

TOWARDS A CATALOGUE OF CONFLICTS AMONG NON-FUNCTIONAL REQUIREMENTS

Dewi Mairiza, Didar Zowghi and Nurie Nurmuliani
Faculty of Engineering and Information Technology
University of Technology, Sydney, Australia

Keywords: Non-Functional Requirements (NFRs), Conflicts, Catalogue, Relative, Relationship.

Abstract: Two of the most significant characteristics of non-functional requirements (NFRs) are “interacting” and “relative”. Interacting means NFRs tend to interfere, conflict, and contradict with one other while relative means the interpretation of NFRs may vary depending on many factors, such as the context of the system being developed and the extent of stakeholder involvement. For the purpose of understanding the interacting characteristic of NFRs, several potential conflict models have been presented in the literature. These models represent the positive or negative inter-relationships among various types of NFRs. However, none of them deal with the relative characteristic of NFRs. In fact, multiple interpretations of NFRs in the system being developed may lead to positive or negative inter-relationships that are not always obvious. As a result, the existing potential conflict models remain in disagreement with one other. This paper presents the result of an extensive and systematic investigation of the extant literature over the notion of NFRs and conflicts among them. A catalogue of NFRs conflicts with respect to the NFRs relative characteristic is presented. The relativity of conflicts is characterized by three categories: *absolute conflict*; *relative conflict*; and *never conflict*. This catalogue could assist software developers in identifying the conflicts among NFRs, performing further conflict analysis, and identifying the potential strategies to resolve those conflicts.

1 INTRODUCTION

In the early eighties, the term Non-Functional Requirements (NFRs) was introduced as the requirements that restrict the type of solution that a software system might consider (Yeh 1982). However, although this term has been in use for almost three decades, studies to date indicate that currently there is still no general consensus in the software engineering community regarding the notion of NFRs. In the literature, the term NFRs is considered within two different perspectives: (1) NFRs as the requirements that describe the properties, characteristics or constraints that a software system must exhibit; and (2) NFRs as the requirements that describe the quality attributes that the software product must have (Mairiza, Zowghi & Nurmuliani 2009).

In software development, NFRs are recognized as a very critical factor to the success of software projects. NFRs address the essential issue of the quality of the system (Chung et al. 2000; Ebert 1998; Firesmith 2003). Without well-defined NFRs,

a number of potential problems may occur, such as a software which is inconsistent and of poor quality; dissatisfaction of clients, end-users, and developers toward the software; and causing time and cost overrun for fixing the software (Chung et al. 2000). NFRs are also considered as the constraints or qualifications of the operations (Mittermeir et al. 1989). NFRs place restrictions on the product being developed, development process, and specify external constraints that the product must exhibit (Kotonya & Sommerville 1998). Furthermore, Charette (1990) claims that NFRs are often more critical than individual Functional Requirements (FRs) in the determination of a system's perceived success or failure (Sommerville 2004; Wiegiers 2003). Neglecting NFRs has led to a series of software failures. For example systemic failure in London Ambulance System (Breitman, Prado Leite & Finkelstein 1999; Finkelstein & Dowell 1996), performance and scalability failure in the New Jersey Department of Motor Vehicles Licensing System (Boehm & In 1996b), failure in the initial design of the ARPANet Interface Message Processor

Mairiza D., Zowghi D. and Nurmuliani N. (2010).

TOWARDS A CATALOGUE OF CONFLICTS AMONG NON-FUNCTIONAL REQUIREMENTS.

In *Proceedings of the Fifth International Conference on Evaluation of Novel Approaches to Software Engineering*, pages 20-29

Copyright © SciTePress

Software (Boehm & In 1996a), and some other examples as described in (Boehm & In 1996a, 1996b; Breitman, Prado Leite & Finkelstein 1999; Leveson & Turner 1993).

Although NFRs are widely recognized to be very significant in the software development, a number of empirical studies reveal that NFRs are often neglected, poorly understood and not considered adequately in developing a software application. In the development of software system, users naturally focus on specifying their functional or behavioral requirements, i.e. the things the product must do (Chung et al. 2000; Wiegers 2003). NFRs are often overlooked in the software development process (Ebert 1998; Grimshaw & Draper 2001). A number of studies investigating practices of dealing with NFRs in the software industry also reported that commonly software developers do not pay sufficient attention to NFRs (Ebert 1998; Grimshaw & Draper 2001; Heumesser et al. 2003; Yusop, Zowghi & Lowe 2008). 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 (Heumesser et al. 2003; Yusop, Zowghi & Lowe 2008). Furthermore, in the requirements engineering literature, NFRs have received less attention and not as well understood as FRs (Chung et al. 2000). Majority of software engineering research, particularly within requirements engineering area only deal with FRs, i.e. ensuring that the necessary functionality of the system is delivered to the user (Paech & Kerkow 2004). Consequently, capturing, specifying, and managing NFRs are still difficult to perform due to most of software developers do not have adequate knowledge about NFRs and little help is available in the literature (Lauesen 2002).

