Special Issue Published in International Journal of Trend in Research and Development (IJTRD), ISSN: 2394-9333, www.ijtrd.com

Documenting Software Requirements Specification – A View-based Approach

[1]Huda Alomair and [2]Pr. Azeddine CHIKH,
[1]King Saud University, KSA
[2]University of Tlemcen – Algeria

Abstract: Software Requirements Engineering (SRE) consists of five main activities: Elicitation, Analysis and Negotiation, Documentation, Validation, and Management. We are interested in the present work to the documentation activity in general and more precisely to software requirements specification (SRS). Our hypothesis is that a view-based mechanism makes the SRS more convenient and adaptable for its readers represented by the different software stakeholders. Indeed, in our proposed approach of SRS documentation, views are organized through three different layers.A requirements document view displays a list of written requirements by dividing them into packages in order to avoid manipulating the entire set of requirements; to provide views on particular logical subsets of the requirements; and to follow some users/readers preferences such as the zoom level.Our main contribution will be mainly a conceptual level framework focusing on the mechanism of constructing views than the views themselves.

Keywords: *SRS; SRE; documentation;*

I. INTRODUCTION

Requirements are descriptions of how a software product should perform. A requirement typically refers to some aspect of a new or enhanced product or service. A requirement also is a collection of needs arising from the user and various other stakeholders all of which must be met [2].Requirements have many categories: data requirements; functional requirements; and non-functional requirements [1] [3]. Requirement engineering process consists of five main activities: elicitation, analysis and negotiation, documentation, validation, and management. We will concentrate on the requirements documentation activity in general and more precisely to building the Software Requirements Specification (SRS).

II. PROBLEM IDENTIFICATION AND BASIC PRINCIPLE

The purpose of SRS is to communicate requirements between stakeholders and developers. The SRS is the baseline for evaluating subsequent products and processes (design, testing, verification and validation activities) and for change control. A good SRS is unambiguous, complete, correct, understandable, consistent, concise, and feasible [4] [5]. SRS is basically an organization's understanding of a customer or potential client's system requirements and dependencies at a particular point in time usually before any actual design or development work [6]. There are many ways of combining requirements and auxiliary information into a specification. The best-known standard in the area is IEEE Recommended Practice for Software Requirements Specification (IEEE 830)[6],[7].

Many researchers work on many directions in order to improve different aspects of SRS (Content integrity; Formal representation; Resolving requirements conflict; …). Authors in [9] propose a method for simplifying ambiguity of requirement specification documents through two concepts of ontology-based probabilistic text processing: text classification and text filtering.Authors in [2] argue that it is no longer appropriate for SRS to focus only on functional and non-functional aspects of the intended system. Instead, they call for a broader perspective in order to gain a better understanding of the interdependencies between enterprise stakeholders, processes, and software systems, which would in turn give rise to higher-quality systems. Many publications related to SRS address problems and provide solutions for solving them. The most frequently researched SRS problems and improvement methods are listed and referenced in a mapping study [11]. As final result of the search process, publications are analyzed and mapped to each other.

Finally authors in [12] argue that SRS serves as a source of communication and information for a variety of roles involved in downstream activities like architecture, design, and testing. So in order to create high-quality requirements specifications that fit the specific demands of successive document stakeholders, we need to better understand the particular information needs of downstream development roles. The authors introduce the idea of view-based requirements specifications. Two scenarios illustrate (1) current problems and challenges related to the research underlying the envisioned idea and (2) how these problems could be solved in the future. Based on these scenarios, challenges and research questions are outlined and supplemented with current results of exemplary user studies.

We are interested in the present work to develop further the concept view-based requirements specifications proposed in [12]. Our hypothesis is that a view based mechanism makes the SRS more convenient and adaptable for its readers represented by the different software stakeholders. Indeed, in our proposed approach of SRS document, views are organized through three different layers

An SRS view displays a subset of requirements adapted to a given stakeholder's profile and responding to his: (1) reading needs (such as items or aspects of SRS to read); and (2) reading preferences (such as the level of details, the reading language, the requirement style, etc.).

Hence we aim to help stakeholders to read adaptive SRS views that are suitable for theirs profiles and their needs, offering them a better understanding and allowing them saving their time from reading useless details.

These views make the knowledge more explicit and facilitate operations such as searching, browsing, and comparison with other views.

III. METHODOLOGY

We solve in this paper the problem of complexity of the SRS document by adopting a view-based approach.

A view is an abstraction of a system that deliberately

focuses on some aspects of this system when excluding others. The benefit of this is that smaller amounts of related information can be collected, processed, organized and analyzed, applying various specific techniques pertinent to the aspects under study.

