# Quality-Oriented Requirements Engineering of RESTful Web Service for Systemic Consenting

Michael Gebhart, Pascal Giessler
iteratec GmbH
Stuttgart, Germany
michael.gebhart@iteratec.de,
pascal.giessler@iteratec.de

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

*Abstract*—**Making decisions is a typical and recurring challenge in a society as humans often have different opinions concerning a certain issue. Consensuses have to be found that satisfy all participants. To support the finding of consensuses, at the Karlsruhe Institute of Technology a new software service is developed, the Participation Service, to support the systemic consenting. This service is expected to be part of the already existing service-oriented campus system of the university that supports students in their daily life. The Participation Service is expected to be developed in an agile manner. Furthermore, as the entire architecture is based on the Representational State Transfer paradigm, also the new service is expected to be RESTful. One of the key success factors of such projects is the gathering of requirements as the software bases on them. In agile projects, scenarios are an appropriate way to describe a system from the user's point of view. However, it is not obvious how to specify the requirements so that they are of high quality. This article presents an enhancement of scenario-based requirements engineering techniques, so that the resulting requirements fulfill the quality characteristics of the international standard ISO/IEC/IEEE 29148. The requirements engineering technique has been created for the development of RESTful web services. For that reason, this article demonstrates its application by means of the Participation Service. Functional and non-functional requirements are elicited and constraints that emerged from the existing RESTful service-oriented architecture are considered.**

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

## I. INTRODUCTION

This article is an extended version of [1]. It describes the requirements engineering approach that has been applied for the Participation Service, a web service for systemic consenting more in detail. Furthermore, compared to the original work, it is shown that the approach is not necessarily limited to RESTful web services as REST is only a constraint in the methodology. Nevertheless, the focus is still on web services in a service-oriented architecture. The general applicability on all kind of software systems is possible, but not yet proven. This kind of applicability should be considered in future research work.

Decision-making is always a typical and recurring challenge in a society. When having a certain issue, stakeholders and participants have different opinions. They defend their points of view and try to convince the others of their personal opinion. To make a decision, consensuses have to be found that satisfy all stakeholders and participants.

At the Karlsruhe Institute of Technology (KIT) a new software service, the Participation Service, is expected to be developed that supports the finding of consensus. 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 for the entire group and all stakeholders and participants. For that purpose, possible solutions are scored with points. However, compared to usual decision-making processes, the solutions are not scored with agreement points but with refusing points. This means, after describing the issue and collecting possible solutions, the one solution is selected that has the fewest refusing points. This solution represents the one with minimum resistance.

The Participation Service is expected to be part of the already existing service-oriented campus system of the university. The so-called SmartCampus is a system that provides functionality for students to support their daily life. For example, today the SmartCampus offers functionality to find free workplaces or to determine the route to a certain destination, such as the library of the university. As the services of the SmartCampus are expected to be used by several different devices, such as notebooks, smartphones and tablets, the software services are developed as web services based on the Representational State Transfer (REST) paradigm [2] as lightweight alternative to technologies, such as SOAP over Hypertext Transfer Protocol (HTTP), Extensible Markup Language (XML), and Web Services Description Language (WSDL). The RESTful web services are invoked by a web application that is responsive and can be therefore used on all the required devices. Furthermore, the service is developed in an agile manner to rapidly receive feedback about its usability.

For successful software projects, one key success factor is the requirements engineering with its underlying process and methodology [3][4]. In this phase, the functional and non-functional requirements are gathered and described. They represent the basis for the entire software project. In agile projects, the usage of scenario has evolved as an appropriate way to describe the requirements. Scenarios represent the requirements from a user's point of view. As the entire software project bases on the requirements, their high quality is very important. For that reason, the IEEE has created a set of quality characteristics for requirements. They are summarized in the IEEE recommended practice for software requirements specifications IEEE Std 830-1998 [5] and its successor, the ISO/IEC/IEEE 29148 [6]. However, existing scenario-based requirements engineering methodologies do not consider these quality characteristics explicitly.

This article enhances existing requirements engineering methodologies for agile projects in a way that quality characteristics of the international standard ISO/IEC/IEEE 29148 [6] are considered. For that purpose, in a first step, existing methodologies are analyzed and described. In a next step, the most appropriate methodology is reused and adapted where necessary. In this phase, these parts of other methodologies that support the achievement of certain quality characteristics of the international standard are reused and combined with the chosen methodology. As result, a methodology is created that combines the best parts of all analyzed methodologies.

