“© 2013 IEEE. Personal use of this material is permitted. Permission from IEEE must be obtained for all other uses, in any current or future media, including reprinting/republishing this material for advertising or promotional purposes, creating new collective works, for resale or redistribution to servers or lists, or reuse of any copyrighted component of this work in other works.”

Conflict Characterization and Analysis of Non Functional Requirements: An Experimental Approach

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

Vincenzo Gervasi
Dipartimento di Informatica
Università di Pisa, I-56125 Pisa, Italy
gervasi@di.unipi.it

***Abstract*—Prior studies reveal that conflicts among Non Functional Requirements (NFRs) are not always absolute. They can also be relative depending on the context of the system being developed. Given that existing techniques to manage the NFRs conflicts are mainly focused on cataloguing the interrelationships among various types of NFRs, hence a technique to manage the NFRs conflicts with respect to NFRs relative characteristic is needed. This paper presents a novel framework to manage the conflicts among NFRs with respect to NFRs relative characteristic. By applying an experimental approach, the quantitative evidence of NFRs conflicts will be obtained and modeled. NFRs metrics and measures will be used in the experiments as parameters to generate the quantitative evidence. This evidence can then allow developers to identify and reason about the NFRs conflicts. We also provide an example of how this framework could be applied.**

Keywords–non-functional requirements, conflict, relative, identification, characterization, analysis, management, framework, experiment.

I. INTRODUCTION

It is widely acknowledge in the literature that the correct implementation of Non-Functional Requirements is recognized as a critical factor to the success of software projects. NFRs address the essential issue of software quality [1-3]; and they are also considered as the qualifications of the operations [4, 5]. However, although NFRs have been a focus of attention by researchers and practitioners alike for almost three decades, studies to date indicate that there is not enough progress made in dealing with NFRs compared to Functional Requirements (FRs). Managing NFRs is still difficult to perform due to the fact that most software developers do not have adequate knowledge about NFRs and little help is available in the literature [6]. Capturing, specifying, and managing NFRs are still difficult to perform and NFRs are often poorly articulated in the software requirements document [7, 8].

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 [1, 2]. Certain combinations of NFRs in the software systems may affect the inescapable trade offs [2, 9, 10]. Achieving a particular type of NFRs can prevent the achievement of the other type(s) of NFRs.

Dealing with NFRs conflict is essential due to several reasons. Firstly, conflict among software requirements is inevitable [1, 12, 13]. Conflicting requirements are one of the three main problems in the software development in terms of the additional effort or mistakes attributed to them [13]. A study of two-year period multiple-project analysis conducted by Egyed & Boehm [14, 15] 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 [16]. 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 from severe tradeoffs among the major groups of NFRs. In fact, conflict resolutions for handling NFRs conflicts often result in changing overall design guidelines, not by simply changing one module.

II. PROBLEM DEFINITION

A number of techniques to manage conflict among NFRs have been discussed in the literature [11]. Majority of them provide documentation, catalogue, or list of potential conflict. These catalogues represent the interrelationships among various types of NFRs. Some examples are: the

QARCC win-win approach [10, 17, 18], trace analyzer of the requirements traceability technique [19], and a technique that adopts a hierarchical constraint logic programming approach [20]. Apart from strength and weaknesses of each technique, NFRs can be viewed, interpreted, and evaluated differently by different people and different context within which the system is being developed. 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 conflict 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.

Prior studies [11, 21-23] reveal that the relativity of NFRs conflict can be presented in three categories: *absolute conflict* (labeled as "X")*; relative conflict* (labeled as "*")*;* and *never in conflict* (labeled as "O"). As illustrated in Figure 1 nineteen pairs of NFRs in this catalogue have *relative conflict*, which means that they are not always in conflict because they are claimed to be in conflict in certain cases but not in conflict in others.

