European Journal of Scientific Research
ISSN 1450-216X Vol.68 No.1 (2012), pp. 101-109
© EuroJournals Publishing, Inc. 2012
http://www.europeanjournalofscientificresearch.com

Communing Different Views on Quality Attributes Relationships' Nature

Saleh Aldaajeh
Department of Software Engineering
Al-Zaytoonah University of Jordan, Amman, Jordan
E-mail: saleh.aldajah@yahoo.ca

Tariq Asghar
Systems and Software Development Department
Brainglass AB, Stockholm, Sweden
E-mail: tariqasghar@yahoo.com

Abed Ali Khan
Systems Engineering, Research and Development Dept.
Hanns Systems AB., Copenhagen, Denmark
E-mail: Aaka07@student.bth.se

Muhammad ZakaUllah
Systems Engineering, Research and Development Dept.
Hanns Systems AB., Copenhagen, Denmark
E-mail: Zumu08@student.bth.se

Abstract

The understanding of software products quality attributes relationships' nature impersonates a focal role in sustaining a sufficient level of quality to software products and their development processes. In order to conduct correct tradeoffs between software development solutions and therefore develop a software system in a cost effective way, we need to take into account the relationship between quality attributes. This research study discusses the previous views on quality attributes relationships. Additionally, this study reveals the nature between software product different quality attributes relationships based on the utilized results from previous studies investigating quality attributes relationships for software products.

Keywords: Real-Time Systems Embedded Systems, Dependability, Software Architecture, Software Architecture Strategies, Software Quality Attributes Relationships, Tradeoffs, and ISO-9126 Quality Model.

1. Introduction

Quality of a software system has received an increased attention for being a key characteristic, in its own or in its implications. Introducing quality and precision to software products and to its development processes in a cost effective way (i.e. less cost, and shorter time) is an ultimate goal for most of the software engineering activities [1] [2]. The achievement of quality to a software system is never isolated to a certain technique and or a distinct stage. Therefore, when developing a software system it is necessary to consolidate the achievement of quality to a software system via conducting the correct tradeoffs. Quality attributes relationships can be undertaken as one of the most critical aspects of conducting successful tradeoffs to achieve quality to a software system. Nevertheless, the understanding of quality attributes relationships will not only increase the likelihood of sustaining a sufficient level of quality to a software system but also will help in the prioritization of quality attributes for software products (i.e. which one is to be focused on and which is to be forsaken).

The software quality life cycle starts as nonfunctional requirements which describe how the system should respond to a certain stimulus, "how the system is proposed to perform things" [1]. The software quality is defined according to [3] as the degree to which a system, component or process meets specified requirements, customer or user needs and expectations. When developing a software system, it is necessary to sustain a sufficient level of quality to the software product and their development processes [2]. Introducing quality to software at earlier stages is better (i. e. less maintenance cost) than doing it at later ones [1] [4]. The achievement of quality for a software system is not a clean-up process for errors and defects [5]. Techniques such as inspection, reviews, and testing have already been used in practice since decades as candidate techniques to introduce quality to software [5] [6]. Nevertheless, the achievement of quality for software systems is usually done via three basic ways; the oldest two methods are nowadays practiced within software verification and validation activities, while the third and the recent method is engineering quality for software systems and is used throughout the development stages [6]. These methods are usually used together for interchanging goals to sustain a sufficient level of quality to software system throughout its development processes [5]. Additionally, it's widely accepted that engineering quality for software systems is less expensive, especially when introducing quality to software at earlier stages than doing it at later ones [1] [7] [6]. For example, introducing quality to software at the software architecture stage costs less than introducing it during at the maintenance stage. Engineering quality for software is the recent approach adopted to achieve quality for software systems. This method requires a large effort in coordinating activities and selecting the appropriate tools and engineering techniques.

Software system quality achievement's life cycle is continuous throughout the software development process were the aforementioned methodologies/approaches are adopted in almost in every stage of the software development processes. Thus, the software quality achievement's life cycle is never isolated to a distinct development process of the product life cycle or to a specific methodology. Thus, the afore stated methodologies are continuing and overlapping over all development processes with an interchanging between these methods, goals and practices to be undertaken for the different tradeoffs embracing different factors which may have an influence on the system's quality [6] [8]. According to [9], the relationships between quality attributes exist. Furthermore, the understanding of the quality attributes relationships minimizes the chances of the occurrence of unwelcome system behavior during runtime and it increases the chances that the developed system will meet its specifications [10] [11].