To illustrate the resulting methodology, the Participation Service for the SmartCampus as a real-world project is considered. Its requirements are gathered and described using the elaborated methodology. Based on this approach, in a first step, the stakeholders are identified. Afterwards, the goals of the Participation Service are elicited and prioritized. In the last step, the functional and non-functional requirements are formalized and it is shown that they fulfill the quality characteristics of ISO/IEC/IEEE 29148.

The article is structured as follows: Section II examines existing work in the context of requirements engineering methodologies and quality characteristics for requirements. Section III introduces the Participation Service as exemplarily scenario. In this context, the idea behind the service is described in detail. Our quality-oriented requirements engineering methodology is presented in Section IV. Section V concludes this article and introduces future research work in the context of a quality-oriented development of RESTful web services.

## 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 [7].

In IEEE Std 830-1998 [5], the IEEE offers an official recommended practice for software requirements specifications, which was replaced by the new international standard ISO/IEC/IEEE 29148 [6]. 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. [8] 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 into 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. [9] 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 [10] 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 ISO/IEC 25010:2011 [11].

For eliciting functional requirements, Rolland et al. [12] present a goal modeling approach by using scenarios. A goal represents something that the stakeholders want to achieve 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 ISO/IEC/IEEE 29148 [6], 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 [13] introduce some interview questions that can be used for identification of the initial goals. Furthermore, elicitation techniques can be found in [3]. 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 [14] 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 [15], a simple classification approach with three different scales based on IEEE Std 830-1998 [5] is best suited for the initial prioritization.

When writing scenarios, the quality characteristics by [6] have to be considered. Glinz [16] presents an approach, which respects the quality characteristics by the old recommendation IEEE Std 830-1998 [5]. His findings will be used to improve the quality of requirements.

Also, Terzakis [17] 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 [11], 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. [18] 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 [11] can be derived. However, for many quality characteristics it can be very time-consuming.

To reduce the effort, the decision-making approach by Saaty [19] will be applied by using pairwise comparison of the quality characteristics in ISO/IEC 25010:2011 [11] 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 [10] will be considered.

## III. Scenario

To illustrate the requirements engineering approach, the SmartCampus System at KIT is to be enhanced by a new service, the Participation Service. The SmartCampus system is a service-oriented system to support professors, students, and other KIT members in their daily life. For example, the SmartCampus system already provides services to determine the route to a certain room or to find free workplaces.

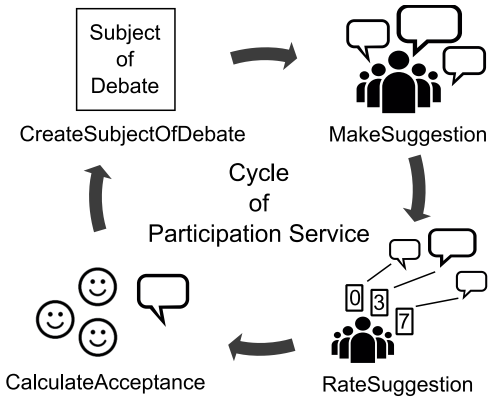

Figure 1. Systemic consenting process.

The services of the SmartCampus can be used by means of web applications that can be also used on mobile devices, such as smartphones and tables. For that reason, the web applications are developed with a responsive layout using modern and standardized web technologies, such as Hypertext Markup Language (HTML) 5.

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).

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 of RESTful Web Service for Systemtic Consenting

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 [6]. This can be proven to the customer. First, the quality characteristics of the standards IEEE Std 830-1998 [5] and ISO/IEC/IEEE 29148 [6] 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 and non-functional requirements are discovered and documented according to the derived quality characteristics of [6] and the provided taxonomy by Glinz [10]. Finally, the elicited requirements for iteration were verified according to specific quality characteristics in [6]. The entire requirements engineering methodology is shown in Fig. 2.

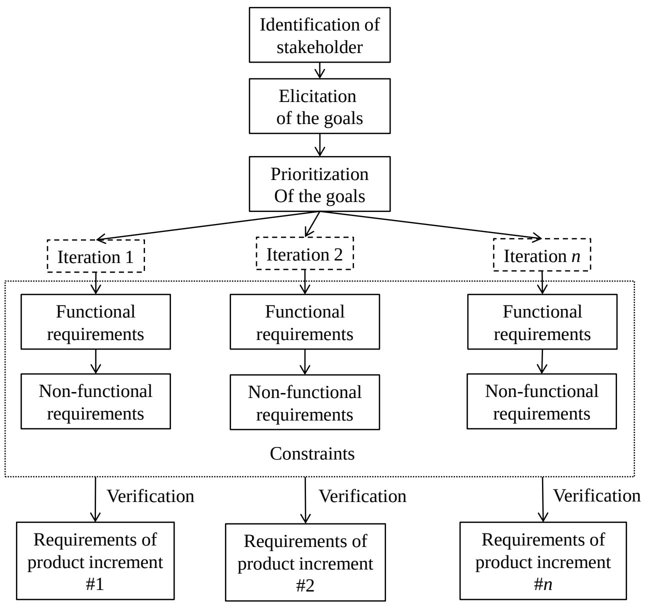
Identification of stakeholder
Elicitation of the goals
Prioritization Of the goals
Iteration 1
Iteration 2
Iteration n
Functional requirements
Non-functional requirements
Constraints
Verification
Requirements of product increment #1
Requirements of product increment #2
Requirements of product increment #n

Figure 2. Requirements engineering methodology for agile development of RESTful web services.

