An Ontological Framework to Manage the Relative Conflicts between Security and Usability Requirements

Dewi Mairiza and Didar Zowghi
Faculty of Engineering and Information Technology
University of Technology Sydney, Australia
(Dewi.Mairiza; Didar.Zowghi)@uts.edu.au

Abstract— **Non Functional Requirements (NFRs) are relative, so are the conflicts among them. In our previously developed catalogue of NFRs conflicts it can be observed that a number of specific pairs of NFRs are claimed to be in conflicts in some cases but they are also claimed not to be in conflict in the other cases. These relative conflicts occur because the positive or negative relationships among NFRs are not always clear and obvious. These relationships might change depending on the meaning of NFRs within the system being developed. This paper focuses on the application of ontology in managing the relative conflicts among NFRs, particularly the relative conflicts between security and usability requirements. The aim is to develop a framework to identify, characterize, and define corresponding resolution strategies for the security-usability conflicts. This paper thus describes the sureCM framework to manage these conflicts; summarizes the security-usability conflicts ontology; and demonstrates how the ontology will be used as a basis to assist analysts in managing conflicts between security and usability requirements.**

Keywords–non-functional requirements, conflicts, relative, framework, management, ontology, security, usability

I. INTRODUCTION

NFRs are recognized as a critical factor to the success of software projects. NFRs address the essential issue of the quality of the systems [1-3]; and are also considered as the qualifications of the operations [4, 5]. In the determination of a system's perceived success or failure, experience shows that NFRs are often more critical than individual Functional Requirements (FRs). Neglecting NFRs has led to a series of software failures, such as a number of systemic failures discussed in the literature [6-9].

However, although NFRs are widely recognized to be very critical, several studies reveal that NFRs are often neglected, poorly understood and not considered adequately in developing a software application. In the development of software systems, users naturally focus on specifying their FRs [1, 10]. Similarly, software developers also do not pay sufficient attention to NFRs [2, 11-13]. NFRs are not elicited at the same time and the same level of details as the FRs and they are often poorly articulated in the requirements document [12, 13]. Capturing, specifying, and managing NFRs are still difficult to perform because most software developers do not have adequate knowledge about NFRs and little help is available in the literature [14].

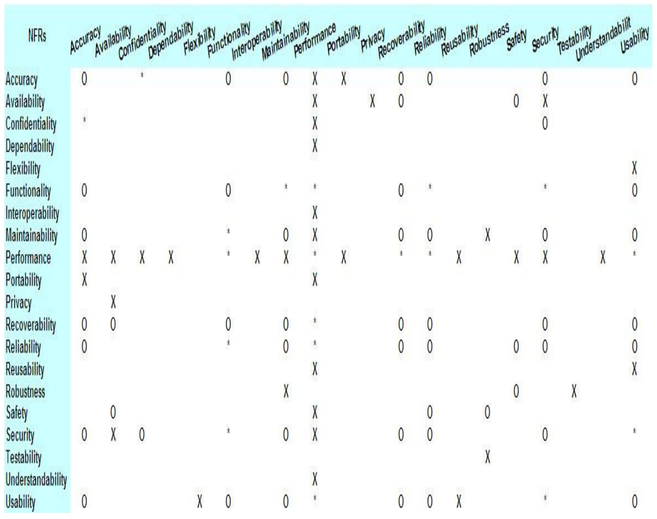
One of the characteristics of NFRs is "interacting", which means NFRs tend to interfere, conflict, and contradict with one other. Unlike FRs, this inevitable conflict arises as a result of inherent contradiction among various types of NFRs [1, 2]. Certain combinations of NFRs in the software systems may affect the inescapable trade offs [2, 7, 10]. Achieving a particular type of NFRs can hurt the achievement of the other type(s) of NFRs.