The understanding of quality attributes relationships nature as stated in [2], will aid in deciding which quality attributes to prioritize and which one to forsake during the development stage. Unfortunately, quality attributes relationships' nature is poorly explored.

This study has two main contributions: To commune different views on quality attribute relationships for various types of software systems, and to reveal the nature of the relationships between quality attributes for software products. This article is organized as follows: section 2 describes the scope of this study. Section 3 presents the framework in which this study results is conducted. Furthermore, the results of this study is presented and analyzed in section 4. Finally, the research discussion and conclusions from this study are illustrated in section 5.

2. Background and Motivations

This study is primarily concerned with identifying and exploring the nature of the relationships between quality attributes and to match the different views on quality attributes relationships. However, software quality is one of the thematic blocks in which this study is focused on. The achievement of software quality can be done via three basic ways [6]. The recent and earliest way is to engineer quality solution and concerns at the different development processes. However, the software architecture plays a vital role and constrains up to 45% of the achievement of the overall quality to a software system [12], engineering quality via software architecture stage is further discussed in section 2.1. The development of the software system may have several options which requires tradeoffs, it is also necessary to compromise between different quality attributes to sustain a sufficient level of quality to a software system [13]. Tradeoff is further discussed in the context of quality attributes relationships in section 2.2.

2.1. Software Quality

First, software quality attributes are elicited as nonfunctional requirements which described how the system is supposed to react toward a certain stimuli. Once software quality requirements are elicited and correctly documented, in most cases, the second stage of a software development life cycle is the "‘High-level Design"’ or the "‘Architecture Stage"’ [14]. Software architecture in terms of components and connectors supports the achievement of the system’s quality attributes by providing design decisions and specifications that satisfy the means of quality attributes. The IEEE Standard 1471 defines software architecture as the fundamental organization of a system embodied in its components, their relationships to each other and to the environment and the principles guiding its design and evolution. However, The Software Architecture Society has been active in this field and it has shown how software architecture constrains the achievement of quality [7] [8] [13] [15] [16]. According to L. Bass et al., software architecture is the structure or structures of the system, which compromise software components, the externally visible properties of those components and the relationships between them [17].

Software architecture is documented via multiple methodologies and structured via different styles. Additionally, a study of monitoring the progress of achieving quality throughout the software life cycle stages indicates that the software architecture stage constraints up to 45% of the overall achievement of the systems’ quality attributes [12]. The achievement of quality attributes for a system in software architecture can be separated into two levels: Requirements and Solutions. The software architecture community has developed several frameworks used to elicit, specify and categorize quality requirements [17] [18] [19]. These frameworks are used to create what is known as quality attributes general scenarios, which help in developing quality attributes concrete scenarios and therefore assist in evaluating software architectures [17]. General scenarios are and can be used as a template to define and construct multiple concrete scenarios for quality attributes depending on how much the quality requirement embraces of the system’s non-functional requirements [20]. Quality attributes concrete scenarios are methods to create a description of the problem (constraint or requirement) which questions whether the software architecture candidate solutions satisfy the desired nonfunctional requirements set on the system or not [20]. To achieve quality for a software system at the architecture stage, it is important to evaluate the system architecture with respect to the desired quality and the requirements placed on the system. In general, the evaluation of software architecture aims at providing evidence as to whether the architecture is suitable with respect to its functional and nonfunctional requirements.

Software architecture evaluation methods are usually based on critical analysis of how the system fulfills its specification. There are several methods that can be used to assess software architecture and predict the quality of software at the architectural level (e.g. ATAM, SAAM, etc.) [17]. Software architecture evaluation does not only help the evaluators to be able to reason about the achievement of software quality attributes and to have a certain level of predictability of the system’s behavior, but it also helps to select suitable tactics to achieve the desired quality attributes and hence supports the trade-offs and therefore the selection of an appropriate architectural pattern. Objective reasoning is one of the approaches used to assess the software quality requirements achievement in software architecture through reasoning based on logical arguments conducted from experiences, best practices, etc [13]. Since, the solution for achieving software quality attributes during the software architecture stage is supported by selecting appropriate architectural strategies “Tactics” and patterns/style [16] [17]. For example, software architects according to [10] and [21] will have the choice to select the software architecture pattern/style and tactics necessary to achieve a certain quality attribute and hence resulting in less hindrance to the achievement of others. Thus, it pays off to assess the quality attributes relationships at this stage. Therefore, the understanding of the quality attributes relationships will assist the development of the software system to meet its specifications at early stages through selecting the most suitable alternatives.

