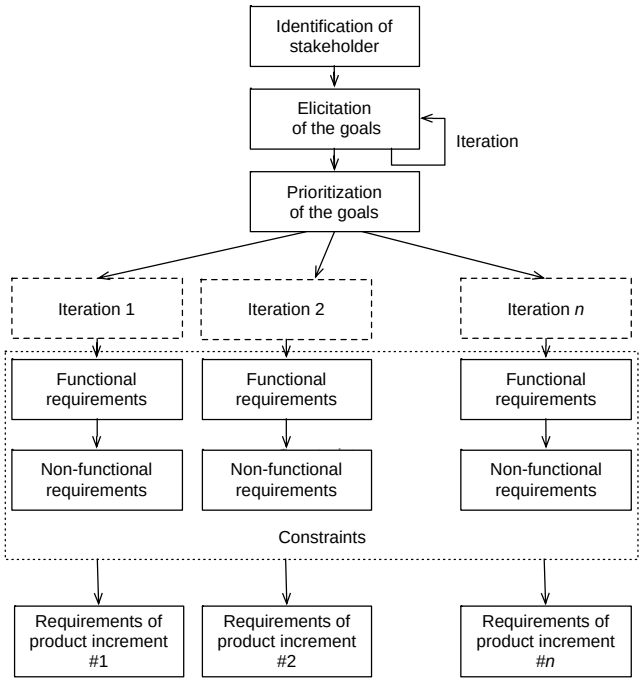


Figure 2. Requirements engineering methodology for agile development of RESTful Participation Service.

### A. Quality Characteristics for Requirements

According to the IEEE [4], the requirements quality focuses on correctness, unambiguousness, completeness, consistence, prioritization, verifiability, modifiability, and traceability. [4] was replaced by the international standard ISO/IEC/IEEE 29148 [5], which introduces feasibility, necessity, free of implementation, and singularity as new characteristics for requirements while removing prioritization, correctness and modifiability. Furthermore, the new standard distinguishes between individual and a set of requirements. According to them, a set of requirements shall be complete, consistent, affordable, and bounded. These can be fulfilled by ensuring the individual ones. In the following, we consider the full set of quality characteristics for individual requirements of the current standard [5].

### B. Identification of Stakeholders

In the elicitation phase, all stakeholders of the project have to be identified. A missing stakeholder can lead to incomplete requirements, which endanger the project success. For this purpose, we apply the approach by Sharp et al. [7]. Based on the four groups a) users, b) developers, c) legislators, and d) decision-makers, for the Participation Service, we could identify all stakeholders as listed in Table I and assign them to the corresponding scrum role.

The prioritization of the stakeholders with regard to their influence on the project was not necessary at this point. Due to the fact that the complexity of the project and the amount of involved stakeholders is not as high as in an industrial project.

TABLE I. STAKEHOLDERS OF THE PARTICPATION SERVICE

| Group | Stakeholders |
|---|---|
| Users | Enrolled students and members of the KIT |
| Developers | Students at C&M and KIT as operator of the Participation Service |
| Legislators | State of Baden-Wuerttemberg and Federal Republic of Germany |
| Decision-Makers | C&M leader, C&M members and one expert of systemic consenting |

### C. Elicitation of Goals

After the identification of stakeholders, the elicitation of goals can be initiated. For this purpose, the interview and brainstorming technique was chosen and the questions introduced by Bruegge and Dutoit [12] were used for easier discovery of the goals according to the definition by [11]. Each goal corresponds exactly to one requirement in order to fulfill the singularity according to [5]. An excerpt of the determined goals is shown in Table II. Goal *G2* will be further refined in the upcoming sections.

TABLE II. EXCERPT OF GOALS OF THE PARTICIPATION SERVICE

| ID | Goal | Stakeholder |
|---|---|---|
| G1 | Logs in at the Participation Service | C&M member |
| G2 | Rate a suggestion | C&M member |
| G3 | Add a new proposal for solution | C&M member |

In contrast to traditional software methodologies, such as the waterfall approach, in agile development more goals can be added in the course of the software project.

By investigating the quality characteristic of the current standard [5], we discovered that the meaning was changed compared to [4]. In [4], requirements were expected to be complete for the entire system. According to the current standard, a set of requirements contains everything to define a system or only a system element. This allows us, to use iterations in which system elements are described.