Prior studies reveal that dealing with NFRs conflicts is essential due to several reasons [15]. First of all, conflicts among software requirements are inevitable [1, 16, 17]. Conflicting requirements are one of the three main problems in the software development in term of the additional effort or mistakes attributed to them [17]. A study of two-year multiple-project analysis conducted by Egyed & Boehm [18, 19] reports that between 40% and 60% of requirements involved are in conflict, and among them, NFRs involved the greatest conflict, which was nearly half of requirements conflict [20]. Lessons learnt from practices also confirm that one of the essential issues during NFRs specification is management of conflict among interacting NFRs [2]. Experience shows that most systems suffer with severe tradeoffs among the major groups of NFRs. In fact, conflict resolutions for handling NFRs conflicts often results in changing overall design guidelines, not by simply changing one module. Therefore, since conflicts among NFRs have also been widely acknowledged as one of NFRs characteristics, managing this conflict as well as making this conflict explicit is important [21]. NFRs conflicts management is essential for finding the right balance of attributes satisfaction in achieving successful software product [7, 10].

II. PROBLEM DEFINITION

A number of techniques to manage the conflicts among NFRs have been discussed in the literature [15]. Majority of them provide documentation, catalogue, or list of potential conflicts among NFRs. These catalogues represent the interrelationships among various types of NFRs. Some examples are: the QARCC win-win approach [7, 22, 23], trace analyzer of the requirements traceability technique [24], and a technique that adopts a hierarchical constraint logic programming approach [25]. Apart from strength and weaknesses of each technique, however, NFRs are also relative [1]. NFRs can be viewed, interpreted, and evaluated differently by different people and different context within which the system is being developed. The interpretation and importance of NFRs may vary depending on the particular system and/or the extent of stakeholder involvement. Consequently, the positive or negative relationships among NFRs are not always obvious. These relationships might change depending on the meaning of NFRs in the context of the system being developed. Due to this relative characteristic, cataloguing the NFRs relationships in order to represent the conflicts among NFRs would inevitably produce disagreement. Identifying the NFRs conflict without understanding the meaning of NFRs in the system being developed may produce the erroneous conflict identification and analysis. Therefore, a technique to identify the conflict among NFRs by considering the relative characteristic of NFRs is essential. This technique will allow developers to identify and reason case by case in each system which NFRs of the system are in conflict and which NFRs are not.

To understand how NFRs conflict with each other, a catalogue of conflicts among NFRs with respect to NFRs relative characteristic has been developed from the literature [26]. This catalogue is a two-dimensional matrix that represents the typical interrelationships among NFRs, in term of the conflict emerges among them. In this catalogue, the relativity of NFRs conflicts is presented in three categories: *absolute conflict* (labeled as "X")*; relative conflict* (labeled as "*")*;* and *never conflict* (labeled as "O"). As illustrated in Figure 1, 19 pairs of NFRs in the catalogue are indicated have the *relative conflicts*, which means they are not always in conflict as they are claimed to be in conflict in the certain cases but they are also claimed as not being in conflict in the other cases. By combining this result with two other parameters: frequently listed NFRs; and concerned NFRs in various types of systems and applications domains [9], then this research focuses on investigating the relative conflicts between security and usability requirements.



NFRs	Accuracy	Availability	Confidentiality	Dependability	Flexibility	Functionality	Interoperability	Maintainability	Performance	Portability	Privacy	Recoverability	Reliability	Reusability	Robustness	Safety	Security	Testability	Understandability	Usability
Accuracy	O		*			O		O	X	X		O	O				O			O
Availability									X		X	O				O	X			
Confidentiality	*								X								O			
Dependability									X											
Flexibility																				X
Functionality	O					O		*	*			O	*				*			O
Interoperability									X											
Maintainability	O					*		O	X			O	O		X		O			O
Performance	X	X	X	X		*	X	X	*	X		*	*	X		X	X		X	*
Portability	X								X											
Privacy		X																		
Recoverability	O	O				O		O	*			O	O				O			O
Reliability	O					*		O	*			O	O			O	O			O
Reusability									X											X
Robustness								X								O		X		
Safety		O							X				O		O					
Security	O	X	O			*		O	X			O	O				O			*
Testability															X					
Understandability									X											
Usability	O				X	O		O	*			O	O	X			*			O