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

Figure 1 - Catalogue of Conflicts among NFRs [21]

Given the above context, we are motivated to perform a further investigation into the conflict among NFRs in general, in order to increase our understanding about how NFRs conflict with, and affect one another; and how this conflict might be managed. Our research question has been formulated as follow:

"With respect to the NFRs relative characteristic, how can we create a framework that can assist developers to identify and characterize the conflict among NFRs?"

A framework to characterize and analyze the relative conflict among NFRs is presented as the novel contribution of this paper. The framework utilizes an experimental-based approach as the foundation to characterize the conflict and to perform conflict decision analysis. NFRs metrics and measure are used as parameters to collect the

quantitative evidence and to model the NFRs conflict relationships.

III. An Experimental Approach for NFRs Conflict Management

In our previous work, we have proposed a preliminary framework to manage the relative conflict between two types of NFRs, which are security and usability requirements [24]. This framework focuses specifically on the application of ontology in managing the relative conflict between security and usability. By following the Helix-Spindle model for ontological engineering [25], an ontological model of the security-usability requirements conflict has been developed. This ontology shows *when security and usability are in conflict, what the impacts of the conflict are, and what the relevant strategies to resolve this conflict are*. Therefore this framework can be used as a basis to assist analysts in managing conflict between security and usability requirements.

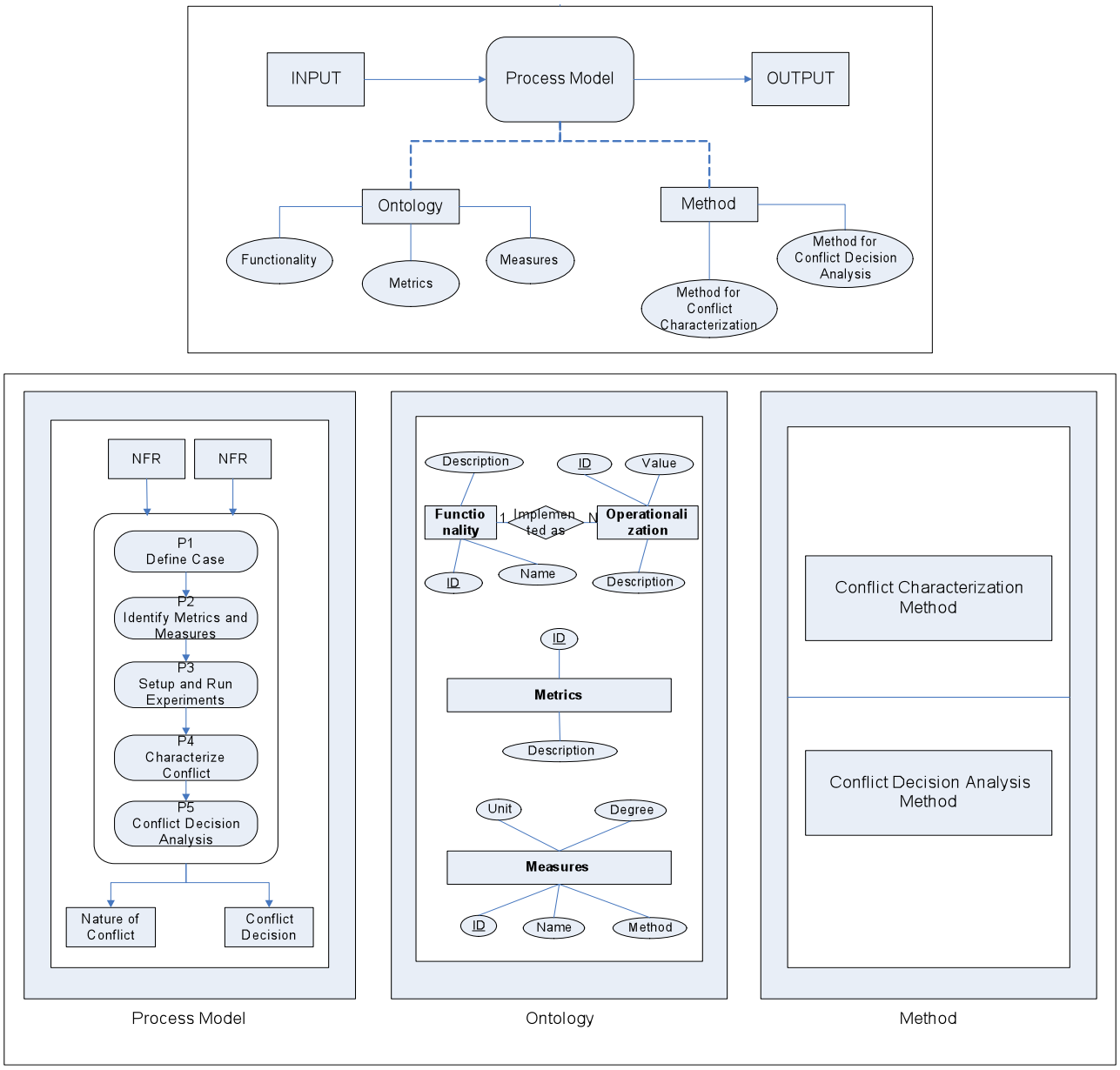
Inspired by this framework, in this paper we present the sureCM framework that is able: (1) to manage conflict among various types of NFRs; and (2) to provide quantitative reasoning about the NFRs conflict. We have adopted an experimental approach as the basis to attain the evidence to identify and characterize the conflict among NFRs. In this framework, NFRs are characterized as the associated system functionality and systems operationalizations, and NFRs metrics and measures are used as parameters to gather the quantitative evidence in the experiments. This empirical evidence will be used to perform conflict decision analysis. The sureCM Framework is depicted in Figure 2 and the main terminology used in the framework is presented in Table 1.