### A. *Quality Characteristics for Requirements*

According to IEEE Std 830-1998 [5], the requirements quality focuses on correctness, unambiguousness, completeness, consistence, prioritization, verifiability, modifiability, and traceability. The IEEE Std 830-1998 [5] was replaced by the international standard ISO/IEC/IEEE 29148 [6], 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. The full set of quality characteristics with its definition is shown in Tables I and II [6].

TABLE I. QUALITY CHARACTERISTICS FOR INDIVIDUAL REQUIREMENTS

| Quality Characteristic | Definition |
|---|---|
| Necessary | "The requirement defines an essential capability, characteristic, constraint, and/or quality factor. If it is removed or deleted, a deficiency will exist, which cannot be fulfilled by other capabilities of the product or process…" [6] |
| Implementation free | "The requirement, while addressing what is necessary and sufficient in the system, avoids placing unnecessary constraints on the architectural design…" [6] |
| Unambiguous | "The requirement is stated in such a way so that it can be interpreted in only one way. The requirement is stated simply and is easy to understand." [6] |
| Consistent | "The requirement is free of conflicts with other requirements." [6] |
| Complete | "The stated requirement needs no further amplification because it is measurable and sufficiently describes the capability and characteristics to meet the stakeholder's need." [6] |
| Singular | "The requirement statement includes only one requirement with no use of conjunctions." [6] |
| Feasible | "The requirement is technically achievable, does not require major technology advances, and fits within system constraints (e.g., cost, schedule, technical, legal, regulatory) with acceptable risk." [6] |
| Traceable | "The requirement is upwards traceable to specific documented stakeholder statement(s) of need… The requirement is also downwards traceable to the specific requirements in the lower tier requirements specification or…" [6] |
| Verifiable | "The requirement has the means to prove that the system satisfies the specified requirement. Evidence may be collected that proves that the system can satisfy the specified requirement…" [6] |

In [1], we took the assumption that the full set of quality characteristics can be fulfilled by ensuring the individual ones. But, this is not true for the full set of quality characteristics since a complete requirement does not provide information about the completeness of a set of requirements.

TABLE II. QUALITY CHARACTERISTICS FOR A SET OF REQUIREMENTS

| Quality Characteristic | Definition |
|---|---|
| Complete | "The set of requirements needs no further amplification because it contains everything pertinent to the definition of the system or system element being specified." [6] |
| Consistent | "The set of requirements does not have individual requirements which are contradictory. Requirements are not duplicated. The same term is used for the same item in all requirements." [6] |
| Affordable | "The complete set of requirements can be satisfied by a solution that is obtainable/feasible within life cycle constraints (e.g., cost, schedule, technical, legal, regulatory)." [6] |
| Bounded | "The set of requirements maintains the identified scope for the intended solution without increasing beyond what is needed to satisfy user needs." [6] |

Due to that, we formalized the quality characteristics in Table II in a way that it can be applied on a set of requirements for easier quality control at the end of a requirements engineering phase. The formalization for each quality characteristic is shown in Equations (1)-(4), while Table III will give the explanation of the used elements. The necessary information for the interpretation of the results will be given in Table IV.

$$\mathrm{COM}(\mathrm{R_d}) = \frac{\mathrm{R_d} \cap \mathrm{R_s}}{\mathrm{R_s}} \text{ if } |\mathrm{R_s}| > 0 \text{ else } 1 \quad (1)$$

$$CON_1(\mathrm{R_d}) = 1 - \frac{|R(R_d)|}{|\mathrm{R_d}|} \text{ if } |\mathrm{R}_d| > 0 \text{ else } 1 \quad (2)$$

$$CON_2(\mathrm{R_d}) = 1 - \frac{|C(R_d)|}{|\mathrm{R_d}|} \text{ if } |\mathrm{R}_d| > 0 \text{ else } 1$$

$$CON_3(\mathrm{R_d}) = 1 - \frac{|T(R_d)|}{|\mathrm{R_d}|} \text{ if } |\mathrm{R}_d| > 0 \text{ else } 1$$