NFRs tend to interfere, conflict, and contradict with one another. Unlike FRs, this inevitable conflict arises as a result of inherent contradiction among various types of NFRs (Chung et al. 2000; Ebert 1998). Certain combinations of NFRs in the software system may affect the inescapable trade offs (Boehm & In 1996b; Ebert 1998; Wiegers 2003). Achieving a particular type of NFRs can hurt the achievement of the other type(s) of NFRs. Hence, this conflict is widely acknowledged as one of many characteristics of NFRs (Chung et al. 2000).

Prior studies reveal that dealing with NFRs conflicts is essential due to several reasons (Mairiza, Zowghi & Nurmuliani 2009). First of all, conflicts among software requirements are inevitable (Chung, Nixon & Yu 1995; Chung, Nixon & Yu 1996; Chung et al. 2000; Curtis, Krasner & Iscoe 1988).

Conflicting requirements are one of the three main problems in software development in term of the additional effort or mistakes attributed to them (Curtis, Krasner & Iscoe 1988). A study of two-year multiple-project analysis conducted by Egyed & Boehm (Boehm & Egyed 1998; Egyed & Boehm 1998) 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 (Robinson, Pawlowski & Volkov 2003). Lessons learnt from industrial practices also confirm that one of the essential aspects during NFRs specification is management of conflict among interacting NFRs (Ebert 1998). Experience shows most systems suffer with severe tradeoffs among the major groups of NFRs. For example: the tradeoffs between security and performance requirements; or between security and usability requirements. In fact, conflict resolutions for handling NFRs conflicts often results in changing overall design guidelines, not by simply changing one module (Ebert 1998). Therefore, since conflict among NFRs has also been widely acknowledged as one of NFRs characteristics, managing this conflict as well as making this conflict explicit is important (Paech & Kerkow 2004). NFRs conflicts management is important for finding the right balance of attributes satisfaction, in achieving successful software products (Boehm & In 1996b; Wiegers 2003).

A review of various techniques to manage conflicts among NFRs have been presented in the literature (Mairiza, Zowghi & Nurmuliani 2009). Majority of these techniques provide a documentation, catalogue, or list of potential conflicts among various types of NFRs. These catalogues represent the interrelationships among those NFRs types. Apart from strength and weaknesses of each technique, however, NFRs are also relative (Chung et al. 2000). 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 being developed as well as the extent of stakeholder involvement. Consequently, the positive or negative relationships among those NFRs types 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 of NFRs, existing potential conflict models that represent the relationship among NFRs are often in disagreement with each other. For example,

according to Wieggers (2003), efficiency requirements have negative relationship (conflict) with usability requirements, but according to Egyed & Grünbacher (2004), these two types of NFRs have positive relationship (support). Given that none of the existing conflicts catalogues deal with the relative characteristics of NFRs, we are motivated to pose the following research question:

“Can we develop a catalogue of conflicts among NFRs with respect to the relative characteristic of NFRs?”

A catalogue of conflicts with respect to the NFRs relative characteristic is presented as the novel contribution of this paper. This catalogue is developed as a two-dimensional conflict-relationship between various types of NFRs. It represents the relationship between each NFR type, such as how each type of NFR is associated with the other types of NFRs considering the NFRs relative characteristic. The conflict-relationships are represented in three categories: *absolute conflict*; *relative conflict*; and *never conflict*.

This paper is organized in six sections. The first section is the introduction to NFRs and conflicts among them. The second section describes the research framework and source of information used in this study. The superset list of NFRs is presented in section three continued by presenting the catalogue of NFRs conflicts in section four. Section five describes the benefits and potential applications of the conflicts catalogue in the software development projects. Then, section six concludes this paper by highlighting some open issues which are acquired from the investigation.

2 CATALOGUE FRAMEWORK

To get a significant and comprehensive snapshot of the NFRs conflicts model, an extensive investigation of the literature over the last three decades has been performed. This investigation was conducted by exploring the articles from academic resources and documents from software development industry. Four general types of sources of information have been identified: (1) journal papers; (2) conference proceedings; (3) books; and (4) documents from software industry. Selection of those sources is made in order to confirm the completeness of the information by obtaining the academics and practitioners perspectives related to the notion of NFRs and conflicts among them. The study conducted by Chung et al. (2000) was used as the

starting point for selection of the papers to be reviewed.