### D. Prioritization of Goals

The next step is the prioritization of the goals with regard to their importance for the stakeholders. Due to the abstraction level of the goals and the statement by Wiegers [14], we applied a simple classification approach based on a three-level scale (essential, conditional, optional) according to [4]. In order to prevent ambiguousness, each stakeholder has agreed on the meaning of each level [14]. After rating of goals, a specific amount of highest ranked goals, which reflects the necessity [5], form the basis for the first iteration. The amount depends on the estimated velocity of the development team and expected effort for the implementation. In this context, the essential goals are those presented in Table II.

### E. Functional Requirements

For each selected goal, a scenario will be authored or reused that describes the required interactions to reach the goal. Based on a scenario, further goals can be derived. The combination of a goal and the corresponding scenario is called requirement chunk as described in [11]. Fig. 3 illustrates this by showing a meta-model that defines the rules and the elements of a requirement chunk.

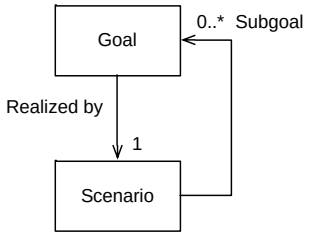


Figure 3. Meta-model of a requirement chunk.

This recursive process with objective of functional decomposition can be aligned with the process defined in the standard [5]. But, this recursive process can be repeated several times, which results in rising costs.

For that reason, we propose three conditions that serve as abort criteria for the process. If all of the following conditions apply, the process can be aborted:

1) no additional benefit in form of new derived goals
2) other scenarios will definitively not reuse atomic actions of the current scenario
3) the size of the scenario exceeds more than 20 atomic actions

According to Glinz [15], the decomposition in user functions and the ease of understanding assure the precondition of correct specification. Furthermore, the decomposition allows us to describe the capability and properties of a given requirement chunk in detail according to the stakeholder's need, which represents the completeness of individual requirements. In the following, authoring and reusing of scenarios will be presented.

#### E.1. Reusing Scenarios

In the best case, a requirement chunk still exists in the repository, which contains all analyzed goals and their scenarios. Therefore, redundant scenarios will be avoided, which ensures the consistence regarding to a set of requirements. As a result, we can compose different requirement chunks to support higher goals. For example, the goal *G1 "Logs in at the Participation Service"* represents a cross-sectional goal, which will be used by *G2* and *G3*. Furthermore, the usage of cross-references results in an increase of consistence [5]. This is because scenarios can be related to one another in a meaningful way, which allows detection of conflicts.

*E.2. Authoring Scenarios*

If no requirement chunk for the given goal can be found in the repository, a new scenario has to be authored while considering the quality characteristics by [5].

The unambiguousness cannot be fulfilled properly as we use the natural language with inherent equivocality for the description of the scenario [4]. So a trade-off between ease of understanding and formalism has to be made. For this, we used the provided meta-model of a scenario by Rolland et al. [11] to reduce equivocality. Moreover, we used the introduced structural constructs of Glinz [15] to further reduce the level of equivocality. To detect ambiguousness during description or validation of scenarios, Terzakis [16] offers a detailed checklist. Also, the current standard [5] provides some terms, such as superlatives or vague pronouns, which should be prevented to ensure bound and unambiguousness. For newly introduced terms and units of measure, we have created a separate document, which acts as a glossary.

According to [5], a scenario should be implementation free. This means that no architectural design decisions take place in this phase. This is the nature of a scenario as it describes what is needed in form of a concrete instance to achieve its intended goals. The nature of a scenario also allows us to derive acceptance criteria to verify the requirements in the form of test cases [15], which fulfills the verifiability [5].

The feasibility is another quality characteristic of the standard [5] with focuses on technical realization of the requirement. At this point, the scenario has to be investigated with regard to system constraints such as the existing environment (cf. Section G).

| | | | |
|---|---|---|---|
| Title: | Rate a proposed suggestion | ID: G2 | Priority: High |
| Source: | C&M member | Risk: Middle | Difficulty: Nominal |
| Rationale: | Integral ingredient of systemic finding | Version: 1.0 | Type: Functional |

| | |
|---|---|
| Initial State: | User wants to rate a proposed solution |
| Final State: | User rated a proposed solution |
| Dependable goals: | No dependable goals |

| Nr. | Normal action flow | Ref. |
|---|---|---|
| 1 | User logs in at the Participation service | G1 |
| | System verifies the credentials | |
| 2 | System redirects him to the secured area (Def. 1.1) | - |
| | User gets a list of available subjects of debate | |
| 3 | User selects a subject from the provided list | - |
| | System receives the selection and redirects him to the subject of debate | |
| 4 | User rates a proposed solution by selecting the refusing points | G5 |
| | System calculates the acceptance of the suggested solution | |
| Nr. | Concurreny / Alternative action flow | |
| 2' | *IF* the list of available subjects is empty<br>*THEN* the system displays „There are currently no subjects of debate“<br>*TERMINATE* | |

Figure 4. Style for representation of scenarios.