However, software products quality can be lost when the software development is poorly planned, implemented and tested. Therefore, achieving quality to a software product requires a large effort invested in the collaboration and the coordination of the different development activities throughout the entire software development life cycle. Furthermore, it is better to introduce quality to a software product at the early development stage, especially as the software architecture stage controls almost the half of the overall achievement of quality to software products.

2.2. Tradeoffs

Software products quality attributes are among the important subjects that require trade-offs especially when facing conflicts in quality requirements or customers' desires. In order to conduct a correct and a successful tradeoff a rigorous analysis might be required. According to [10], trade-off process can be recognized from two dimensions: Techniques used, and the depth of the analysis of the trade-offs processes. There are several categories for tradeoffs techniques such as experienced based, model based, and mathematically based [8]. The analyses of the situation where the tradeoff process is required focuses on appraising the possible factors which may affect the selection of the appropriate choice suitable for a certain situation. However, the critical part of tradeoffs processes is to define the factors. Software architects are the ones who make the final decisions on how the system shall meet its specifications and customers’ expectations.

In different types of software systems trade-offs are done at two main levels with respect to its quality. The first level is to prioritize the desired system quality attributes, for example, a landing system must have high availability while the system is not required to be reliable for longer periods. The second level of trade-offs is the selection of a software architecture pattern/style that supports the pre-prioritized quality attributes the best way possible [8]. The second main tradeoffs a sub-tradeoffs is needed in the form of technical reviews. The intention of technical reviews is to evaluate the solution(s) against the predefined specifications and compliance with standards and other documentations [22].

3. Different Views on Quality Attributes Relationships Nature Communing Framework

Comparison framework adopted in this study is dependent upon the very recent three studies investigating the relationships between quality attributes [2] [23] [10]. As stated, this study has two main contributions: investigating the nature of the relationships between quality attributes and to identify the relationships between quality attributes. However, the comparison process is adopted in this study is a disciplined systematic process. Each of the afore stated studies is investigated from the methodology adopted, quality models considered. Furthermore, the outcomes of each study is carefully examined and compared to each other. Figure 3 depicts the comparison framework. The study conducted by Svahnberg et al. is a case study which investigates the quality attributes relationships from different perspectives: industrial, academic and literature views. According to Svahnberg et al. study, the match between the different perspectives on quality attributes relationships indicates that there is a negative relationship between the quality attribute “usability” and the quality attribute “efficiency”. Furthermore, the study indicates that there are also negative relationships between the quality attribute “Functionality” and the quality attribute “efficiency”. However, the mapping of relations between the three views (academia, industry, and literature) is depicted in table 2.

Unfortunately, the study conducted by Svahnberg et al. does not elaborate on the nature of the relationships or investigate the creation of these relations between the different software product quality attributes.

Another study conducted by Zulzalil et al. investigating quality attributes relationships for web-based software systems. Zulzalil et al. study is a survey based study. According to H. Zulzalil et al. study the relations between the quality attributes is dependent on the type of the software product (i.e. quality attributes relationships for a web-based software application may be different or change for computer-based or embedded systems software applications). Furthermore, in their study the evaluation of quality attributes relationships is totally dependent on the user. For example, the quality attributes “functionality” may have a negative relation with the quality attribute “usability” for non-professional web-based software applications users. The, relationships indicators utilized from H. Zulzalil et al. are illustrated in table 1.

Table 1: Quality Attributes Relationships Found in [23]

QAs	**Functionality**	**Efficiency**	**Reliability**	**Usability**	**Maintainability**
Efficiency	Neg				
Reliability	Pos	Independent			
Usability	Pos	Neg	Pos		
Maintainability	Pos	Neg	Pos	Independent	
Portability	Independent	Neg	Independent	Independent	Positive

Another study conducted by [10] is focused on providing a framework to explore the nature of the relationships between the software products quality attributes based on software architectural strategies and scenarios. S. AlDaajeh et al. study focuses on exploring the nature of the relationships between quality attributes of the system using a thorough analysis of an experienced software architects. However, the study also provides indicators to investigate the quality attribute relationships' nature and strength. The last and the latest study enclosed in this study framework, is another study conducted by S. Al-Daajeh et al. [24]. In this study the relationships between dependability and embedded systems quality attributes is further investigated by applying the framework proposed in [10].Table 3 depicts the relationships between the embedded systems and the dependability quality attribute “Maintainability” with other dependability and embedded systems quality attributes.