Figure 1 - Catalogue of Conflicts among NFRs [26]

Security requirements are widely recognized to be in conflict with usability requirements [27-29]. Security usually aims to make operations harder to do while usability aims to make operations easier [27]. Studies to date also indicate that current trend and challenge in the software engineering research and practices is producing such secure usable software products [28-30]. Systems that are secure but not usable will not be used, while systems that are usable but not secure will get hacked and compromised [28]. In fact, literature review reveals that the conflicts between security and usability are still not well understood. Braz, Seffah & Raihi in [31] even claim that "there is a very limited amount of work has been conducted on the security – usability conflicts, particularly on the intimate relationship that exists between security and usability".

Given the above context we are motivated to perform an investigation into the conflicts among NFRs, particularly the security-usability relative conflicts in order to increase our understanding about how these two NFRs conflict with and affect one another and how this conflict might be managed. Our research questions have been formulated as follow:

"With respect to the NFRs relative characteristic:

(1) Can we create a conceptual model of the conflicts between security and usability requirements?
(2) Can software developer use this model to manage (identify, characterize, and find the potential strategy to resolve) the conflict?"

III. Framework Development

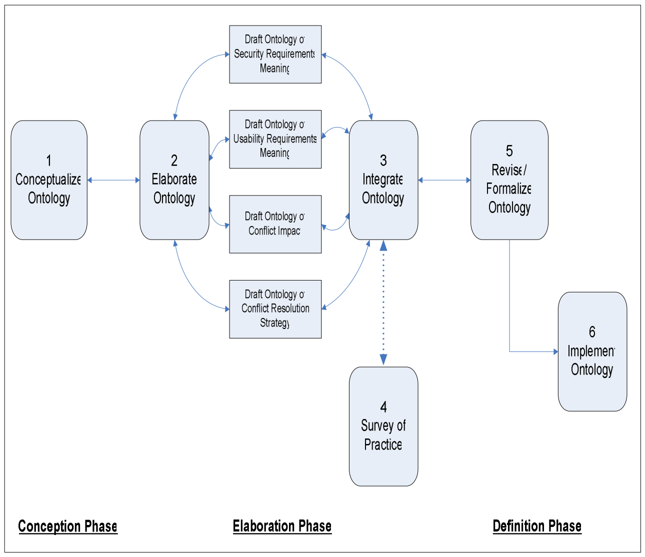
By adopting the IEEE Standard and ISO/IEC 9126, this research considers *security requirements* as the requirements that concern the protection of system, program and data from unauthorized access or malicious harm; and *usability requirements* as the requirements that specify the capability of the software product to be understood, learned, used, and attract the user [32, 33]. With respect to the NFRs relative characteristic, the term "*NFRs conflict*" is defined as a case where the satisfaction of a pair of NFRs is not possible within a specific context of the system being developed.

An iterative ontological engineering approach will be used as a basis of the framework. The reasons are ontology offers many benefits such as presenting an explicit semantic and taxonomy [34]; providing a clear link between concepts and their relationships [34]; assisting people in developing the representations or images of reality [35]; and facilitating knowledge sharing in the community [36, 37]. Thus, we believe that ontology will enable us to conceptualize the knowledge of security and usability conflicts. Ontology will assist us in investigating the conflicts phenomena, collecting the relevant information, conceptualizing the knowledge, and representing the conceptualization. Therefore, in this research, an ontological model of the security and usability conflicts with respect to NFRs relative characteristic will be developed. This ontology then will be used as the basis of the framework to manage the conflicts between security and usability requirements.

In building the framework and its ontology, we follow the Helix-Spindle Model for ontological engineering [38]. The reasons are:

(1) It has been recognized (e.g. [39]) that ontology is never complete. Thus, successful development of ontology is an iterative and incremental process [40]. The Helix-Spindle model reflects this iterative incremental development process through its three major phases: conception phase, elaboration phase, and definition phase.