Functionality	:	a feature of a software system
Operationalization	:	a way of implementing functionality Other similar terms for operationalization are technology or design architecture
Metric	:	an attribute of a specified non-functional requirement
Measure	:	a quantitative measure of the degree to which a case, component or process possesses a given attribute.

Table 1 – sureCM Terminology

As shown in Figure 2, the sureCM Framework consists of one type of input: NFRs as written in software requirements documents; five-layer process: P1, P2, P3, P4, P5; and two types of output: nature of conflict and conflict decision. In the process model, the ontology and methods will be used as the basis to execute the five-layer process. The ontology consists of functionality; NFRs metrics and NFRs measures. The knowledge from this ontology will be used as the parameter to set up and run a series of experiments. Furthermore, two methods have been defined to manage the conflict: Method for Conflict Characterization and Method for Conflict Decision Analysis. Conflict characterization method will be used to characterize the conflict from the results of the experiments, while conflict decision analysis method will be used to identify the conflict decision based on the characterization defined. The relationship and interaction between these components is illustrated in Figure 3.

As shown in Figure 3, two types of NFRs (e.g. security requirement and usability requirement) will be used as the input to the framework. Then the conflict management process begins with the case definition, which is identifying the associate functionality of the system, i.e. relevant features of a software system, and the operationalizations, i.e. a way of implementing the defined functionality. Functionality part of the ontology will be used to assist with the identification process of operationalizations.



sureCM Framework

Figure 2 – sure CM Framework

By utilizing the ontology of NFRs metrics and NFRs measures, the NFRs that come as the input to the framework will be analyzed to identify their meanings and their associate metrics and measures in the context of the system being developed. These metrics and measures will be used in the experiments to quantify the NFRs level/degree of satisficing[1] in the system being developed.