Our study has examined 182 sources of information. All of them are literatures within the discipline of software engineering. They cover various issues of NFRs and conflicts among them. The research articles reviewed are published in key journals and conference proceedings of the software engineering literature, such as the Journal of Systems and Software; IEEE Transaction on Software Engineering; IEEE Software; Lecture Notes in Computer Science; Journal of Information and Software Technology; Requirements Engineering Journal; Requirements Engineering Conference, International Conference on Software Engineering, and Requirements Engineering Foundations of Software Quality Workshop.

Each source was then systematically analyzed using content analysis technique. Content analysis is a research technique that uses a set of procedures to make valid inferences from texts or other meaningful matter (Krippendorff 2004; Weber 1989). This technique is well-founded and has been in used for over sixty years. The analysis covers three essential issues: the NFRs types, the definition and attributes¹ of each type, and the conflict interdependencies among these types. Content analysis technique was selected because it enables researchers to identify trends and patterns in the literature through the frequency of key words, and by coding and categorizing the data into a group of words with similar meaning or connotations (Stemler 2001; Weber 1989). This technique is also applicable to all domain contexts (Krippendorff 2004; Neuendorf 2001).

To develop a catalogue of NFRs conflicts, a research framework was followed. This framework consists of three research stages as follows:

- (1) To create a comprehensive catalogue of NFRs types, their definition and attributes characterization
- (2) To identify the interdependencies among various NFRs types
- (3) To perform a normalization process to standardize the NFRs in the conflicts catalogue.

¹ In this paper, the term attribute is considered as the major components of each NFRs type. In the literature, attribute is also referred as NFRs subtypes (Chung et al. 2000) or quality sub factors (Firesmith 2003).

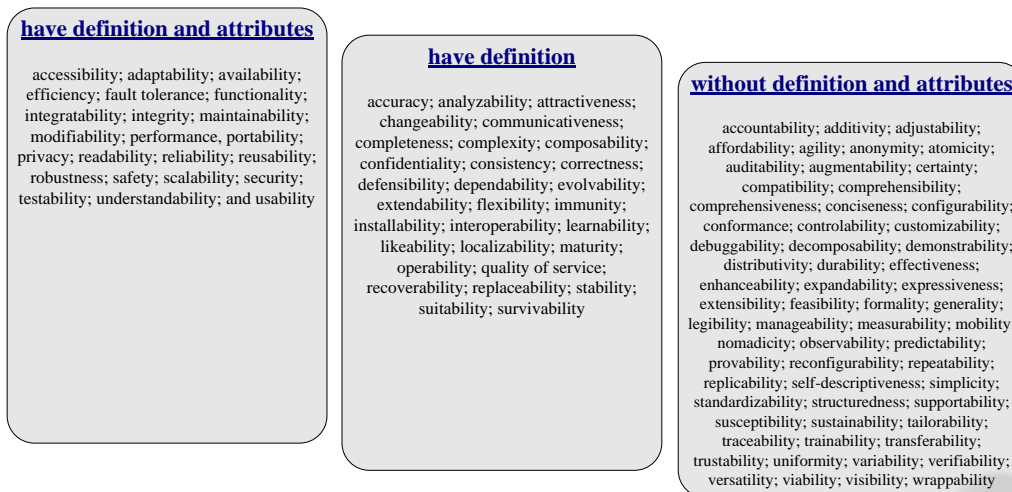


Figure 1: NFRs Types in the Literature.

Since there is no standard catalogue of NFRs types available in the literature and previous studies (Glinz 2005, 2007; Mairiza, Zowghi & Nurmuliani 2010) also claimed that many types of NFRs were introduced without definition or attributes characterization, the first stage of the research was creating a comprehensive catalogue of NFRs types. Each type of NFR discussed in the literature was recorded. The definition and attributes correspond to each of NFR type were also documented. Conflicting terminologies and definitions were handled through the frequency analysis technique and keywords identification.

The second stage of the research was creating an initial catalogue of the conflicts among NFRs. In this stage, NFRs conflict relationships were used as the criteria to develop the catalogue. This stage was initiated by identifying the interdependencies among various types of NFRs. These interdependencies represent the typical interrelationships of a particular type of NFR towards another type of NFR (e.g. positive, negative, or neutral interrelationships). This investigation produced the initial catalogue that presents the conflict relationships among 26 types of NFRs. These NFRs types are listed in Table 1.

The next stage was performing a normalization process against 26 types of NFRs that have been identified in the initial catalogue. This normalization was conducted in order to standardize the data obtained in the previous stage.