Where a large amount of complex information has to be managed, views provide a means of zooming in, collecting together subsets of the data for a particular purpose and zooming out once more to appreciate the whole. It aids in maintaining a system-wide grasp through focusing on small amounts of information at a time.

By applying the general concept of view to the SRS, we get a more specific concept of SRS view.

Accordingly, the previous general definition of view could be specialized and customized to fit the new specific concept of SRS view as following:

An SRS view is an abstraction of an SRS document that:

1.deliberately focuses on some aspects of the SRS, that are under consideration (selected filtering attributes) such as the category of requirements (functional, non-functional requirements); the level of requirements (business, domain, product or design requirements); etc.; or a combination of, when excluding the rest of SRS (non-selected filtering attributes);

2.is adapted to a given category of stakeholders (Customers views, Managers views, Developers views, …);

3.is personalized to stakeholders' preferences: the preferred zoom level; the preferred sorting criteria.

Figure 1 below illustrates the three layers of the SRS that respectively emerge from the three levels of selection of the definition above. The SRS content layer of the SRS view corresponds to the aspects of SRS that are under consideration. The Stakeholder category layer of the SRS view corresponds to a given category of stakeholders.The Stakeholders' preferences layer of the SRS view corresponds to stakeholders' preferences.

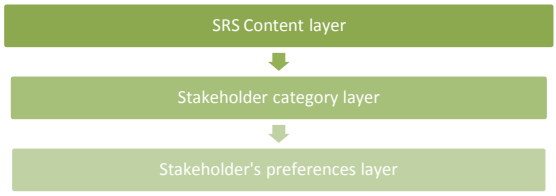

Fig.1 multi-layer view for SRS

A. *SRS content layer*

focuses on some aspects of SRS, that are under consideration (selected filtering attributes) such as the category of requirements (functional, non-functional requirements); the level of requirements (business, domain, product or design requirements); etc.; or a combination of, when excluding the rest of SRS (non-selected filtering attributes).

Categories of requirements are:

Data requirement: Data requirements is an important part of the requirements it contain what data should the system input and output, and what data should the system store internally

Functional requirement: Functional requirements specify what data is to be used for, how it is recorded, computed, transformed, updated, transmitted, etc. Functional requirements 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, And it May state what the system should not do[13].

Non-Functional requirement: Are the constraints on the services or functions offered by the system such as timing constraints, constraints on the development process, standards, etc. The restrictions on time and standards would assist in speeding up in reaching the goal for each agent

Managerial Requirement:contains information about deliverables like when they will be delivered, the price and when to pay it, how to check that everything is working (verification),what happens if things go wrong (legal responsibilities, penalties, etc.. In addition, it specifies who owns the software and other intellectual properties. Also it includes a specification of the development process to be used.

Levels of requirements are:

Goal-level requirement: This requirement states the business goal. They call it a goal-level requirement because it is a business goal that can be verified, although only after some period of operation. Unfortunately, if you ask a software house to accept this requirement, they will refuse [3].

Domain-level requirement: It outlines the tasks involved and requires support for these tasks. The analyst has carefully identified the right tasks. This model primarily focuses on describing the user tasks in co-operation with expert users. Also collect information on the data to be stored in the computer.

Product-level requirement: Where we specify what comes in and goes out of the product. Essentially we just identify the function or feature without giving all the details, it is this classical model interviews the stakeholders, study existing documents, conduct workshops and Brainstorms. Finally, you analyze and write the requirements specification

Design-level requirement: Where we specify one of the product interfaces in detail. Although a design-level requirement specifies the interface exactly, it doesn't show how to implement it inside the product [3].

B. *Stakeholder layer*: an SRS view is designed for each category of stakeholder (users, customer, manager, analyst, designer, developer, tester…)

Stakeholders are Individuals or organizations who stand to gain or lose from the success or failure of a system. Stakeholders include customers or clients (who pay for the system), developers (who design, construct and maintain the system), and users (who interact with the system to get their work done).Also Stakeholders defined as everyone who affect or are affected by the software product [3] [16].Every stakeholder type looks for some criteria, properties and categories and styles that satisfy his needs. We can collect this information by interviewing them, the view should give every stakeholder the exact requirements he looks for and he can get benefit from, of course this needs a deep study and a huge research work.

In our hypothesis, after views of layer 1 are defined, we can later in the layer 2 adapt their content to match stakeholders' categories.We assume that on this layer the views will be produced and generated by need. We assume that there will be a predefined view for every type of stakeholders, and there is an extra view which is a default view in which every stakeholder can choose because it has a default properties

suitable for everyone.