As a result of P1: Define Case, and P2: Identify Metrics and Measures, four output will be generated: system functionality; associate operationalization; NFR1 metric and measure; NFR2 metric and measure. Those output that have been defined with respect to the context of the system being developed, will then be used as the input to the next process layer, P3: Setup and Run Experiments. The process P3, is dedicated to

[1] Satisfice is the term first coined by Hebert Simon [26]

designing the suitable experiments to collect the numerical data by utilizing the outputs attained from process P2. In this process, each applicable operationalization will be quantified using the defined NFRs metric and measure. The result of this experiment is the NFR's satisficing level/degree in the system. The results obtained from the experiments will then be analyzed by using Conflict Characterization Method. This method will be executed in the P4: Characterize Conflict process. In characterizing NFRs conflict, conflict characterization method begins by creating a two-dimensional conflict relationship graph based on the quantitative data obtained in P3. Each operationalization conducted in the experiments will be plotted based on its NFRs metrics scale. By plotting all of the defined operationalizations in P4, a conflict relationship characterization will be created.

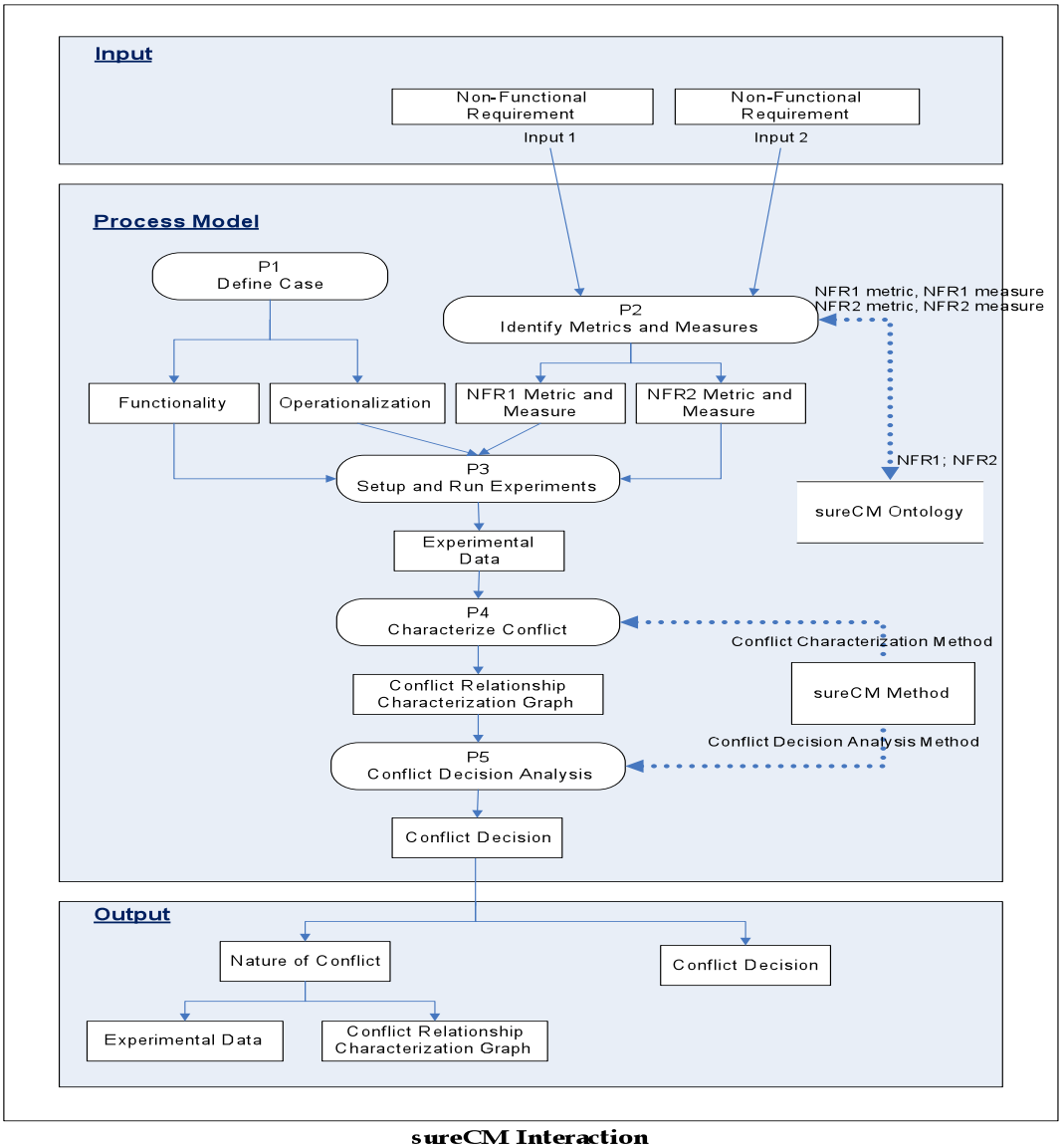


Figure 3 – sureCM Interaction

Lastly, by applying Conflict Decision Analysis Method, developers should be able to analyze the output from previous process, P4, to identify the suitable conflict decision to assist with the

development of strategy to deal with the conflict. This analysis will be conducted in the process P5: Conflict Decision Analysis process. The analysis conducted will then determine whether the identified conflict is strong, weak, or even if there is no conflict among these NFRs, and decision about this severity of conflict is based on the shape of the graph plotted in process P4.

We now present an example of the application of our approach in managing NFRs conflict.

IV. APPLYING THE APPROACH

Consider the following two NFRs given in a Software Requirements Specification document:

NFR 1: The Chemical Tracking System shall have identified/authenticated the user and protect user's personal information.

NFR 2: A chemist who has never used the system before shall be able to learn using the system easily and independently.

NFR1 and NFR2 will be used as the inputs and the associate functionality and their operationalizations are defined in Table 2.

Functionality	:	Getting Access
Operationalizations	:	Fixed Key Smart Card Scrambled Key Geometrical Pin Code Finger Print Palm Scanner Retina Scanner

