American Journal of Information Science and Computer Engineering
Vol. 3, No. 4, 2017, pp. 52-55
http://www.aiscience.org/journal/ajisce
ISSN: 2381-7488 (Print); ISSN: 2381-7496 (Online)

Impact of Avoiding Non-functional Requirements in Software Development Stage

Muhammad Shahid[1, *], Khawaja Awais Tasneem[2]

Centre of Excellence in Science and Applied Technologies (CESAT), Islamabad, Pakistan

Abstract

Non-functional requirements such as security, performance, usability, scalability and maintenance define the quality attributes or constraints of software to be developed. During the development of software, they are as critical as their counterpart functional requirements because they ensure the quality of software product. Avoiding non-functional requirements or constraints during the requirements engineering process could lead to the failure of software projects. In order to investigate this problem, we have reviewed several published research papers. Based on the study of previously published research this paper gives an overview of impact of avoiding nonfunctional requirements of software project during requirements engineering process. In this research we will also give our recommendations for the improvement of requirements engineering practices.

Keywords

Non-functional Requirements, Requirements Engineering, Software Development models, Security, Scalability, Maintainability

Received: March 21, 2017 / Accepted: April 7, 2017 / Published online: June 14, 2017

1. Introduction

During the software development both functional requirements and nonfunctional requirements have to be taken into consideration. Software's functional requirements [1] are the services that the intended software must deliver. It simply defines that how the intended software must response and act to specific inputs and conditions. Nonfunctional requirements are the restraints on the functionalities provided by the intended software like usability; flexibility, efficiency, availability and portability. These are applied on the software in general, rather than separate software functionalities. Good quality software product can be produced by applying the optimum mixture of functional and nonfunctional requirements. Also, achievement of the quality attributes of the intended software is linked with software architecture for that system [2]. Requirements engineering can be subdivided into requirements eliciting, requirements analyzing, requirements documentation, requirements validating and requirements management. Though nonfunctional requirements are often more important than functional requirements [1], but it has been observed that most of the requirements management artifacts emphasis on the desired functionality of software project. Real-time software's are mainly quality driven than functionality concerned. Numerous researches stress the definition, grouping and representation of nonfunctional requirements but they lack in providing a framework for the proper consideration and management of software's nonfunctional requirements.

We will discuss the impacts of ignoring the software's nonfunctional requirements during software development process.

This paper is organized as follows: Section 2 will provide a brief description of the requirements engineering process and its sub-activities. Sections 3 will discuss the various available requirements engineering process models. Section 4 will give

* Corresponding author
E-mail address: shahidshaikh62@yahoo.com (M. Shahid), awaistasneem@gmail.com (K. A. Tasneem)

the comparison of the already available models. At last, in Section 5 we will conclude the paper and give our suggestions for further improvement of software development process.

2. Requirements Engineering

Requirements are the foundation of a software project. Since requirements gathering are the first step of the softwaredevelopment lifecycle, therefore it sets sensible objectives to be attained. Requirements engineering process consists of some sub-processes like requirements elicitation, requirements analysis and negotiation, requirements specification, requirements modeling, requirements verification and validation, requirements traceability and requirements management.

Errors produced at starting level of software development may lead to failures and canincrease the cost of maintenance during the maintenance phase of the software. According to the Standish Group's CHAOS Reports from 1994 and 1997 [STA97], the maximum contribution in the failure of a software project is due to poor software requirements engineering. Another research [COM97] of five hundred Information Technology managers in the U.S.A and United Kingdom stated that about seventy-six percent of the managers experienced project failure throughout their jobs, and the major source of project failure was "changing user requirements. During the process of requirements management, both types of requirements should be given importance in order to produce quality software but literature [15] shows that NFRs have not been given their due importance as compared to functional requirements during the requirements engineering process.

3. Software Development Models

Waterfall Model

Waterfall model is a sequential model which works like a flowing water fall. This model was presented by W. W. Royce [14]. Waterfall model of software development consists of following phase's i.e. Requirements analysis, software requirements specification, Software design, Implementation, Testing, Integration, Deployment, Maintenance. In this model each phase has a separate starting and ending point. Code is written after the complete understanding of requirements therefore it ensures timely delivery of software to the clients. Errors could be corrected during the design phase of model but it become difficult with the passage of time. With the passage of time after integration of different modules, the code becomes more and more complex due to which change become very difficult and costly [3].