$$CON_1(\mathrm{R_d}) = \frac{1}{3} * \left(CON_1(\mathrm{R_d}) + CON_2(\mathrm{R_d}) + CON_3(\mathrm{R_d})\right)$$

$$\mathrm{AFF}(\mathrm{R_d}) = \frac{|A(R_d)|}{|\mathrm{R_d}|} \text{ if } |\mathrm{R_d}| > 0 \text{ else } 1 \quad (3)$$

$$\mathrm{BOU}(\mathrm{R_d}) = \frac{|R_d \setminus R_s|}{|\mathrm{R_d}|} \text{ if } |\mathrm{R_d}| > 0 \text{ else } 1 \quad (4)$$

TABLE III. EXPLANATION OF THE METRICS

| Element | Explanation |
|---|---|
| $R_d$ | Set of requirements, which should be considered |
| $R_s$ | Not absolutely necessary right now |
| $A(R_d)$ | Set of feasible requirements |
| $C(R_d)$ | Set of requirements with conflicts |
| $R(R_d)$ | Set of duplicated requirements |
| $T(R_d)$ | Set of requirements in which introduced terms are not used consistently |

TABLE IV. EXPLANATION OF THE RESULTS

| Result | Explanation |
|---|---|
| 1 | The quality characteristic is completely fulfilled |
| < 1 | The quality characteristic is not completely fulfilled |

## 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. [8]. 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 V and assign them to the corresponding scrum role.

TABLE V. 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 |

The user represents people, groups, or organizations, which interact with the system or make use of the provided information. The developers are the stakeholders of the requirement engineering process, such as analysts or operators. The legislators represent government authorities that provide guidelines for the development and operation of the Participation Service. The last group stands for the development manager and the user manager, who have the power to make decisions with regard to the characteristics of the system in development.

Depending on the quantity of the stakeholders, a prioritization step is sometimes necessary to assess the importance of the elicited requirements regarding to the influence of the stakeholder. For this reason, Sharp et al. [8] provide an outlook how network theories can be used to determine the influence of a stakeholder. But, such approaches can be time-consuming. A more pragmatic method is the usage of power-interest grid by which the stakeholders are classified in quadrants [22].

In this project, 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.

## 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 [13] were used for easier discovery of the goals according to the definition by [12], which is shown in Fig. 3. Each goal corresponds exactly to one requirement in order to fulfill the singularity according to [6]. An excerpt of the determined goals is shown in Table VI. Goal G2 will be further refined in the upcoming sections.

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.

TABLE VI. 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 |

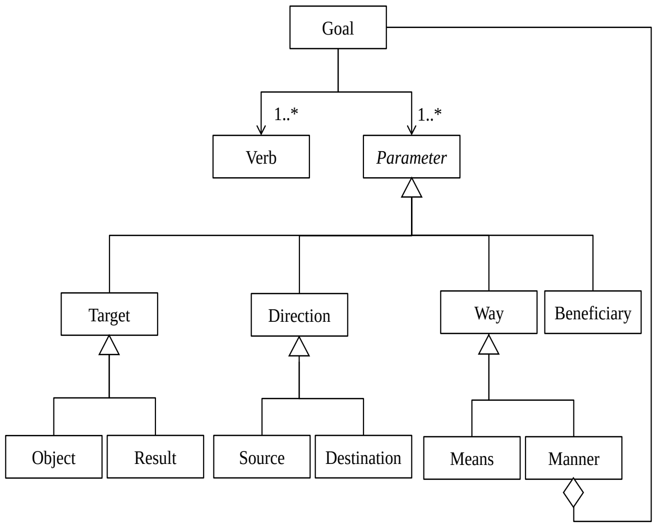

Figure 3. Meta-model of a goal.

By investigating the quality characteristic of the current standard [6], we discovered that the meaning was changed compared to IEEE Std 830-1998 [5]. In [5], 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 [15], we applied a simple classification approach based on a three-level scale that is shown in Table VII according to IEEE Std 830-1998 [5]. In order to prevent ambiguousness, each stakeholder has agreed on the meaning of each level [15]. After rating of goals, a specific amount of highest ranked goals, which reflects the necessity [6], 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 VI.

TABLE VII. CLASSIFICATION FOR GOAL PRIORITIZATION

| Group | Meaning |
|---|---|
| Essential | Essential for the next release |
| Desireable | Not absolutely necessary right now |
| Optional | Would be nice to have someday |

### 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 [12].

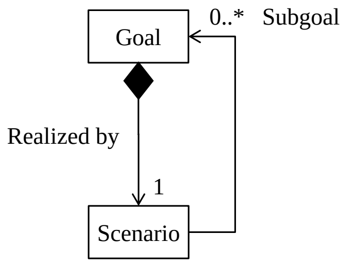

Figure 4. Meta-model of a requirement chunk.