To ensure the traceability [5], each scenario must have a unique identifier. In the course of modification over time, the scenarios also need a version number representing the current state. Due to the fact of reusing scenarios, each scenario should also be aware of dependable requirement chunks to clarify, which requirement chunks will be affected by modifications of one scenario.

Based on these findings, the representation in [15], and the provided requirement attributes in [5], we created a style for representation of scenarios, which is illustrated in Fig. 4. Similar to the approach by Glinz [15], the representation can also be easily transformed into a state chart.

*F. Non-Functional Requirements*

After all goals have been analyzed, the resulting requirement chunks represent the functional aspects of the system. Each scenario can now be investigated with regard to non-functional aspects. For this purpose, we use quality attribute scenarios by Ozkaya et al. [17] and link these with the corresponding requirement chunk.

The stimulus represents the condition for the release of the event, while its source is the entity that triggers it. The response is the activity of the stimulus. The environment, such as normal operation of a service, stands for the constraint under which the stimulus occurred. The functional scenario represents the stimulated artifact. Finally, the response measure represents the measure for evaluating the response of the system.

To align this with the product strategy, the product quality characteristics [10] have to be ranked by their importance for the stakeholders. For example, the security is probably more important than the user experience for a product in the bank sector. This is why we used pairwise comparisons of the quality attributes according to the Analytical Hierarchy Process (AHP) by Saaty [18].

The results have shown that for the Participation Service security, functionality and usability are more important than the others. Based on this result, we could focus on the most important quality attributes. Nevertheless, we still have to keep the quality attributes with minor importance for the product strategy in mind. We can thus reduce the effort for eliciting the non-functional requirements since resources, such as time, often limit a project.

| | | | |
|---|---|---|---|
| Type: | Usability | ID: N2 | Priority: 0.18 |
| Source: | C&M member, students | Risk: Low | Difficulty: Easy |
| Rationale: | Better user experience | Version: 1.0 | Ref: G2 |
| Quality attribute scenario | *Source of stimulus:* | User | |
| | *Stimulus:* | clicks on the button | |
| | *Environment:* | during normal operation, | |
| | *Response:* | the system gives a feedback | |
| | *Response measure:* | within a period of 200ms | |

Figure 5. Style for representation of quality attribute scenarios.

Similar to the description of the functional scenarios (c.f. Section E), we have to respect the same conditions. This is why we do not describe this in detail at this point.

For the prioritization of non-functional requirements, we used the ranked result of the AHP. But, it is also possible to add another prioritization step, such as the ones mentioned in [14] or [17]. Fig. 5 shows one non-functional requirement of goal *G2*.

### G. Constraints

According to Glinz [9], the constraints restrict the solution space for the functional and non-functional requirements. For example, a constraint can be company-based human interface guidelines, legal issues, or existing environments [9]. With regard to the Participation Service, we only had to investigate the constraints emerging from the existing environment: As described in the introduction, the Participation Service should be a part of the existing service-oriented KIT Smart Campus System based on REST. The usage of REST as an architectural style requires the consideration of six specific characteristics according to Fielding [1]: client-server, stateless, caching, uniform interface, layered architecture, and optionally code on demand. These constraints were written down in a separate constraints document similarly to the glossary so that we are able to reference this over the whole iteration cycle with regard to the feasibility [5].

## V. Conclusion and Outlook

In this article, we introduced a requirements engineering methodology that is based on existing approaches and considers quality characteristics of the ISO/IEC/IEEE 29148 standard [5]. For that purpose, we analyzed the quality characteristics of [5] and enhanced existing methodologies for scenario-based requirements engineering.

We illustrated our approach by means of the Participation Service developed at the KIT. By applying our methodology on the Participation Service we could improve the quality of our requirements. For example, we detected some inconsistencies during the authoring of the scenarios and reduced the communication effort and the costs emerged from misunderstandings.