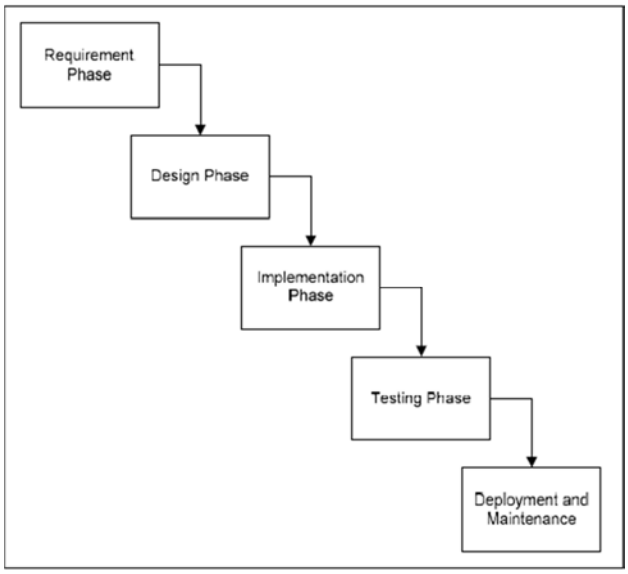

Figure 1. Waterfall Model.

Prototyping Model

Prototyping model talks about creating the prototypes of software applications, for example, incomplete versions of the software program being developed. It helps to envision different components of software under development to bind the gap of confusion to the end user's requirements by the development team. Moreover, it reduces the number of iterations that may occur in the waterfall model and tough to implement because of rigidity of waterfall model. So, after the development of final prototype, software requirement is considered as locked [4].

Spiral Model

It is the union of components of design and prototyping in phases. It was developed by taking the best features of top down and bottom up approaches. This process model unites the functionality of prototyping process model and waterfall process model. Spiral process model is preferred for large, costly, high risk and mission critical projects [17]. It uses some of the stages of waterfall process model, in effect with the similar order, divided into project planning, project risk assessment, and construction of prototypes & simulations [5].

Extreme Programming (XP)

It is based on iterative and incremental development, where requirements and solutions evolve through collaboration between cross-functional teams. It can be used with any type of the project, but it needs more involvement from customer and to be interactive. Also, it can be used when the customer needs to have some functional requirement ready in less than three weeks. It decrease the time required to avail some system features. Face to face communication and continuous

inputs from customer representative leaves no space for guesswork. The end result is the high quality software in least possible time duration and satisfied customer. Extreme programming favors simple designs, common metaphors, collaboration of programmers and users, frequent communication and feedback. XP needs special skills for the team [6].

V-Model

V-Model was published in "Development Standards for IT Systems of the Federal Republic of Germany in1997" [9]. The phases of V-Model were organized in a serial that dorms a VShape. V-Model is designed into a particular project because V-Model is independent of project. V-Model includes three stages: the life-cycle process model, the distribution of methods and functional tool requirements. At each stage the standards are organized according to the areas of functionality. There are four sub-methods: project management, quality assurance, system development and the configuration management [7].

Volere

The Volere requirements engineering model is a complete requirements engineering process model with detailed guidance to collect the requirements of software system. The process model delivers a rigorous guideline to categorize and improve requirements, both non-functional & functional. Volere also suggested a method for taking requirements of agile projects and then determining requirements with the help of prototypes and deviances by testing mock up projects for user's work. This process model would be taken as being iterative [9].

Web Site Design Method (WSDM)

Website design method was developed by [13] and reflected as one of the latest methods for the web application software design & development. Using WSDM, new and creative websites could be developed. Website design method is end user driven method due to the fact that it uses requirements of the end users to determine and run the process of website development. Moreover, it uses the idea "Audience Class" to mention each end user(s) who is stakeholder and have his specific requirements.

SCRUM

SCRUM is an agile based approach in which software is developed by dividing it into small set of functionality called sprints which typically span into four weeks. It is managed by scrum master who arrange meetings on daily progress and review of activities of the last day. This method was documented by Pittman and after him it was further extended upon by Booch. It may use the same roles for project staff as outlined by Graham [11], but it organizes and manages the team process in a new way. SCRUM is a management, enhancement and maintenance methodology for an existing system or production prototype. It assumes existing design and code which is virtually always the case in object-oriented development due to the presence of class libraries.

RUP

Rational unified process model is architecture and use case driven model [10]. Every use case can be comprised of different scenario(s) which is considered as an effective method of taking software requirements. It is also iterative method. RUP was created for addressing particular software development requirements and its design. As a descendent of Object oriented process model, it was created in 1990s by the Rational Software. Therefore, it is generally recognized as Rational Unified Process model. IBM developed Rational Software back in 2003 and continued to produce & market the process model as part of different SDLC tool. The Rational Unified Process provides six guidelines to implement successful projects. These are outlined in below Figure 2.

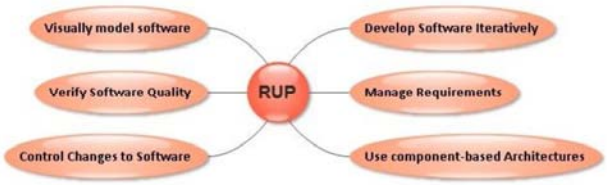

Figure 2. The Principles of RUP.