Normalization is the process of removing the irrelevant NFRs, i.e. the types of NFRs that do not have definition and/or attributes, from the initial catalogue.

The objective is to produce a conflicts catalogue of the well-defined NFRs types.

Table 1: NFRs Types in the Initial Catalogue.

NFRs Types		
Accuracy	Interoperability	Reliability
Analyzability	Legibility	Reusability
Availability	Maintainability	Robustness
Compatibility	Performance	Safety
Confidentiality	Portability	Security
Dependability	Privacy	Testability
Expressiveness	Provability	Understandability
Flexibility	Recoverability	Usability
Functionality		Verifiability

In this normalization, the catalogue of NFRs types, their definition, and their attributes are utilized as the basis of removing those irrelevant NFRs. This process has removed six NFRs from the initial catalogue. They are *compatibility*, *expressiveness*, *legibility*, *provability*, *verifiability* and *analyzability*. Therefore, the final conflicts catalogue is a two-dimensional matrix that represents the conflicts interrelationships among 20 types of “normalized” NFRs.

3 NFRs TYPES

Various authors (e.g. Chung et al. (2000), Alexander & Maiden (2004), and Robertson & Robertson (2006)) define the term NFRs as the requirements that specify the desired quality attributes of the system. According to this definition, our analysis of NFRs types in the literature has resulted in identification of 114 types of NFRs.

Table 2: NFRs Definition and Attributes (Mairiza, Zowghi, & Nurmuliani 2010).

NFRs	Definition	Attributes
Performance	requirements that specify the capability of software product to provide appropriate performance relative to the amount of resources needed to perform full functionality under stated conditions	response time, space, capacity, latency, throughput, computation, execution speed, transit delay, workload, resource utilization, memory usage, accuracy, efficiency compliance, modes, delay, miss rates, data loss, concurrent transaction processing
Reliability	requirements that specify the capability of software product to operates without failure and maintains a specified level of performance when used under specified normal conditions during a given time period	completeness, accuracy, consistency, availability, integrity, correctness, maturity, fault tolerance, recoverability, reliability, compliance, failure rate/critical failure
Usability	requirements that specify the end-user-interactions with the system and the effort required to learn, operate, prepare input, and interpret the output of the system	learnability, understandability, operability, attractiveness, usability compliance, ease of use, human engineering, user friendliness, memorability, efficiency, user productivity, usefulness, likeability, user reaction time
Security	requirements that concern about preventing unauthorized access to the system, programs, and data	confidentiality, integrity, availability, access control, authentication
Maintainability	requirements that describe the capability of the software product to be modified that may include correcting a defect or make an improvement or change in the software	testability, understandability, modifiability, analyzability, changeability, stability, maintainability compliance

The superset list of these 114 NFRs types can be found in our previous publication (Mairiza, Zowghi & Nurmuliani 2010).

Further investigation to the superset list indicates that 23 NFRs types (20.18%) have definition and attributes, 30 types (26.32%) only have definition, and the rest 61 types (53.50%) were introduced without definition or attributes. Since this finding indicates that more than 50% of NFRs types listed in the literature do not have any definitions and attributes characterization, therefore, it confirms the previous claim made by Glinz (2005, 2007) which stated that “in the literature, many NFRs were introduced without definition or clarifying examples”. The detailed list of these classifications is presented in Figure 1.

The top five of the most frequently discussed NFRs types in the literature are presented in Table 2. These top five NFRs were identified by performing the frequency analysis techniques towards each NFR type listed in the literature. The definitions and attributes are decomposed by integrating the existing definitions and attributes based on general complementary description stated in the scholarly literatures.

4 CATALOGUE OF CONFLICTS

The catalogue of conflicts is a two-dimensional matrix that represents the typical interrelationships among 20 types of normalized NFRs, in term of the conflict emerges among them. In this catalogue, the relativity of NFRs conflicts is presented in three categories: *absolute conflict*; *relative conflict*; and *never conflict* (as presented in Figure 2).

- **Absolute Conflict:** this relationship represents a pair of NFRs types that are always in conflict. In the catalogue, this conflict relationship is labeled as ‘X’.
- **Relative Conflict:** this relationship represents a pair of NFRs types that are sometimes in conflict. It consists of all pairs of NFRs that are claimed to be in conflict in the literature but they are also claimed as not being in conflict in the other cases. This disagreement occurs due to several factors, such as the different interpretation/meaning of NFRs in the system being developed, the context of the system, the stakeholder involvement, and the architectural design strategy implemented in that system. In the conflict catalogue, this type of conflict relationship is labeled as ‘*’.