(2) Helix-Spindle model combines both the theoretic and the pragmatic approaches to ontology development. Thus, it will increase the likelihood of maturity of the ontology



Draft Ontology of Security Requirements Meaning
Draft Ontology of Usability Requirements Meaning
Draft Ontology of Conflict Impact
Draft Ontology of Conflict Resolution Strategy
1 Conceptualize Ontology
2 Elaborate Ontology
3 Integrate Ontology
5 Revise / Formalize Ontology
6 Implement Ontology
4 Survey of Practice
Conception Phase
Elaboration Phase
Definition Phase

Figure 2 - sureCM Framework Development Process

As shown in Figure 2, the framework and its ontology will be developed in three major phases: *conception phase*, *elaboration phase*, and *definition phase*, following the Helix-Spindle model of ontological engineering. In the conception phase, the framework and its ontology will be conceptualized. Concepts, relationships, and behaviors of the ontology will be identified. An abstract level of the ontology will become the output of this phase. Then, each concept in the ontology will be elaborated in the elaboration phase. The hierarchy of security requirements meaning, usability requirements meaning, the conflicts impact, and the potential resolution strategies are created within this phase. Literature will be used as the main source for developing this ontology. Furthermore, in order to triangulate and enrich the ontology, a survey of practice will also be conducted. In definition phase, the ontology will be revised and formalized by using an ontological representation. Finally, the proposed ontological framework will be realized through the development of a proof of concept software tool that will be evaluated using experimentation.

IV. THE FRAMEWORK

A preliminary model of the security-usability requirements conflicts management (sureCM) framework is shown in Figure 3. The framework is specifically designed to identify and to characterize the conflicts between security and usability requirements and to discover the corresponding strategy for conflicts resolution. This framework consists of four types of input (i.e. security requirements, system context, application domain, usability requirements); four-layer process (i.e. P1, P2, P3, P4); and three types of output (i.e. list of conflict, nature of conflict, conflict resolution strategy). The security-usability conflicts ontology will be used as the basis to execute the four-layer process of the framework.

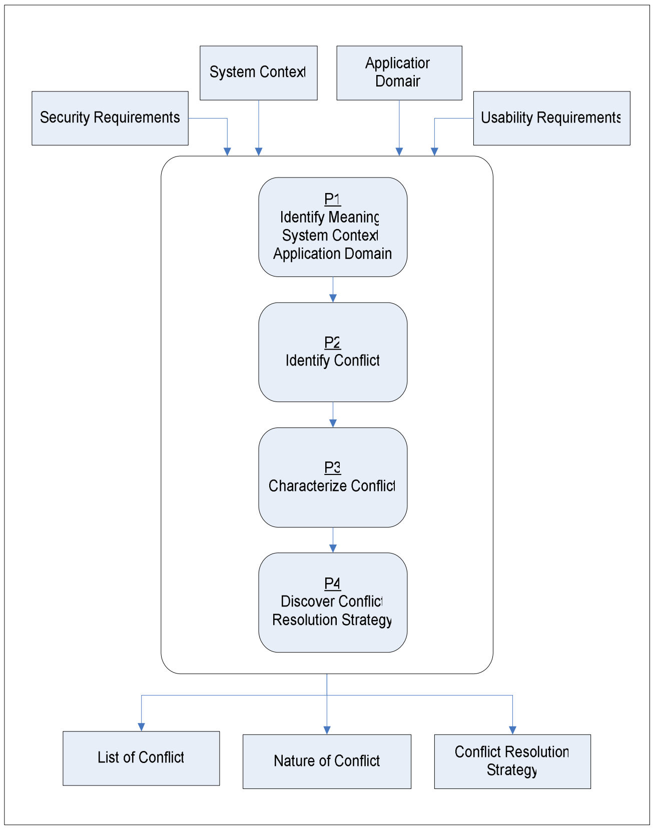


Figure 3 - Security-Usability Requirements Conflicts Management Framework

Figure 4 shows a preliminary conceptual model of the security-usability conflicts ontology. With respect to NFRs relative characteristic, two key-parameters are used to identify the existence of conflict: (1) the meaning of NFRs; and (2) the system context. The meaning of NFRs refers to the interpretation of NFRs in the system being developed while the system context refers to the context within which the system is being developed, that is characterized as the system type. The nature of conflicts is characterized as the impact of the conflicts against various components in the software development, e.g. personnel and schedule. Based on the nature of this conflict, then the corresponding strategies for conflict resolution will be identified with respect to the system's application domain.