Table 2: Quality Attributes Relationships According to [23]

Quality Attribute	**Vs**	**Quality Attribute**	**Relations**		
			Industry	**Academia**	**Literature**
Efficiency		Functionality	neg		neg
Functionality		Portability		neg	pos
Functionality		Time to Market	neg		
Maintainability		Reliability	neg		Pos
Reliability		Usability	pos	neg/pos	
Reliability		Portability		pos	
Portability		Maintainability	neg		neg
Usability		Efficiency	neg	neg	neg

Table 3: Dependability Quality Attribute "Maintainability" Candidate Architectural Tactics Impact Evaluation

Candidate Tactics	**Functionality**	**Efficiency**	**Usability**	**Portability**	**Availability**	**Reliability**	**Integrity**	**Confidentiality**	**Safety**	**Level of Agreement**	**∑ Impact on QAs**	**Rank**
Manage Input/Output:												
Record /Playback	0	0	0	0	+1	+1	0	-2	+2	80%	+2	2
Separate Interfaces from Implementation	0	0	0	+4	+2	+2	0	0	+2	100%	+10	1
Specialized Access Routines/Interfaces	0	+4	0	-1	0	0	-1	0	0	60%	+2	2
Localize Changes:												
Semantic Coherence	0	-2	+4							80%	+2	1
Anticipated Changes	0	-2	+4	+4	0	0	0	0	0	80%	+2	1
Generalize Module	0	-1	+2	+2	0	0	0	0	0	100%	+1	2
Limit Possible Options	-2	+1	0	0	0	0	0	0	+1	60%	-2	3
Abstract Common Services	0	-2	+4	+4	0	0	0	0	0	80%	+2	1
Prevention of Ripple Effect:												
Hide Information	0	-2	+4	+4	0	0	0	0	0	100%	+2	3
Maintain Existing Information	0	-1	+2	+2	0	0	0	0	0	80%	+1	4
Restrict Communication Paths	0	+2	+1	+1	0	0	+3	0	0	60%	+6	2
Use and Intermediary	0	-4	+2	+2	+4	+4	0	-1	+3	80%	+10	1
Internal Monitoring:												
Built-in Monitors	0	-1	+2	-3	+4	+4	+4	-2	+4	100%	+10	1
Defer Binding Time:												
Run-time registration	0	-4	0	+2	0	0	0	0	0	80%	-2	4
Configuration Files	+2	0	+2	0	0	0	-2	-1	0	100%	+1	3
Polymorphism	0	-2	0	0	0	0	0	0	0	80%	-2	4
Component Replacement	+4	-2	+2	+2	0	0	0	-2	0	100%	+4	2
Adherence to Define Protocols	+2	+2	0	+3	+1	0	0	-1	0	60%	+6	1
Relationships	Pos	Neg	Pos	Pos	Pos	Pos	Neg	Neg	Pos			
Relationship Strength (Impact Median Value)	0	0	+1	+1.5	0	+0.5	-1	-0.5	+1			

4. Analysis

As stated before, this research study has two main contributions: to match the different views on software products quality attributes relationships, and to explore the nature of the relationships between the software products quality attributes. The results utilized from the study conducted by Svahnberg et al. investigates the quality attributes relations from different perspectives as illustrated in industrial studies, academia, and literature. However, most of the views on quality attributes relationships are a high level investigation on the quality attributes relationships.

Therefore, the relationships according to Svahnberg et al. study are subjective. Additionally, the study of Svahnberg et al. does not investigate the reason of why this relationship is for example positive or negative and how the relationship nature could be understood. Nevertheless, the relationships enclosed in the study of H. Zulzalil et al. indicates that the relationships between quality attributes is changing depending on the type of the software product and the system users' perspective and level of professionalism. The investigation of the software product quality attributes relationships' of H. Zulzalil et al. study do not contribute into exploring the software products' quality attributes relationships nature. Since the level of users' professionalism use of software applications is the base for evaluating the quality attributes relationships the results of H. Zulzalil et al. is totally subjective and may add a threat to conduct a correct tradeoff between the quality attributes relationships when developing a software system.