NFRs	Accuracy	Availability	Confidentiality	Dependability	Flexibility	Functionality	Interoperability	Maintainability	Performance	Portability	Privacy	Recoverability	Reliability	Reusability	Robustness	Safety	Security	Testability	Understandability	Usability
Accuracy	0		*			0		0	X	X		0	0				0			0
Availability		0							X		X	0				0	X			0
Confidentiality	*		0						X								0			
Dependability				0					X											
Flexibility					0															X
Functionality	0				0		*	*				0	*				*			0
Interoperability						0		X												
Maintainability	0				*	X	0	X			0	0		X		0				0
Performance	X	X	X	X	*	X	X	*	X	X	*	*	X		X	X	X	X	X	*
Portability	X								X											
Privacy		X																		
Recoverability	0	0			0	0	*				0	0					0			0
Reliability	0				*	0	*				0	0				0	0			0
Reusability								X												X
Robustness							X								0			X		
Safety		0							X			0			0					
Security	0	X	0		*	0	X				0	0					0			*
Testability															X					
Understandability									X								*			
Usability	0				X	0	0	0	*		0	0	X				*			0

Figure 2: Catalogue of Conflicts among NFRs.

- Never Conflict:** this relationship represents a pair of NFRs types that are never in conflict in the software development projects. It consists of all pairs of NFRs who have never been declared as being in conflict with each other. They may contribute either positively (e.g. support (Sadana & Liu 2007) or cooperative (Egyed & Grünbacher 2004)) or indifferent to one another (e.g. low or very little impact on the other (Wiegiers 2003)). In the conflict catalogue, this conflict relationship is labeled as 'O'.

Further analysis of this conflicts catalogue indicates that 36 pairs of NFRs are absolute conflict (e.g. accuracy and performance; security and performance; and usability and reusability); 19 pairs are relative conflict (e.g. reliability and performance; usability and security; and performance and usability); and 50 pairs are never conflict (e.g. accuracy and maintainability; security and accuracy; and usability and recoverability). The rest of relationships are not known due to there is no information available in the literature about how those pairs of NFRs contribute to each other. In the conflicts catalogue, it is presented as "the blank spaces".

Furthermore, this catalogue shows that NFRs with the most conflict with other NFRs is performance. Performance has absolute conflict with accuracy, availability, confidentiality, dependability, interoperability, maintainability, portability, reusability, safety, security, and understandability,

and it has relative conflict with functionality, recoverability, reliability, and usability.

The investigation also indicates that certain attributes of a particular type of NFR can be in conflicts with each other. These conflicts point to the self-conflicting relationships for a particular type of NFR. Self-conflicting relationship is defined as a situation where the attributes of a single type of NFR are in conflict. For example, the relative conflict between performance and performance requirements. Performance requirements can be characterized among others by "response time" and "capacity". In many systems, these two attributes are in conflict. For example in a road traffic pricing system (Brito & A. 2004; Moreira, Araujo & Brito 2002), multi-user attribute² has negative contribution to the response time of the system. This means that increasing the number of concurrent users in the system may diminish the response time of the system.

The investigation by using frequency analysis technique also indicates that conflicts between security and performance requirements are the most frequently conflicts discussed in the literature. 33.33% of the reviewed articles talk about this conflict, followed by conflicts between security and usability requirements (23.33%) and conflicts between availability and performance requirements (20%). This result indicates that those three types of conflicts (i.e. conflicts between security and

² In these papers (Brito & A. 2004; Moreira, Araujo & Brito 2002), the term "attribute" is considered as "concern".

performance, between security and usability, and between availability and performance) are the three most frequent conflicts in the software projects and the most considered and essential to deal with in the software development process. The top ten conflicting NFRs that are often discussed in the literature are presented in Table 3.

Table 3: Conflicting NFRs in Literature.

Conflicting NFRs	Nature of Conflict	%
Security and Performance	absolute	33%
Security and Usability	relative	23%
Availability and Performance	absolute	20%
Performance and Portability	absolute	17%
Reusability and Performance	absolute	17%
Interoperability and Performance	absolute	10%
Maintainability and Performance	absolute	10%
Reliability and Performance	relative	10%
Usability and Performance	relative	10%
Usability and Reusability	absolute	3%

5 USING THE CATALOGUE

The catalogue of conflicts among NFRs, as presented in Figure 2, extends and complements previously published NFRs conflict models. Our work focuses on the extent and relativity of NFRs conflicts, that is, on negative links between NFRs and its corresponding level. Most of the existing conflict models in the literature, however, concentrate on both positive and negative interrelationships. For example, Wiegers (2003) has developed a relationship matrix that represents the positive and negative relationships between particular type of NFRs; Egyed & Grünbacher (2004) created a model of potential conflict and cooperation among NFRs; and Sadana & Liu (2007) have also defined conflict relationship and support relationship as the two types of contribution of a particular type of NFR on the other types of NFRs.