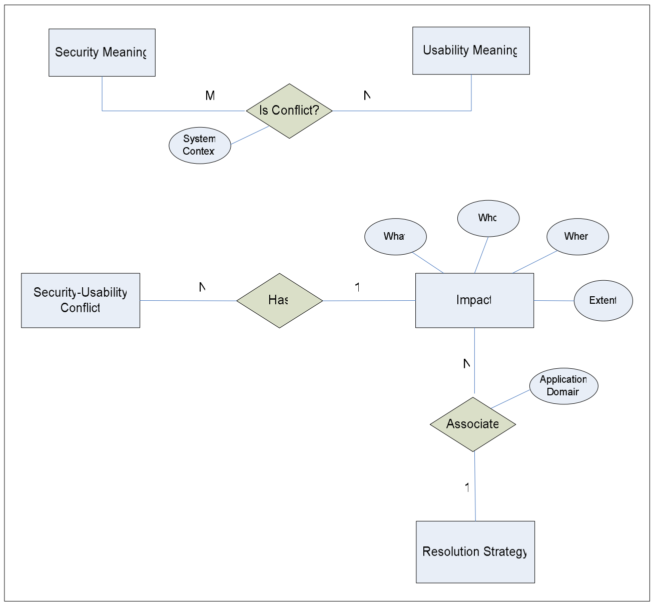


Figure 4 - Security-Usability Conflicts Ontology

As shown in Figure 4, this ontology has five concepts: security meaning; usability meaning; security-usability conflict; impact of conflict; and resolution strategy. The ontology behavior is represented by 3 types of relationships: is conflict; has; and associate. "Is conflict" represents which pair of security meanings are in conflict with usability meanings considering the system context. For each defined conflict, the corresponding potential consequences are then identified and linked by "has" relationship. Finally, with respect to the application domain, the conflict and its consequences are associated to the particular conflict resolution strategies through "associate" relationship.

The ontology illustrated in Figure 4 then will be elaborated by characterizing the meaning of security and usability requirements in term of the existence of conflicts among them; the impact of the conflicts; and the potential strategy to resolve the conflicts. From this elaboration, a security-usability conflicts knowledge-based will be developed. This knowledge-based is a combination of concepts and values deriving from the security-usability conflicts ontology. Some key-components of this knowledge-based are listed in Table 1.

Table 1 - Security-Usability Conflicts Knowledge-based

1 Security Meaning	2 Usability Meaning	3 System Context	4 Is Conflict	5 What (Impact)	6 Who (Impact)	7 When (Impact)	8 Extent (Impact)	9 Application Domain	10 Strategy Resolution

The sureCM framework represents both a process for identifying, characterizing, and discovering resolution strategy of the conflicts between security and usability; and the ontology of the security-usability conflicts. The process model for the entire framework can be illustrated as shown in Figure 5.