4. Comparison of Requirements Engineering Models with Respect to NFRs Management

Following Table 1 is the comparison of the selected software development lifecycle models with respect to their ability to manage NFRs during the process of requirements engineering. Note that here "Y" represents yes i.e. the model manages NFRs during the particular phase of requirements engineering,

"N" represents that the model does not manage NFRs at that particular phase of the model and similarly "P" represents that model partially manages NFRs. Here comparison has been done between the requirements models and the various phases of SDLC models during the process of requirements engineering.

Table 1. Comparison of Software Life Cycle Models.

RE Activities	WF Model	Prototyping Model	Spiral Model	XP	V Model	Volero	WSDM	SCRUM	RUP
Elicitation	Y	Y	Y	N	P	P	P	N	P
Analysis	Y	N	N	N	P	P	P	N	P
Specification	Y	N	N	N	P	Y	P	N	N
Modeling	Y	P	N	N	N	N	N	N	N
V & V	P	P	Y	N	P	P	P	N	P
Management	Y	Y	Y	N	P	P	P	N	P
Traceability	N	N	N	N	P	P	N	N	N

On the basis of above comparison results, we conclude that none of the selected process models completely mange nonfunctional requirements.

5. Conclusion and Future Work

The key objective of writing this paper was to assess different available requirements engineering process models with respect to their capability to entirely consider NFRs and the effects of ignoring the important NFRs. An evaluation measures associated to the various stages of requirements engineering (to check the ability of process model to manage the NFRs during requirements engineering phase) was identified by doing a comprehensive literature review. Using the above criteria nine process models, Waterfall model, prototyping model, spiral model, rational unified process model, Volere model, V-Model, XP, Scrum methodology and Website design method were assessed by debating on how NFRs are dealt during the execution of major tasks of requirements engineering by evaluating the results. The results show that none of the selected process models completely mange NFRs, though nonfunctional requirements are the critical kind of software system features. This introduces the complexity in the developed software and can cause a system failure because of ignoring the nonfunctional requirements during the development phase. It has already been observed that mostly system failures occur due to the absence of nonfunctional requirements support [16].

Moreover, cost of incorporating the nonfunctional requirements during the developments early stages is far more less than the cost it takes during the maintenance phase. However, there is an existing work emphasizes on one individual or some tasks of the RE as stated in the introduction section of this paper. Not considering NFRs will increase the level of software failure risk and reduce the level of software quality, hence we will propose a new model in future that combines the concepts stated and reviewed in this paper.

References

[1] Somerville, I. (2015). Software Engineering, Addison Wesley, Harlow, England, 10th edition.

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

[3] Boehm, Barry. "Anchoring the software process." *IEEE software* 13.4 (1996): 73-82.

[4] Doyle, William P. "Business modeling, software engineering and prototyping method and apparatus." U.S. Patent No. 5,233,513. 3 Aug. 1993.

[5] Boehm, Barry. "A view of 20th and 21st century software engineering." *Proceedings of the 28th international conference on Software engineering*. ACM, 2006.

[6] Paulk, Mark C. (2001) Extreme programming from a CMM perspective. *IEEE software* 18(6), 19-26.

[7] Cysneiros, Luiz Marcio, and Julio Cesar Sampaio do Prado Leite. (2004) "Nonfunctional requirements: From elicitation to conceptual models." *IEEE Transactions on Software Engineering* 30(5), 328-350.

[8] Somerville, I. (2010). Software Engineering, Addison Wesley, Harlow, England, 9 Edition.

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

[10] S. Robertson and J. Robertson, (1999), *Mastering the Requirements* Process (Addison-Wesley).

[11] De Troyer, O. M. F., & Leune, C. J. (1998). WSDM: a user centered design method for Web sites. Computer Networks and ISDN systems, 30(1-7), 85-94.

[12] Schwaber, Ken. (1997), Scrum development process. *Business Object Design and Implementation*. Springer London, pp.117-134.

[13] Karmokar, S., H., & Tan, F. B. (2016), Using Multidisciplinary Design Principles to improve the Website Design Process. Pacific Asia Journal of the Association for Information Systems, 8(3).

[14] W. W. Royce (1987), Managing the Development of Large Software systems: Concepts and Techniques, Proceedings of the 9th International Conference on Software Engineering (ICSE), pp. 328-338.

[15] Alashqar, A. M., Elfetouh, A. A., &El-Bakry, H. M. (2015). Evaluating User Interface Management Systems based on Quality Attributes and Unit Operations. *International Journal of Computer Applications* 116(9).

[16] Trupti Suryawanshi (2016), Modeling of Nonfunctional Requirements for Agile Development Processes, Vol-4, IJITE, 2016.

[17] Chandra, V. (2015), Comparison between Various Software Development Methodologies, International Journal of Computer Applications, 131(9), 7-10.