Utilizing our NFRs catalogue of conflicts in conjunction with the existing conflict models extends the overall understanding of how NFRs associate with each other (positive or negative) and how this negative association can be characterized in term of the relative characteristic of NFRs.

Our NFRs catalogue of conflicts could be used by software developers in dealing with conflicts among NFRs. The conflicts catalogue can be used to identify which NFRs of the system that are really in

conflict, including how relative the conflicts are in term of the relative characteristic of NFRs. If the conflict identified is an “absolute conflict”, then software developers may need to identify the potential strategies to resolve this conflict, such as the prioritization. On the other hand, if it is a “relative conflict”, then software developers need to understand and evaluate this particular NFR in term of numerous factors involve in the development project, such as the meaning of this particular type of NFR in the context of the system being developed; the stakeholder involvement; or system development methodology used in the project, in order to further investigate whether those NFRs are really in conflict.

This catalogue can also be used to analyze the conflicts among NFRs. By using this catalogue in conjunction with the framework presented by Sadana & Liu (2007), software developers would be able to develop a structural hierarchy of functional and non-functional requirements affected by each conflict type. Therefore, this catalogue could further assist in the analysis of NFRs conflicts from the perspective of functional requirements.

By utilizing this catalogue in conjunction with the “NFR Prioritizer” method presented by Mala & Uma (2006), this catalogue would assist software developers to analyze the tradeoffs among NFRs and prioritize the NFRs. In term of analyzing the NFRs tradeoff, this catalogue can be used as the basis to develop the “NFR Taxonomy” that will be used to identify the type of relationships among NFRs. The NFR Taxonomy represents the conflicting or dependable association between each NFR type. The example of NFR taxonomy is presented as follow (Mala & Uma 2006):

$$Usability\#Accessibility+\#Installability+\#Operability+\#Maintainability-$$

This taxonomy represents that usability contributes positively to accessibility, installability, operability while it also contributes negatively to maintainability. Then, by combining the weight of user preference on each NFR type and the level of NFRs tradeoff derived from the NFR Taxonomy, software developers would be able to prioritize the NFRs of the system in term of the existence of conflicts among them.

Furthermore, this catalogue can also be used in conjunction with the “Trace Analyzer” technique developed by Egyed & Grünbacher (2004). The aim of this technique is to identify the true conflicts among NFRs. By tracing the relationship between

the system test case and the software program code, trace analyzer can analyze whether the conflicts listed in the NFRs conflict catalogue are “really in conflict” in the developed system.

In term of resolution, the catalogue of conflicts can also be used to resolve the conflicts among NFRs. For example, by using this catalogue in conjunction with the “Non-Functional Decomposition (NFD)” framework developed by Poort & de With (2004), software developers would be able to decompose the NFRs of the system when the NFRs conflicts identified.

6 DISCUSSION & CONCLUSIONS

Majority of techniques to manage the NFRs conflicts provide the NFRs documentation, catalogue, or list of potential conflicts among various types of NFRs. However, none of them deal with relative characteristic of NFRs. In fact, it is widely acknowledged that NFRs are also relative. 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 them may also vary depending on the particular system being developed as well as the extent of stakeholders’ involvement. Consequently, the positive or negative relationships among them are not always obvious.

In this paper we present a catalogue of conflicts among NFRs by considering this relative characteristic. We present the relativity of conflicts based on three categories: *absolute conflict*; *relative conflict*; and *never conflict*. This distinction would assist developer to perform further analysis towards the conflict identified and investigate the potential strategy to resolve the conflict.

This catalogue can be used to identify the conflicts among NFRs in various phase of software development project. For example, in the requirements engineering phase, during the elicitation process, system analysts would be able to identify which NFRs of the system that will be in conflict and how relative this conflict is. This analysis would allow developers to identify the conflicts among NFRs early and to discuss this potential conflict with the system’s stakeholder before specifying the software requirements. As another example, during the architecture design process, system designers would also be able to use this catalogue to analyze the potential conflict among NFRs in term of the architecture decision (e.g. layering, clustering, and modularity). The

relativity of conflict relationship presented in the catalogue, would allow system designers to investigate the potential architecture strategies to get the best solution based on the type of conflicts among NFRs. Furthermore, by using this catalogue as the basis of conflict identification, we can adopt numerous existing conflict analysis and conflict resolution techniques, such as (Egyed & Grünbacher 2004; Mala & Uma 2006; Poort & de With 2004; Sadana & Liu 2007) to further investigate and evaluate the NFRs conflicts. Some examples of the existing techniques and the potential utilization of the catalogue in these techniques have been described in Section 5 – “Using the Catalogue”.