Table 2 – Functionality and Operationalizations

NFR1 represents a security requirement and NFR2 a usability requirement. Previously defined ontology for security and usability metrics and measure are then utilized to identify the suitable metrics and measures for these requirements within the given context, as shown in Table 3.

By using the parameters generated from previous processes (as shown in Table 4), a series of experiments to measure the security and usability level of each operationalization were conducted. In the experiment, first step was identifying the potential instruments for each operationalization. For example, pincode used to get access in the Automated Teller Machine (ATM) or smartphone can be used as the instrument for Fixed Key operationalization, or scrambled pincode device can be used for Scrambled Key operationalization. By applying the measuring method of chosen NFRs metrics and measures; the quantitative security and usability level can be obtained. In this example, the length of pincode fixed-digit was considered as the representation of security measure frequency of review. The length of time (in seconds) needed by the user to learn to use the function correctly was used to measure the system usability.

Security

Security Metrics

Metric	Description
Privacy	Whether user's personal information is appropriately protected.
Data Completeness	How complete the data in the system.
Availability	Can user easily select parameter values for his/her convenient operation?
... etc	

Security Measures - Privacy

Measure	Unit	Method	Security Level
X = Number of policy violations, such as privacy and confidentiality incidents.	Count	1. Provide the list of "Privacy Incidents". 2. Count the number of privacy incidents in each experiment.	Confidentiality level = {(max – X) / max } * 100%
X = Frequency of reviews/audits	Count	Count the number of review/checking conducted by the system	Security level = X

Usability

Usability Metrics

Metric	Description
Task Efficiency	How efficient the system to the user.
Ease of Function Learning	How long does the user take to learn to use a function?
Customizability	Can user easily customize operation procedures for his/her convenience?
... etc	

Usability Measures – Ease of Function Learning

Measure	Unit	Method	Usability Level
X = T T = Mean time taken to learn to use a function correctly	Time	Count the length of time (in seconds) that the user needed to learn to use a function correctly. Counting starts when the user inserts the card into the machine and the screen appears, and counting stops when the user chooses one menu/option among various options appear on the screen.	Usability Level = {1/(1+X)}*100%