Compared to the previous IEEE recommendation [4], it is easier to meet the desired qualities of ISO/IEC/IEEE 29148 [5]. The reason for this is that the new standard does not give tough specifications for the satisfaction of the quality characteristics. Due to the fact that in a scenario-based approach we are using the natural language for describing requirements, we can only merely reduce the ambiguousness and not prevent it completely. However, this does not imply bad requirements but rather potential for improvements.

Our approach helps requirements engineers and business analysts with capturing and describing high-quality functional and non-functional requirements in a systematic manner. The quality characteristics are standardized [5] and represent recognized criteria for requirements. With our approach, requirements engineers and business analysts can capture new requirements that fulfill these criteria in a systematic manner or improve existing requirements.

In this article, we have focused on the requirements analysis phase of RESTful services. For the future, we plan to focus on the quality assurance of RESTful services during the design phase as part of an agile development process. After we have shown how to assure the quality of functional and non-functional requirements that constitute the basis for the development, the design phase has to consider quality as well. We will examine how to evaluate a service design as the one for the Participation Service regarding widespread quality characteristics and design patterns for RESTful services. For that purpose, we will enhance our existing work in the context of quality assurance of service-oriented architectures [19].

## References


[1] R. Fielding, “Architectural styles and the design of network-based software architectures,” University of California, Irvine, 2000.

[2] Standish group, “Chaos report,” http://www.projectsmart.co.uk/docs/chaos-report.pdf, 1995, Accessed 2014-05-21.

[3] A. F. Hooks and K. A. Farry, “Customer centered products: creating successful products through smart requirements management,” American Management Association, 2000, ISBN 978-0814405680.

[4] IEEE, IEEE Std 830-1998 “Recommended practice for software requirements specifications,” 1998.

[5] ISO/IEC/IEEE, ISO/IEC/IEEE 29148:2011 “Systems and software engineering – life cycle processes – requirements engineering,” 2011.

[6] B. Nuseibeh and S. Easterbrook, “Requirements engineering: a roadmap,” The Future of Software Engineering, Special Volume published in conjunction with ICSE, 2000, pp. 35-46.

[7] H. Sharp, A. Finkelstein, and G. Galal, “Stakeholder identification in the requirements engineering process,” Database and Expert Systems Applications, 1999, pp. 387-391.

[8] F. Ackermann and C. Eden, “Strategic management of stakeholders: theory and practice,” Long Range Planning, Volume 44, No. 3, June 2011, pp. 179-196.

[9] M. Glinz, “On non-functional requirements,” 15th IEEE International Requirements Engineering Conference (RE 2007), 2007, pp. 21-26.

[10] ISO, ISO/IEC 25010:2011 “Systems and software engineering - systems and software quality requirements and evaluation (SQuaRE) - system and software quality models,” 2011.

[11] B. C. Rolland, C. Souveyet, and C. B. Achour, “Guiding goal modeling using scenarios,” IEEE Transactions on Software Engineering, Volume 24, No. 12, 1998, pp. 1055-1071.

[12] B. Bruegge and A. H. Dutoit, “Object-oriented software engineering: using uml, patterns and java,” Pearson Education, 2009, pp. 166-168.

[13] J. Karlsson and K. Ryan, “A cost-value approach for prioritizing requirements,” IEEE Software, Volume 14, No. 5, 1997, pp. 67-74.

[14] K. Wiegers, “First things first: prioritizing requirements,” Software Development, No. 9, Volume 7, Miller Freeman, Inc, September 1999, pp. 48-53.

[15] M. Glinz, “Improving the quality of requirements with scenarios,” Proceedings of the Second World Congress on Software Quality, Yokohama, 2000, pp. 55-60.

[16] J. Terzakis, “Tutorial writing higher quality software requirements,” ICCGI, http://www.iaria.org/conferences2010/filesICCGI10/ICCGI_Software_Requirements_Tutorial.pdf, 2010, Accessed 2014-07-16.

[17] I. Ozkaya, L. Bass, R. L. Nord, and R. S. Sangwan, “Making practical use of quality attribute information,” IEEE Software, April 2008, pp. 25-33.

[18] T. L. Saaty, “How to make a decision: the analytic hierarchy process,” Informs, Volume 24, No. 6, 1994, pp. 19-43.

[19] M. Gebhart, “Measuring design quality of service-oriented architectures based on web services,” Eighth International Conference on Software Engineering Advances (ICSEA 2013), Venice, Italy, October 2013, pp. 504-509.