We will discuss some of the stakeholder's types and we will try to suggest some criteria for the requirements' view they need.

Customers: Customers are those who buy the product.If the customer is looking for requirements. the view should satisfy certain properties for the desired requirement because the customer will check that requirements meet his needs. Then we will apply the third level (adapted level). Most of the time the customer has little technical information, so the view, that we will show him, should concentrate on the user requirements and not on the system requirements. In addition, the requirements should be in a high abstract level with some diagrams (like context diagram) that are easy to understand by a normal customer.

End-users: Users are the people who interact with the system to get their work done.If the End-users are looking for requirements, the view should satisfy the certain properties for the requirements they want to look at because the End-users will read the requirements to check that they meet their real demands, for example they read the requirements in certain view then they work with specialists or developers for enrichment.The Users, most of the time, have little technical information so the view we will show them should concentrate on the user requirement not the system requirements, and the requirements should be in a high abstract level with some diagrams that easy to understand by normal Users like context diagram or Prototype. The view should contain information about the complete functional features of the systems and final look and design.

Analysts, architects, designers, developers and testers maintenance engineers: They are developers who design, construct, test and maintain the system, their technical knowledge is high so the view will be produced after third layer and show a high detailed system requirements with technical styles that could be understood by the experts only, and every one of the developers will get his own specific view with the criteria that meet his demand. For example, the requirement view for testers should help them to validate test for the system, requirements view for engineers will help them to understand the system and the relationships between its parts.

As we mentioned above after applying the second layer and choosing the stakeholder this will led to the third layer in which we apply the dimensions of that layer based on the stakeholder choices about the preferred style of requirements, the exact order and the zooming level of the requirements document as we will show next.

C. *Stakeholder's preferences layer*: the previous views are adapted according the stakeholders' preferences (zoom level, sorting criteria)

After 2nd layer we will apply a third layer of classification which gives Frequency of views for every stakeholder based on individual preferences according the two Parameters, the preferred zoom level; the preferred sorting criteria; etc.

Zoom level: which defines the level of details of the requirements. It might have three different values: abstract, normal, detailed

The preferred sorting orders: The stakeholder can choose to sort the requirement according requirement number, or requirement importance, or requirement name.

IV. RESULTS AND DISCUSSIONS

Case study: Library system

This case study has complete requirements about a Library system. It is a good case study that will be considered in the upcoming next parts of this report. It was used in Research Seminar called " Engineering of Software Requirements Specification" by Abeer AlSanad and Supervised by Dr. Azeddine Chikh [14] this research work main goal was to improve the model of SRS of IEEE 830 standard. In addition, it aims to represent Software Requirement Specification (SRS) of IEEE 830 standard as semi-structured data using XML , and we will use it to illustrate our conceptual level of our view mechanism, the first layer of our mechanism will be using the XML schema produced on the Research Seminar " Engineering of Software Requirements Specification", then we will continue to apply the next two layers and we will show the proposed view.

Case study overview

SCOPE: Library Management system of the University of Ballarat.

INTRODUCTION

The University of Ballarat has various campuses distributed across Australia. It intends to build an online Library Management System. This project is basically updating the manual library system into internet based application so that the users can know the information of their account, availability of books, etc. This system is being developed by Innovative Library Management Solutions team based on the requirements illustrated in [17].

The document in [17] gives the detailed description of both functional, non-functional requirements and some constraints developed after a number of consultations with the client.

Our hypothesis has three layers we will apply every layer on the case study after the engineering:

layer 1: SRS content layer

We suppose that some typical views are defined at this layer one view deals with one single aspect. Aspects could be different criteria such as types of requirements or requirement level (business, domain, product, design req), or combination of two criteria.We will consider the SRS from the project we mention before" Engineering of Software Requirements Specification" [14], to represent the requirement documentation that will be produced from this first layer with the combination of tow criteria we choose (Functional Requirements + Product level). The view generated from this first layer will contain all the functional requirements in the product level only.

layer2: Stakeholder layer

On this layer the view produced from the first layer will be adapted to sub views with the predefined template for every stakeholder or the default view.

We will suppose that the stakeholder is: a developer or a customer the sub view related to the customer will contain user requirements and the sub view related to the developer will contain system requirements.Then we apply the 3rd layer to get the finalized view.

layer3: Preferences layer (adaptive)

After 2nd layer we will apply a third layer which gives Frequency of Preferences to personalize the requirements for the stakeholders based on some criteria (Zoom level, sorting order), we produce it for them as a choice to get a flexible look for the views as they need. We assume that the customer chooses the zoom level to be abstract and the sorting order based on requirement number.The developer chooses the zoom level to be detailed and the sorting order based on requirement name. two sub views will be produced from this layer for the two stakeholders with their preferences of the zooming level and the sorting order.