Fig. 4 illustrates this by showing a meta-model that defines the rules and the elements of a requirement chunk. This recursive process with objective of functional decomposition can be aligned with the process defined in the standard [6]. 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 [16], 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.

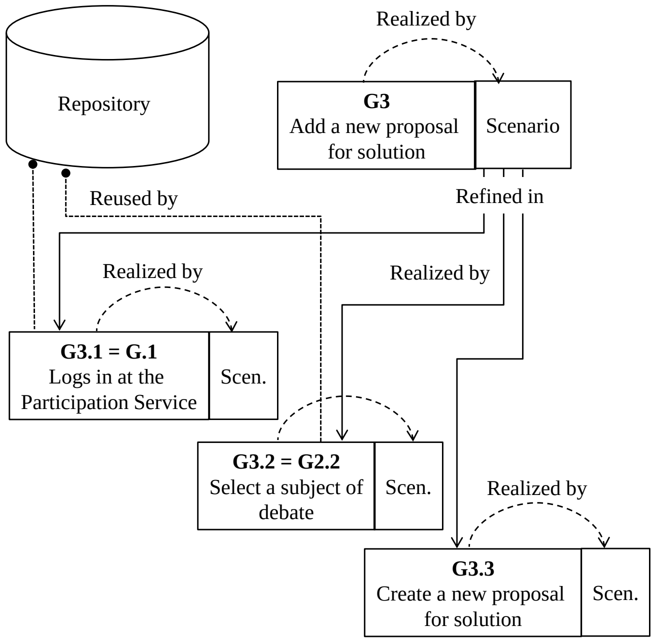
Repository
Realized by
G3
Add a new proposal for solution
Scenario
Reused by
Refined in
Realized by
Realized by
G3.1 = G.1
Logs in at the Participation Service
Scen.
G3.2 = G2.2
Select a subject of debate
Scen.
Realized by
G3.3
Create a new proposal for solution
Scen.

Figure 5. Reusing a requirements chunk from the repository.

### *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*. Fig. 5 shows how the goal *G3* is refined in three different sub goals, while two of them will be reused from the repository.

### *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 [6].

The unambiguousness cannot be fulfilled properly as we use the natural language with inherent equivocality for the description of the scenario [5]. 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. [12] to reduce equivocality, which is shown in Fig. 6 Moreover, we used the introduced structural constructs of Glinz [16] to further reduce the level of equivocality. To detect ambiguousness during description or validation of scenarios, Terzakis [17] offers a detailed checklist. Also, the current standard [6] 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.

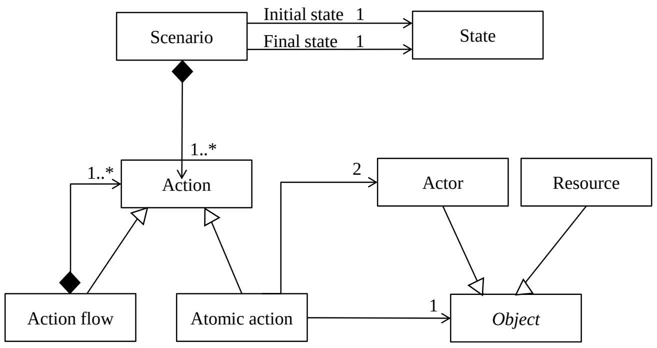

Figure 6. Meta-model of a functional scenario.

According to [6], 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 [16], which fulfills the verifiability [6].

The feasibility is another quality characteristic of the standard [6] 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).

To ensure the traceability [6], 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.

| | | | | | |
|---|---|---|---|---|---|
| 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: | None |

| No. | Normal action flow | Ref. |
|---|---|---|
| 1 | User logs in at the Participation service<br>System verifies the credentials | G1 |
| 2 | System redirects him to the secured area (Def. 1.1)<br>User gets a list of available subjects of debate | - |
| 3 | User selects a subject from the provided list<br>System receives the selection and redirects him to the subject of debate | - |
| 4 | User rates a proposed solution by selecting the refusing points<br>System calculates the acceptance of the suggested solution | G5 |
| No. | Concurrency / 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 7. Style for representation of scenarios.

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 [16], and the provided requirement attributes in [6], we created a style for representation of scenarios, which is illustrated in Fig. 7. Similar to the approach by Glinz [16], 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. [18] and link these with the corresponding requirement chunk. The meta model for quality attribute scenario is shown in Fig. 8.