The studies conducted by Al-Daajeh et al. are a low level investigations of the software product quality attributes relationships [10] [24]. Although the study investigates the quality attributes relationships at the earliest stage of the software development process. The study do not indicates weather the quality attributes relationships is evolving throughout the software development process. Furthermore, according to these studies quality attributes relationships' are interchanging (i.e. when the quality attribute "Integrity and Confidentiality" is the quality attribute in focus it may hinder the achievement of the quality attribute "Maintainability" and therefore creates a negative relationship. On the other hand, if the quality attribute in focus is "Maintainability" then the achievement of this quality attribute may support the achievement of the quality attribute "Integrity and Confidentiality"). However, the relationship strengthens and weaknesses are investigated by exploring the impacts of candidate solutions to achieve a given quality attribute to a software product.

5. Discussion and Conclusions

The understanding of quality attributes relationships will not only increase the likelihood of sustaining a sufficient level of quality to a software system, but will also minimizes the chances of unwelcome behavior for the system during its operation. Therefore, in order to support system development and achieve a sufficient level of precision of product and processes, the nature of the relationship between quality attributes need to be identified and quantified.

Obviously, quality attributes relationships impersonate a focal role in sustaining a sufficient level of quality to a software product and their development processes. Calibrating and adjusting software products quality attributes relationships requires a rigorous analysis to conduct the correct tradeoffs.

The very recent studies that investigate the relationships between quality attributes are based on different approaches. For example the studies conducted by [2] and [23], are experience-based and utilized from industry and user-experiences. These studies investigate quality attributes relationships from a high-level and do not consider a specific development stage. Furthermore, to the knowledge of the investigator, all previous studies do not investigate the strength or weaknesses of the relations between software product quality attributes. Although there are differences between the approaches and the bases undertaken to evaluate the software product quality attributes relationships in both studies conducted by Svahnberg et al. and H. Zulzalil et al. there is a match between the results of both studies. Table 1 provides a match between the aforementioned studies' results [2] [23].

In order to eliminate validity threats to this study's results, we have only considered the studies that investigate quality attributes relationships using the ISO 9126 quality model. In conclusion, the contribution of this study is that quality attributes relationships can be undertaken from different perspectives. Therefore, when developing a software product we must take into account the level of tradeoffs we are required to conduct in order to sustain a sufficient level of quality to the software product. Clearly, more investigations on the quality attributes evolution throughout the software development processes are required.

Examining the previous studies investigating the quality attributes relationships for software products it is now obvious that quality attributes relationships can be interpreted as follows:

- Quality attributes relationships' nature may have different interpretations depending on the level of the investigation.
- Although quality attributes relationships may be different from a perspective to another, some quality attributes relationships may have a static relationship for different types of

systems, users, development stages, etc.. (i.e. the relationships between the quality attributes found by Svahnberg et al. and H. Zulzalil et al.).

- Assuring what both Svahnberg et al. and Zulzalil et al. the quality attribute relationship is negative between the quality attributes “Usability” and the quality attribute “Efficiency”. Additionally, there is a negative relationship between the quality attribute “Functionality” and the quality attribute “Efficiency”.
- The process of calibrating and adjusting quality attributes relationships for a software product shall be conducted from a low-level.
- Relationships’ strength and weaknesses can be only explored at the solution level (i.e. low level investigation).

References

[1] H. V. Vliet, Software Engineering Principles and Practice, 2nd ed. New York: John Wiley, 2000, ch. On Managing Software Quality, pp. 101–126.

[2] M. Svahnberg and K. Henningsson, “Consolidating different views on quality attributes relationships,” in Proceedings of the 8th Software Engineering Research and Practice (SERP), 2008.

[3] I. E. E. E., “IEEE standard glossary of software engineering terminology,” Institute of Electical and Electronic Engineering, Tech. Rep. IEEE Std. 610.12-1990, 2002.

[4] M. Svahnberg, C. Wohlin, L. Lundberrg, and M. Mattsson, “A quality driven decision support method for identifying software architecture candidates,” International Journal of Software Engineering and Knowledge Engineering, vol. 13(5), pp. 547–573, 2003.