In the process of investigating conflicts and developing the conflicts catalogue, we also identified 114 NFRs types listed in the literature. Among these 114 types, more than 50% NFRs were introduced without any definition or attributes characterization while only 20% were provided with definition and attributes characterization. This statistic and the list of NFRs types without definitions and attributes presented in this paper are expected to encourage software engineering community, particularly requirements engineering community to further investigate the unclear NFRs types and establish the clear concept about these types of NFRs.

Further research will focus on collecting data from software practitioners to complement the catalogue. Those NFRs that have been removed from the initial catalogue due to lack of definitions and/or attributes will also be further investigated to improve the completeness of the catalogue. The catalogue from industry can also be compared with the one developed from the content analysis.

Moreover, besides collecting data to develop the conflicts catalogue, we would also perform further research on investigating the relative conflicts among NFRs. This study would not only investigate how those NFRs dynamically generate conflicts with each other in term of the context-based of the system, but also to develop a framework to assist developers in identifying in which situations those NFRs are in conflict and in which situations are not. The self-conflicting relationships will be covered in this study.

This study is conducted as part of a long term project of investigating conflicts among NFRs. Findings of this investigation, especially the conflicts catalogue, will be used as the basis to select those NFRs that are known to be frequently in conflict. The ultimate goal is to develop an integrated framework to effectively manage the

conflicts among particular NFR by considering the NFRs relative characteristic. This framework should be able to identify not only the existence and the extent of conflict, but also to characterize and find the potential strategies to resolve the conflict.

In this study, we do not claim that the catalogue of conflicts presented is an exhaustive list. But, this catalogue represents what could be found in the current literature. We propose to conduct further research to compare and contrast our findings from the comprehensive review of research literature and the state of the practice.

ACKNOWLEDGEMENTS

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

REFERENCES

- Alexander, I. & Maiden, N. 2004, *Scenarios, stories, use cases: through the systems development life-cycle*, John Wiley & Sons, Ltd, Chichester, England.
- Boehm, B. & Egyed, A. 1998, 'WinWin requirements negotiation processes: a multi-project analysis', *5th International Conference on Software Processes*.
- Boehm, B. & In, H. 1996a, 'Aids for identifying conflicts among quality requirements', *IEEE Software*, March 1996.
- Boehm, B. & In, H. 1996b, 'Identifying quality-requirements conflict', *IEEE Software*, vol. 13, no. 2, pp. 25-35.
- Breitman, K.K., Prado Leite, J.C.S. & Finkelstein, A. 1999, 'The world's a stage: a survey on requirements engineering using a real-life case study', *Journal of the Brazilian Computer Society*, vol. 6, no. 1, pp. 1-57.
- Brito, I. & A., M. 2004, 'Integrating the NFR framework in a RE model', paper presented to the *Early Aspects: Aspect-Oriented Requirements Engineering and Architecture Design*, Lancaster, UK.
- Charette, R.N. 1990, *Applications strategies for risk analysis*, McGraw-Hill, New York.
- Chung, L., Nixon, B.A. & Yu, E. 1995, 'Using non-functional requirements to systematically support change', *The second international symposium on requirements engineering*, IEEE, York, pp. 132-139.
- Chung, L., Nixon, B.A. & Yu, E. 1996, 'Dealing with change: an approach using non-functional requirements', *Requirements Engineering*, vol. 1, no. 4, pp. 238-260.
- Chung, L., Nixon, B.A., Yu, E. & Mylopoulos, J. 2000, *Non-functional requirements in software engineering*, Kluwer Academic Publishers, Massachusetts.
- Curtis, B., Krasner, H. & Iscoe, N. 1988, 'A field study of the software design process for large systems', *Communication of the ACM*, vol. 31, no. 11, pp. 1268-1287.
- Ebert, C. 1998, 'Putting requirement management into praxis: dealing with nonfunctional requirements', *Information and Software Technology*, vol. 40, no. 3, pp. 175-185.
- Egyed, A. & Boehm, B. 1998, 'A comparison study in software requirements negotiation', *8th Annual International Symposium on Systems Engineering (INCOSE'98)*.
- Egyed, A. & Grünbacher, P. 2004, 'Identifying requirements conflicts and cooperation: how quality attributes and automated traceability can help', *IEEE Software*, vol. 21, no. 6, pp. 50 - 58.
- Finkelstein, A. & Dowell, J. 1996, 'A comedy of errors: the London ambulance service case study', *Eighth International Workshop Software Specification and Design*, pp. 2-5.
- Firesmith, D. 2003, 'Using quality models to engineer quality requirements', *Journal of Object Technology*, vol. 2, no. 5, pp. 67-75.
- Glinz, M. 2005, 'Rethinking the notion of non-functional requirements', *Third World Congress for Software Quality*, Munich, Germany, pp. 55-64.
- Glinz, M. 2007, 'On non-functional requirements', *15th IEEE International Requirements Engineering Conference (RE '07)*, IEEE, pp. 21-26.
- Grimshaw, D.J. & Draper, G.W. 2001, 'Non-functional requirements analysis: deficiencies in structured methods', *Information and Software Technology*, vol. 43, pp. 629-634.
- Heumesser, N., Trendowicz, A., Kerkow, D., Gross, H. & Loomans, L. 2003, *Essential and requisites for the management of evolution - requirements and incremental validation*, Information Technology for European Advancement, ITEA-EMPRESS consortium.
- Kotonya, G. & Sommerville, I. 1998, *Non-functional requirements*.
- Krippendorff, K. 2004, *Content analysis: and introduction to its methodology*, Second edn, Sage Publications, Inc., Thousand Oaks, USA.
- Lauesen, S. 2002, *Software requirements: styles and techniques*, Addison-Wesley.
- Leveson, N.G. & Turner, C.S. 1993, 'An investigation of the Therac-25 accidents', *IEEE Computer*, vol. 26, no. 7, pp. 18-41.
- Mairiza, D., Zowghi, D. & Nurmulliani, N. 2009, 'Managing conflicts among non-functional requirements', *12th Australian Workshop on Requirements Engineering (AWRE '09)*, Sydney.
- Mairiza, D., Zowghi, D. & Nurmulliani, N. 2010, 'An investigation into the notion of non-functional requirements', *25th ACM Symposium On Applied Computing ACM*, Switzerland.
- Mala, G.S.A. & Uma, G.V. 2006, 'Elicitation of non-functional requirements preference for actors of