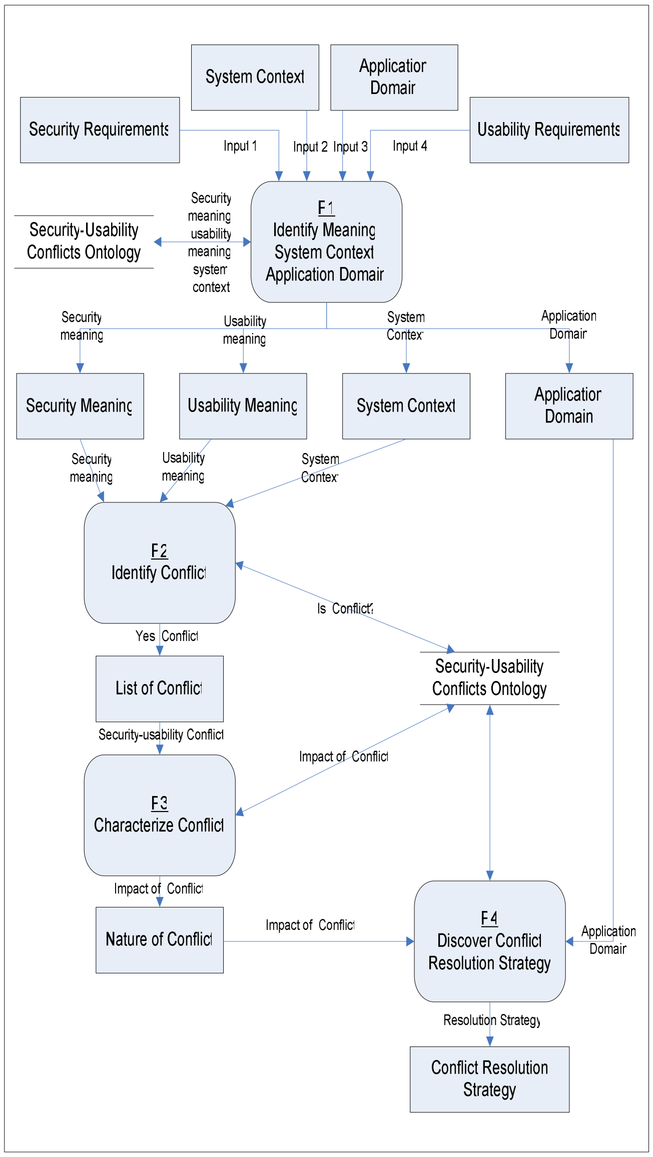


Figure 5 - Framework Process Model

As shown in Figure 5, the conflicts management process begin with the identification of the security and usability requirements meanings, the system context and the application domain. The ontology of security meaning, usability meaning, system context, and application domain will be used as the basis of identification. The second process is identifying the existence of conflicts. This process utilizes some outputs in the previous process: the security requirements meaning; the usability requirements meaning;

and the context of the system, as the parameter identification. Next process is when conflicts are characterized by their impacts, leading to the identification of the corresponding resolution strategies within the system application domain.

V. Conclusion

This paper describes a proposed framework to manage the relative conflicts among NFRs, particularly the relative conflicts between security and usability requirements. The framework uses an ontological approach as the basis to manage the conflicts. An ontological model of the security-usability requirements conflicts has been developed and presented. This ontology shows *when security and usability are in conflict, what the impacts of the conflicts are, and what the relevant strategies to resolve this conflict are*. Therefore, this proposed framework can be used to identify not only the existence of conflict, but also the type and significance of conflict, as well as the appropriate potential strategy to resolve the conflict.

Although the conceptual model of the framework has been established; a number of conference papers on investigating the notion of NFRs, the conflicts among NFRs, and the catalogue of conflicts among NFRs with respect to NFRs relative characteristic have also been published [9, 15, 26], however, several important tasks remain to complete the framework:

1) Elaborate the security-usability conflicts ontology

In the next step, we are going to continue developing the framework by elaborating the conflicts ontology through collecting information from literature. We will characterize the meaning of security and usability requirements in term of the conflict relationships among them with respect to the context of the system being developed; the impact of the conflicts; and the potential strategy to resolve the conflicts. A knowledge-based of security-usability conflicts will be the outcome of this step.

2) Enrich and refine the ontology

To enrich the ontology and to discover the insight from practitioners, we also plan to do the survey of practice. This survey of practice will be conducted to perform a triangulation of the existing potential conflict models and the framework to manage the conflicts among NFRs, particularly security-usability conflicts with respect to NFRs relative characteristic.

3) Develop tool support

To facilitate the framework utilization, we also plan to develop a tool that can assist software developers, particularly requirements engineers to perform managing conflicts between security and usability requirements.