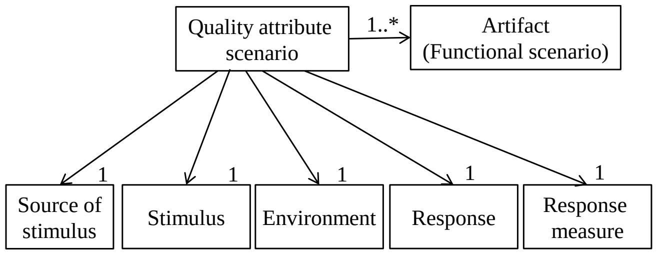

Figure 8. Meta-model of quality attribute scenario.

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 of ISO/IEC 25010:2011 [11] 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 banking sector. This is why we used pairwise comparisons of the quality attributes according to the Analytical Hierarchy Process (AHP) by Saaty [19].

If quality characteristic *A* is more important_than *B*, we assign *A* the value 2 and *B* the value 0. If *A* and *B* are equally important, we assign each of them the value 1.

We took the results of each stakeholder and calculated the average, which is shown in Fig. 9. As Fig. 9 shows, 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.

| | Functional suitability | Performance efficiency | Usability | Compatibility | Reliability | Security | Maintainability | Portability | *Sum* | *Weight* |
|---|---|---|---|---|---|---|---|---|---|---|
| Functional suitability | | 11 | 6 | 10 | 7 | 6 | 9 | 11 | 60 | 0.18 |
| Performance efficiency | 1 | | 0 | 4 | 2 | 0 | 2 | 7 | 16 | 0.04 |
| Usability | 6 | 12 | | 10 | 8 | 3 | 9 | 11 | 59 | 0.18 |
| Compatibility | 2 | 8 | 2 | | 4 | 2 | 3 | 8 | 29 | 0.09 |
| Reliability | 5 | 10 | 4 | 8 | | 3 | 5 | 7 | 42 | 0.13 |
| Security | 6 | 12 | 9 | 10 | 9 | | 9 | 12 | 67 | 0.20 |
| Maintainability | 3 | 10 | 3 | 9 | 7 | 3 | | 10 | 45 | 0.13 |
| Portability | 1 | 5 | 1 | 4 | 5 | 0 | 2 | | 18 | 0.05 |
| Sum | | | | | | | | | 336 | 1.0 |

Figure 9. Results of the Analytical Hierarchical Process (AHP).

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 [15] or [18]. Fig. 10 shows one non-functional requirement of goal *G2*.

| | | | | | |
|---|---|---|---|---|---|
| Type: | Usability | ID: | N2 | Priority: | 0.18 |
| Source: | C&M member, students | Risk: | Low | Difficulty: | Easy |
| Rationale: | Better user experience | Version: | 1.0 | Reference: | 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 10. Style for representation of quality attribute scenarios.

### *G. Constraints*

According to Glinz [10], 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 [10]. 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 SmartCampus System based on REST.