- usecase from domain model', *Lecture Notes in Computer Science*, vol. 4303/2006, pp. 238-243.
- Mittermeir, R.T., Roussopoulos, N., Yeh, R.T. & Ng, P.A. 1989, *Modern software engineering, foundations and current perspectives*, Van Nostrand Reinhold Co, New York, NY, USA.
- Moreira, A., Araujo, J. & Brito, I. 2002, 'Crosscutting quality attributes for requirements engineering', *14th international conference on Software engineering and knowledge engineering*, ACM, Ischia, Italy.
- Neuendorf, K.A. 2001, *The content analysis guidebook*, First edn, Sage Publications, Inc.
- Paech, B. & Kerkow, D. 2004, *Non-functional requirements engineering - quality is essential*, *10th International Workshop on Requirements Engineering: Foundation for Software Quality*, pp. 27-40.
- Poort, E.R. & de With, P.H.N. 2004, 'Resolving requirement conflicts through non-functional decomposition', *Fourth Working IEEE/IFIP Conference on Software Architecture (WICSA '04)*, IEEE.
- Robertson, S. & Robertson, J. 2006, *Mastering the requirements process*, 2nd edn, Addison-Wesley, Boston.
- Robinson, W.N., Pawlowski, S.D. & Volkov, V. 2003, 'Requirements interaction management', *ACM Computing Surveys*, vol. 35, no. 2, pp. 132-190.
- Sadana, V. & Liu, X.F. 2007, 'Analysis of conflict among non-functional requirements using integrated analysis of functional and non-functional requirements', *31st International Computer Software and Applications Conference (COMPSAC 2007)*, IEEE.
- Sommerville, I. 2004, *Software Engineering*, 7 edn, Pearson Education Limited, Essex, England.
- Stemler, S. 2001, 'An overview of content analysis', *Practical Assessment, Research & Evaluation*, vol. 7, no. 17.
- Weber, R.P. 1989, *Basic content analysis*, Sage Publications, Inc.
- Wieggers, K.E. 2003, *Software requirements*, 2nd edn, Microsoft Press, Washington.
- Yeh, R.T. 1982, 'Requirements analysis - a management perspective', *COMPSAC '82*, pp. 410-416.
- Yusop, N., Zowghi, D. & Lowe, D. 2008, 'The impacts of non-functional requirements in web system projects', *International Journal of Value Chain Management* vol. 2, no. 1, pp. 18-32.