Finally, we want to mention that we apply our conceptual level on the previous case study after the engineering of the SRS and we focus on the mechanism of constructing views than the exacted view itself.

The logical level

Here we will suggest the logical representation of the three layers views mechanism:

SRS content layer:Based on project type and system type,

we suggest building an XML schema for every aspect that could be in different criteria (the requirements type and the requirements level or a combination of the tow).

An XML schema is a description of a type of XML document, typically expressed in terms of constraints on the structure and content of documents of that type above and beyond the basic syntactical constraints imposed by XML itself.in this layer we will produce an XML schema for every aspect. Then we will apply the two style sheets we will talk about on the next parts.

Stakeholder layer: In this layer the real view will be produced based on the type of stakeholder. This can be done by creating XML style-sheets using XSLT language.We will design a specific template (XML style sheet) for every stakeholder. A default template will be also designed for all stakeholders. After we decide the type of stakeholder the stakeholder style-sheet will be applied to the resulting documentation template from the first layer.

Preferences layer: In this layer, the XML style-sheets will be generated dynamically from the preferences of the users regarding the presentation order of requirements and the level of zoom detail of the different sections of the SRS.

CONCLUSIONS

In this paper we give an overview about Software Requirements and its categories and Requirement Engineering life cycle. We talk briefly about Software Requirements Specification and we focus on the most important standard which is IEEE Std 830-1998. After that we show the state of the art. Then we introduce our main contribution which is the proposition of a view based mechanism that makes the SRS more convenient and adaptable for its readers. A conceptual and logical model of this mechanism are proposed. A library system case study is used to illustrate the conceptual model of the views mechanism.

Many research topics could be derived from this research project dealing for example with the different views resulting from the combination of the criteria in the three layers of our proposed mechanism..

References

[1] [Online], S. c.,"system context diagram "http://en.wikipedia.org/wiki/System_context_diagram.

[2] Aybüke Aurum, Claes Wohlin. (2005). Engineering and Managing Software Requirements. Springer-Verlag Berlin Heidelberg.

[3] Lauesen, S. (2002). Software requirements Styles and techniques. Pearson Education.

[4] Frauke Paetsch,Dr. Armin Eberlein , Dr. Frank Maurer. (2003). Requirements Engineering and Agile Software Development. Vinu V Das, Member, IEEE, 6.

[5] Pete Sawyer, Gerald Kotonya. SWEBOK: Software Requirements Engineering Knowledge Area Description. 17.

[6] P Sawyer, P.and Kotonya, G. (2001), "Chapter 2 - Software Requirements", © IEEE – Trial Version 100, pp.1-26, 2001.

[7] Kandt, R. K. (2003). Software Quality Improvement Software Requirements Engineering:Practices and Techniques. : Practices and Techniques", JPL Document D-24994, SQI Report R-3, pp.1-35, 2003.

[8] IEEE Computer Society. (2009). IEEE Recommended Practice for Software Requirements SpeciÞcations. 39

[9] Jantima Polpinij (2009).An Ontology-based Text Processing Approach for Simplifying Ambiguity of Requirement Specifications. IEEE Asia-Pacific Services Computing Conference (IEEE APSCC)

[10] K Abernethy,J Kelly,A Sobel, J kipper, J Powell. (2000)Technology Transfer Issue for Formal Methods of Software Specification. IEEE

[11] V Pekar, M Felderer, R Breu. (2014). Improvement Methods for Software Requirement Specifications: A Mapping Study. 9th International Conference on the Quality of Information and Communications Technology

[12] A Gross, JDoerr(2012) What You Need Is What You Get! The Vision of View-Based Requirements Specifications.IEEE

[13] Sommerville, l. Software engineering. 2011: pearson.

[14] A AlSanad, A Chikh(2014). Reengineering of Software Requirement Specification. © Springer International Publishing Switzerland

[15] [Online], S. c.," Decision support system"http://en.wikipedia.org/wiki/Decision_support_system.

[16] Prakash Ranganathan, Kenneth Magel. (2010). Understanding Requirement Engineering (REQ) from a Software Agent Modeling Perspective. IEEE, 3.

[17] PATIL, KAPIL B. Software Requirements specification for LIBRARY SYSTEM. July 8, 2009