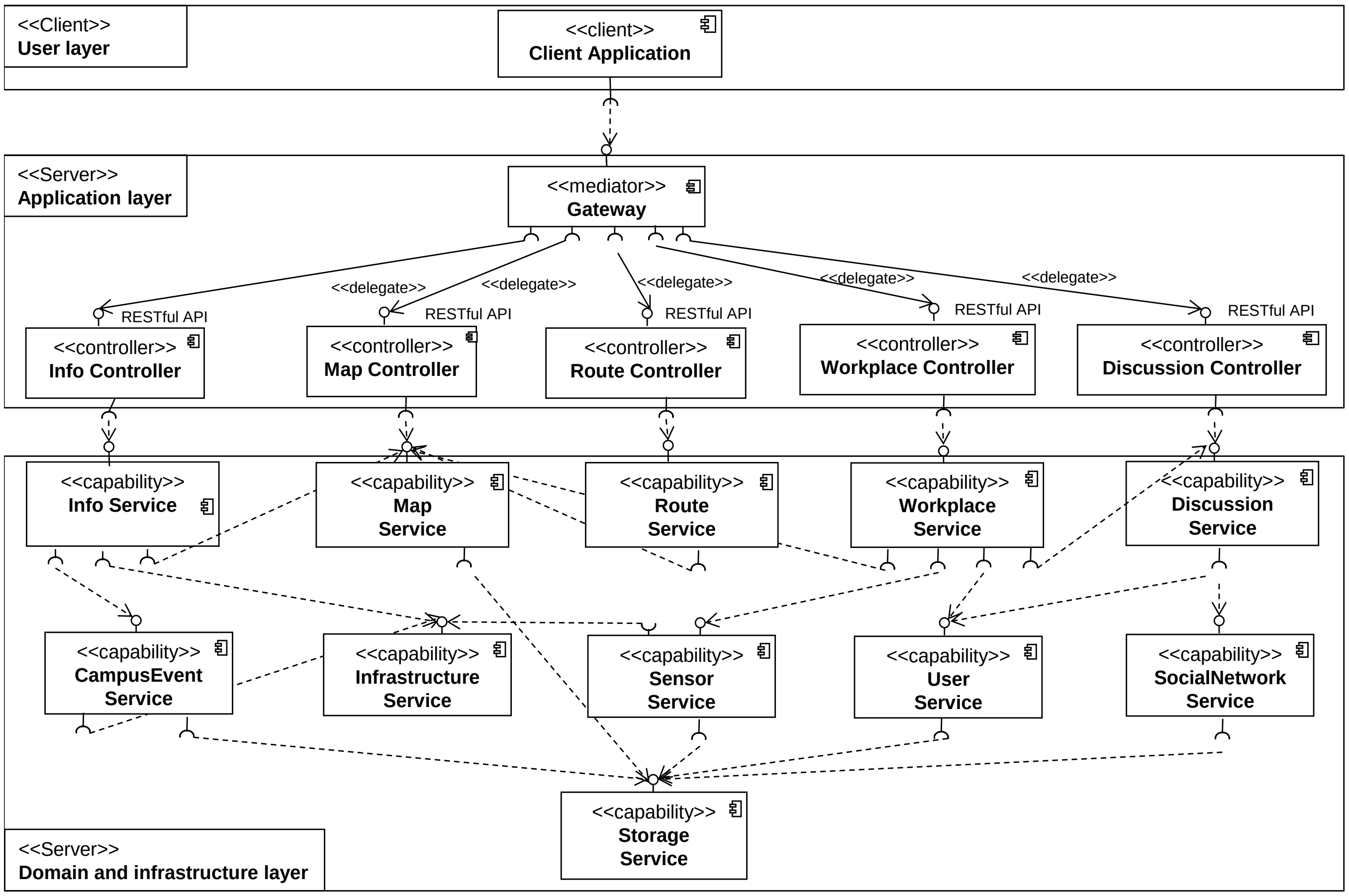

Figure 11. Component diagram of the SmartCampus system at the KIT.

Fig 11 shows the layered architecture according to Evans [23] with components of the current SmartCampus System, which consists of four layers: 1) user layer, 2) application layer, 3) domain, and 4) infrastructure layer. The latter ones are combined in the illustration for a better overview.

REST is a hybrid architectural style for distributed hypermedia systems according to Fielding [2], which he defines as follows: "REST is a hybrid style derived from several of the network-based architectural styles ... and combined with additional constraints that define a uniform connector interface." [1, p. 76]. This definition implies the consideration of several constraints that can be segmented in architectural (1 - 5) and interface constraints (6) [1][21]:

1) Client-Server indicates a client and server component. The client component sends a request to the server that should be performed. Based on the request, the server component either rejects or performs the request.
2) Statelessness avoids the need of maintaining information about a previous request on server side. This leads to an improvement of server scalability.
3) Caching avoids a replication of already transmitted information over the network.
4) Layered architecture facilitates the usage of mediator components for adding features such as load-balancing.
5) Code on demand is an optional constraint, which extends the client functionality at runtime trough downloading an executable artifact.
6) Uniform interface is an "umbrella term for the four interface constraints" [21, p. 356]: the identification of resources, the manipulation of resources through representation, the self-descriptive messages und the hypermedia constraint.

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 [6].

## H. Verification

After the elicitation of the requirements in a quality-oriented way, we have investigated the requirements according to the formalized characteristics for a set of requirements in Section IV. These results give us a hint to what extent the elicited requirements fulfill the quality characteristics of ISO/IEC/IEEE 29148:2011 [6].

TABLE VIII. RESULTS OF THE VERIFIED SET OF REQUIREMENTS IN EACH PERFORMED ITERATION

| Metric | Iteration #1 | Iteration #2 | Iteration #3 | Iteration #4 |
|---|---|---|---|---|
| COM($R_d$) | 0,97 | 0,92 | 0,93 | 0,99 |
| CON($R_d$) | 1,0 | 1,0 | 1,0 | 1,0 |
| AFF($R_d$) | 1,0 | 1,0 | 1,0 | 1,0 |
| BOU($R_d$) | 1,0 | 1,0 | 1,0 | 1,0 |

Based on the results in Table VIII, we could prove our assumption that the full set of quality characteristics can be fulfilled by ensuring the individual ones. The only exception is the completeness, which was already mentioned in Section IV. Because of this, we recommend the investigation of the completeness before designing and implementing the specified system or system element to ascertain the quality of the requirements.