4) Framework empirical evaluation

The framework will be evaluated through controlled experiments. The reason is because "controlled experiments make possible the careful observation and precise manipulation of independent variables (e.g. proposed framework), allowing for greater certainty, and encourage the researcher to try out novel frameworks in a safe and exploratory environment before implementing them in the real world settings" [41]. Effectiveness and efficiency will be used as the evaluation criteria. Effectiveness represents that this framework can be used to manage the NFRs conflicts, i.e. security-usability conflicts, by considering NFRs relative characteristic while efficiency represents how fast people can identify the conflicts using the framework.

Acknowledgment

We would like to thank The International Schlumberger Foundation Paris for funding this research through Faculty for the Future Award Program.

References

[1] L. Chung, B. A. Nixon, E. Yu, and J. Mylopoulos, *Non-functional requirements in software engineering*. Massachusetts: Kluwer Academic Publishers, 2000.

[2] C. Ebert, "Putting requirement management into praxis: dealing with nonfunctional requirements," *Information and Software Technology,* vol. 40, pp. 175-185, 1998.

[3] D. Firesmith, "Using quality models to engineer quality requirements," *Journal of Object Technology,* vol. 2, pp. 67-75, 2003.

[4] G. Kotonya and I. Sommerville, *Non-functional requirements*, 1998.

[5] R. T. Mittermeir, N. Roussopoulos, R. T. Yeh, and P. A. Ng, *Modern software engineering, foundations and current perspectives*. New York, NY, USA: Van Nostrand Reinhold Co, 1989.

[6] K. K. Breitman, J. C. S. Prado Leite, and A. Finkelstein, "The world's a stage: a survey on requirements engineering using a real-life case study," *Journal of the Brazilian Computer Society,* vol. 6, pp. 1-57, 1999.

[7] B. Boehm and H. In, "Identifying quality-requirements conflict," *IEEE Software,* vol. 13, pp. 25-35, 1996.

[8] N. G. Leveson and C. S. Turner, "An investigation of the Therac-25 accidents," *IEEE Computer,* vol. 26, pp. 18-41, 1993.

[9] D. Mairiza, D. Zowghi, and N. Nurmuliani, "An investigation into the notion of non-functional requirements," in *25th ACM Symposium On Applied Computing* Switzerland, 2010.

[10] K. E. Wiegers, *Software requirements*, 2nd ed. Washington: Microsoft Press, 2003.

[11] D. J. Grimshaw and G. W. Draper, "Non-functional requirements analysis: deficiencies in structured methods," *Information and Software Technology,* vol. 43, pp. 629-634, 2001.

[12] N. Heumesser, A. Trendowicz, D. Kerkow, H. Gross, and L. Loomans, "Essential and requisites for the management of evolution - requirements and

incremental validation," Information Technology for European Advancement, ITEA-EMPRESS consortium 2003.

[13] N. Yusop, D. Zowghi, and D. Lowe, "The impacts of non-functional requirements in web system projects," *International Journal of Value Chain Management* vol. 2, pp. 18-32, 2008.

[14] S. Lauesen, *Software requirements: styles and techniques*: Addison-Wesley, 2002.

[15] D. Mairiza, D. Zowghi, and N. Nurmuliani, "Managing conflicts among non-functional requirements," in *12th Australian Workshop on Requirements Engineering (AWRE '09)*, Sydney, Australia, 2009.

[16] L. Chung, B. A. Nixon, and E. Yu, "Dealing with change: an approach using non-functional requirements," *Requirements Engineering,* vol. 1, pp. 238-260, 1996.

[17] B. Curtis, H. Krasner, and N. Iscoe, "A field study of the software design process for large systems," *Communication of the ACM,* vol. 31, pp. 1268-1287, 1988.

[18] B. Boehm and A. Egyed, "WinWin requirements negotiation processes: a multi-project analysis," in *5th International Conference on Software Processes*, 1998.

[19] A. Egyed and B. Boehm, "A comparison study in software requirements negotiation," in *8th Annual International Symposium on Systems Engineering (INCOSE'98)*, 1998.