Table 3 – Metric and Measures Identification

Parameter:

Functionality	:	Getting Access
Operationalizations	:	Fixed Key Smart Card Scrambled Key Geometrical Pin Code Finger Print Palm Scanner Retina Scanner
Security Metric	:	Privacy
Security Measure	:	X = Frequency of reviews/audits
Usability Metric	:	Ease of Function Learning
Usability Measure	:	X = Mean time taken to learn to use a function correctly

Table 4 – Parameter for the Experiments

For each operationalization, the quantitative data obtained from the experiments is recorded and presented in Table 5.

Exp. ID	Ops.	Security	Usability	Note
GA.T.1	Fixed Key	4	0.0641	---
GA.T.2	Smart Card	1	0.1053	---
GA.T.3	Scrambled Key	7	0.0503	---
GA.T.4	Geometrical Pin Code	9	0.0500	---
GA.T.5	Finger Print	1	0.1563	---
GA.T.6	Palm Scanner	1	0.1370	---
GA.T.7	Retina Scanner	1	0.1266	---

Table 5 – sureCM Experimental Results

By using the experimental results in Table 5 and by applying Conflict Characterization Method of sureCM Framework, we then plot the conflict relationship between security and usability requirements as depicted in Figure 5.

Finally, by applying the Conflict Decision Analysis Method from sureCM Framework, practitioners should be able to analyze the nature of conflict to decide what levels of security and usability can be tolerated in the system. According to Figure 5, as the conflict relationship graph shows a non-linear function, there is an obvious conflict between security (privacy metric) and usability (ease of function learning metric). Also, as there is no clear optimum solution/s existing in the diagram, i.e. maximum security and maximum usability, this means that the existing conflict is a strong conflict, and the developer must choose the satisficing-solution for finding the right balance of attributes satisfaction. Conflict relationship graph can be a linear graph or a non-linear one. As long as there is a tradeoff in that graph, it means there is a conflict there.

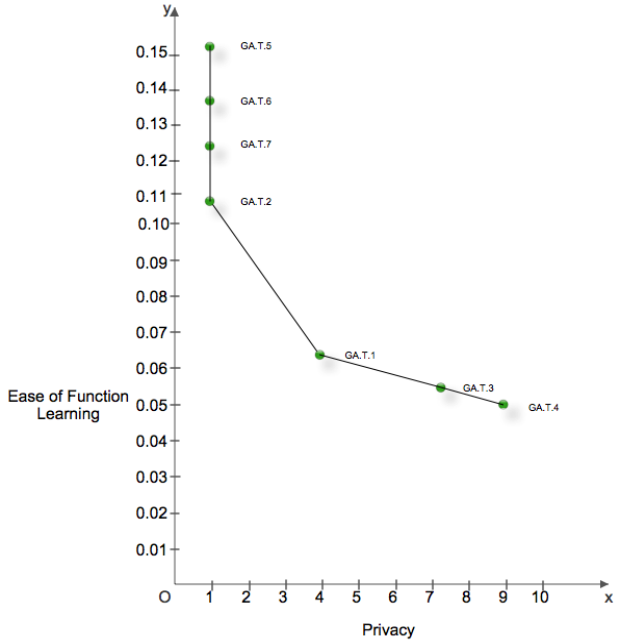


Figure 5 – Nature of Conflict

V. CONCLUSION

This paper describes a novel framework to identify and manage the relative conflict among NFRs. The framework utilizes an experimental-based approach as the foundation to characterize and analyze the conflict. NFRs metrics and measure are used as parameters to gather the quantitative evidence of NFRs conflict relationship. This evidence is used as the basis to characterize the conflict and to perform conflict decision analysis. An example that shows how the approach can be applied has also been presented.