## V. EVALUATION

Our results by applying this technique showed us that we improved the quality of our requirements by using this technique, which considers the quality characteristic of ISO/IEC/IEEE 29148:2011 [6]. For example, we have detected some inconsistencies during the authoring of the scenarios and reduced the communication effort emerged from misunderstandings.

Compared to the previous recommendation [5], it is easier to meet the desired qualities of ISO/IEC/IEEE 29148:2011 [6]. 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 we are using the natural language for describing requirements, we can only merely reduce the ambiguousness and not prevent completely. However, this does not imply bad requirements but rather potential for improvements. Furthermore, sometimes it is adequate to achieve 90 percent of the quality criteria, because the cost to reach 100 percent is too high.

Furthermore, we propose the adjustment of the completeness so that partial specifications in form of iterations are allowed. The precondition of the completeness will be analyzed with regard to the goals of the current iteration.

## VI. CONCLUSION AND OUTLOOK

In this article, we introduced a methodology for requirements engineering of RESTful web service for systemic consenting. The methodology ensures that the requirements fulfill quality characteristics defined by the international standard ISO/IEC/IEEE 29148. For that purpose, we analyzed existing methodologies and combined those parts that consider a certain quality characteristic to a new methodology. Thus, the methodology presented in this article is a combination of existing work.

As stakeholders and participants often have different opinions, it is necessary to find consensuses. For that purpose, the Participation Service implements functionality that is based on the concept of systemic consenting. By applying the requirements engineering methodology presented in this article, the quality of the requirements for the Participation Service could be improved. 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 Std 830-1998 [5], it is easier to meet the desired qualities of ISO/IEC/IEEE 29148 [6]. 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 is currently focused on the Participation Service and its specifics. We assume that the methodology is also applicable for further services or even software systems in general. However, this is not proven yet. With this approach, we expect to support requirements engineers and business analysts when they have to describe the requirements for a RESTful web service. In our scenario, the presented methodology helped with gathering and describing functional and non-functional requirements in a systematic way so that they are of high quality. As the quality characteristics considered in this article are part of an international standard, they can be seen as valid and of importance. Furthermore, requirements engineers and business analysts can apply this methodology to analyze and improve already described requirements regarding their quality. As the requirements constitute the basis for the rest of the development process, it is of high importance that a certain level of quality is reached. For that reason, when generalizing this approach, it will contribute to the development of high-quality software solutions.

For the future, before generalizing the approach, we plan to focus on further parts of the development of high-quality RESTful web services. With this article, we considered the initial phase of the development process, the gathering and description of requirements. In the next step, we will focus on the design of RESTful web services that fulfill the previously gathered requirements. Also in this case, the quality of the result will be considered. For that purpose, we will analyze existing best practices for the design of RESTful web services. We will combine these best practices with quality characteristics of ISO 25010:2011 as a standard for the quality for software products. Especially in environments with limited resources, such as time and money, not all best practices can be considered. By associating best practices with quality characteristics, it will be possible to prioritize best practices for the design of RESTful web services and to select the for a certain project most valuable ones. Finally, we aim to enable an automatic measurement of the best practices to rapidly get an impression of the degree of fulfillment.

For that purpose, we will enhance our existing work in the context of quality assurance of service-oriented architectures [20]. We are also already working on an open source tool, the QA82 Analyzer, to automate the measurement of best practices [24]. After focusing on the requirements engineering, the future work will help us to also design and develop the Participation Service and future web services in a quality-oriented manner.

## REFERENCES

[1] M. Gebhart, P. Giessler, P. Burkhardt, and S. Abeck, "Quality-oriented requirements engineering for agile development of restful participation service," Ninth International Conference on Software Engineering Advances (ICSEA 2014), Nice, France, October 2014, pp. 69-74.

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

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

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

[5] IEEE, IEEE Std 830-1998 "Recommended practice for software requirements specifications," 1998.

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

[7] 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.

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

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

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

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

[12] 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.

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

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

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

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

[17] 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.

[18] 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.

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

[20] 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.

[21] L. Richardson, M. Amundsen, S. Ruby "RESTful Web APIs," O'Reilly, 2013.

[22] F. Ackermann, C. Eden "Strategic Management of Stakeholders: Theory and Practice," Long Range Planning, Volume 44, No. 3, 2011, pp. 179-196.

[23] E. Evans, "Domain-Driven Design: Tacking Complexity In the Heart of Software," Addison-Wesley Longman Publishing Co., Inc., 2003.

[24] QA82, QA82 Analyzer, http://www.qa82.org, Accessed 2015-02-12.