[5] M. Andersson, “Managing software quality through empirical analysis of fault detection,” Ph.D. dissertation, Lund University, Dept. of Communication Systems ,Dissertation (No. 1101-3931), Skåne-Lund, Sweden, 2006.

[6] M. S. Deutsch and R. R. Willis, Software Quality Engineering a Total Technical and Management Approach. New Jersey: Prentice Hall, 1988.

[7] M. Svahnberg and C. Wohlin, “A comparative study of quantitative and qualitative views of software architectures,” in Proceedings of the 7th International Conference on Empirical Assessment in Software Engineering, 2003, pp. 1–8.

[8] P. Berander, L. Damm, J. Eriksson, T. Gorschek, K. Henningsson, P. Jönsson, S. Kågström, D. Milicic, F. Mårtensson, K. Rönkkö, P. Tomaszewski, L. Lundberg, M. Mattsson, and C. Wohlin, “Software quality and trade-offs,” Blekinge Institute of Technology, Ronneby, Blekinge- Sweden, Tech. Rep. Tech.June, 2005.

[9] J. A. McCall, “Quality factors,” Encyclopedia of Software Engineering (Marciniak, J., ed.), pp. 958–969, 1994.

[10] S. Al-Daajeh, F. AlQirem, and R. El-Qutaish, “Evaluating the relationships between the software product quality attributes: A tactic-based framework,” International Journal of Software Engineering and Applications, 2011.

[11] K. Misra, Handbook of Performability Engineering, 1st ed. Germany: Springer, 2008, ch. Dependability Considerations in the Design of a System, pp. 71–80.

[12] C. Ebert and R. Dumke, Software Measurement: Establish- Extract-Evaluate-Execute. New York - London -Springer: Springer, 2007, ch. Assuring the Quality of Software Systems, pp. 230–300.

[13] L. Bass, M. Klein, and G. Moreno, “Applicability of general scenarios to the architecture trade-off analysis method,” Carnegie Mellon University, Pittsburgh:PA, USA, Tech. Rep. TR-014, Oct 2001.

[14] T. Sherman, Advances in Computer and Information Sciences and Engineering. Germany: Springer, 2008, ch. Quality Attributes for Embedded Systems, pp. 535–539.

[15] L. Zhu, M. A. Babar, and R. Jeffery, “Mining patterns to support software architecture evaluation,” Software Architecture, Working IEEE/IFIP Conference on, vol. 0, p. 25, 2004.

[16] F. Bachmann, L. Bass, and R. Nord, “Modifiability tactics,” Software Engineering Institute Carnegie Mellon University, Pittsburgh:PA, USA, Tech. Rep. TR-002, Oct 2007.

[17] L. Bass, P. Clements, and R. Kazman, Software Architecture in Practice, 2nd ed. New York: Addison Wesley, 2003, ch. Understanding Quality Attributes and / Achieving Quality, pp. 71–127.

[18] N. Lassing, D. R¼senbr¼, and H. van Vliet, “On software architecture analysis of flexibility, complexity of changes size isn’t enough,” in Proceedings of Second Nordic Software Architecture Workshop (NOSA ’99), 1999, pp. 1103–1581.

[19] R. Kazman, S. J. Carrière, Woods, and G. Steven, “Toward a discipline of scenario-based architectural engineering,” Ann. Softw. Eng., vol. 9, no. 1-4, pp. 5–33, 2000.

[20] J. Heit, “Impact of methods and mechanisms for improving software dependability on non-functional requirements,” Master’s thesis, Stuttgart University, Dept. Informatics , No.(2548), Stuttgart, Germany, 2007.

[21] M. Svahnberg and C. Wohlin, “An investigation of a method for identifying a software architecture candidate with respect to quality attributes,” Empirical Software Engineering vol. 10, no. 2, pp. 149–181, 2005.

[22] S. R. Rakitin, Software Verification and Validation for Practitioners and Managers, Second Edition. Norwood, MA, USA: Artech House, Inc., 2001.

[23] H. Zulzalil, A. A. A. Ghani, M. H. Selamat, and R. Mahmod, “A case study to identify quality attributes relationships for web-based applications,” in International Journal of computer Science and Network Security (.CSNS), vol. 8(11), 2008, pp. 215–220.

[24] S. Al-Daajeh, “Engineering dependability to embedded systems via tactics,” International Journal of Software Engineering and Its Application, Korea, 2011 (Accepted for Publication)