This framework assumes a pair wise NFRs conflict characterization and decision analysis. However, it can also be easily extended to be applied in situation where conflict exists among more than two NFRs.

Although in this paper the conceptual model of the framework has been established; and a number of articles have also been published on investigating the notion of NFRs, the conflict among NFRs, the catalogue of conflict among NFRs with respect to NFRs relative characteristic and a preliminary framework on security and usability conflict [11, 21-24, 27, 28], however, several important tasks remain:

1) empirical evaluation

The framework will be empirically evaluated through controlled experiments and then followed by a series of industrial case studies. The reason for conducting controlled experiments is because: (a) "controlled experiments make it possible for the careful observation and precise manipulation of independent variables (e.g. proposed framework); (b) allowing for greater certainty; and (c) encourage the researcher to try out novel frameworks in a safe and exploratory environment before implementing them in the real world settings" [29]. Effectiveness and efficiency will be used as the evaluation criteria. Effectiveness means that this framework can be used to manage the NFRs conflict by considering NFRs relative characteristic, while efficiency represents how fast people can identify the conflict using the framework.

2) tool support

To support the framework utilization, we also plan to develop a semi-automatic tool that can assist software developers, particularly requirements engineers to perform conflict management among NFRs.

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, *et al.*, *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, *et al.*, *Modern software engineering, foundations and current perspectives*. New York, NY, USA: Van Nostrand Reinhold Co, 1989.

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

[7] N. Heumesser, *et al.*, "Essential and requisites for the management of evolution - requirements and incremental validation," Information Technology for European Advancement, ITEA-EMPRESS consortium2003.

[8] N. Yusop, *et al.*, "The impacts of non-functional requirements in web system projects," *International Journal of Value Chain Management* vol. 2, pp. 18-32, 2008.

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

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

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

[12] L. Chung, *et al.*, "Dealing with change: an approach using non-functional requirements," *Requirements Engineering,* vol. 1, pp. 238-260, 1996.

[13] B. Curtis, *et al.*, "A field study of the software design process for large systems," *Communication of the ACM,* vol. 31, pp. 1268-1287, 1988.

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

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

[16] W. N. Robinson*, et al.*, "Requirements interaction management," *ACM Computing Surveys,* vol. 35, pp. 132-190, 2003.

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

[18] H. In*, et al.*, "Aplying WinWin to quality requirements: a case study," in *23rd International Conference on Software Engineering*, Toronto, Ontario, Canada, 2001, pp. 555 - 564.

[19] 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.

[20] 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.

[21] D. Mairiza and D. Zowghi, "Constructing a catalogue of conflicts among non-functional requirements," in *Evaluation of novel approaches to software engineering*: Springer-Verlag, 2011.

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

[23] D. Mairiza*,* D. Zowghi, N. Nurmuliani, "Towards a catalogue of conflicts among non-functional requirements," presented at the 5th International Conference on Evaluation of Novel Approaches to Software Engineering (ENASE 2010), Athens, Greece, 2010.

[24] D. Mairiza and D. Zowghi, "An ontological framework to manage the relative conflicts between security and usability requirements," in *The Third International Workshop on Managing Requirements Knowledge (MaRK 2010), in conjunction with the 18th IEEE International Requirements Engineering Conference (RE'10)*, Sydney, Australia, 2010.

[25] R. Kishore*, et al.*, "A helix-spindle model for ontological engineering," *Communication of the ACM,* vol. 47, pp. 69-75, 2004.

[26] H. A. Simon, *The science of the artificial*, 3rd ed.: MIT Press, 1996.

[27] D. Mairiza, "Security usability conflict: a preliminary experiment," presented at the First CCF SIGRE Workshop 2011, Beijing, China, 2011.

[28] D. Mairiza, "Non-functional requirements in software development projects: a systematic review," presented at the ACS – BRASIG 29 September 2011, Sydney, Australia, 2011.

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