[20] W. N. Robinson, S. D. Pawlowski, and V. Volkov, "Requirements interaction management," *ACM Computing Surveys,* vol. 35, pp. 132-190, 2003.

[21] B. Paech and D. Kerkow, "Non-functional requirements engineering - quality is essential," in *10th International Workshop on Requirements Engineering: Foundation for Software Quality*, 2004, pp. 27-40.

[22] B. Boehm and H. In, "Aids for identifying conflicts among quality requirements," *IEEE Software, March 1996,* 1996.

[23] H. In, B. Boehm, T. Rodgers, and M. Deutsch, "Aplying WinWin to quality requirements: a case study," in *23rd International Conference on Software Engineering*, Toronto, Ontario, Canada, 2001, pp. 555 - 564.

[24] A. Egyed and P. Grünbacher, "Identifying requirements conflicts and cooperation: how quality attributes and automated traceability can help," *IEEE Software,* vol. 21, pp. 50 - 58, 2004.

[25] Y. Guan and A. K. Ghose, "Use constraint hierarchy for non-functional requirements analysis," *Lecture Notes in Computer Science,* vol. 3579/2005, pp. 104-109, 2005.

[26] D. Mairiza, "Towards a catalogue of conflicts among non-functional requirements," in *5th International Conference on Evaluation of Novel Approaches to Software Engineering (ENASE 2010)* Athens, Greece: INSTICC, 2010.

[27] K. Yee, "Aligning security and usability," *IEEE Security & Privacy,* vol. 2, pp. 48-55, 2004.

[28] L. F. Cranor and S. Garfinkel, "Secure or usable?," *IEEE Security & Privacy,* vol. 2, pp. 16-18, 2004.

[29] P. Gutmann and I. Grigg, "Security usability," *IEEE Security & Privacy,* vol. 3, pp. 56-58, 2005.

[30] A. J. DeWitt and J. Kuljis, "Is usable security an oxymoron?," *Interaction,* vol. 13, pp. 41 - 44, 2006.

[31] C. Braz, A. Seffah, and D. M. Raihi, "Designing a trade-off between usability and security: a metrics based-model," *Lecture Notes in Computer Science,* vol. 4663, pp. 114-126, 2007.

[32] ISO/IEC, "Software engineering - product quality - part 1: quality model." vol. ISO/IEC 9126-1: 2001 (E) Switzerland: ISO/IEC, 2001.

[33] "IEEE Std 830-1998: IEEE recommended practice for software requirements specifications," I. The Institute of Electrical and Electronics Engineers, Ed. USA, 1998.

[34] D. M. Pisanelli, A. Gangemi, and G. Steve, "Ontologies and information systems: the marriage of the century?," in *LYEE Workshop*, 2002.

[35] C. Calero, F. Ruiz, and M. Piattini, *Ontologies for software engineering and software technology*: Springer, 2006.

[36] M. Uschold and M. Gruninger, "Ontologies: principles, methods, and applications," *Knowledge Engineering Review,* vol. 11, pp. 93 - 115, 1996.

[37] M. Uschold and R. Jasper, "A framework for understanding and classiying ontology applications," in *IJCAI Workshop on Ontologies and Problem-Solving Methods*, 1999.

[38] R. Kishore, H. Zhang, and R. Ramesh, "A helix-spindle model for ontological engineering," *Communication of the ACM,* vol. 47, pp. 69 - 75, 2004.

[39] R. Kishore and R. Sharman, "Computational ontologies and information systems I: foundations," *Communications of the Association for Information Systems,* vol. 14, pp. 158 - 183, 2004.

[40] A. M. Hickey and A. Davis, "An ontological approach to requirements elicitation technique selection," in *Ontologies: a handbook of principles, concepts and applications in information systems*, R. Sharman, R. Kishore, and R. Ramesh, Eds.: Springer Science+Business Media, LLC, 2007, pp. 403 - 431.

[41] D. Damian, "Empirical studies of computer support for distributed requirements negotiation," University of Calgary, 2001.