Towards Extension of UML 2.0 Meta-model

Uzma F. Alfatmi
Department of Computer Engineering, Vishwakarma Institute of Technology, Savitribai Phule Pune University Pune, Maharashtra, India

Prof. M.R. Dube
Department of Computer Engineering, Vishwakarma Institute of Technology, Savitribai Phule Pune University Pune, Maharashtra, India

Abstract - Functional requirements are always the only focused requirement in software development, and the Non Functional requirements (NFRs) are often hidden under Functional Requirements (FRs), they are difficult to achieve and expensive to deal with. So they are often neglected, overlooked, until they pose some difficulties or critical software failures. So, Non-Functional Requirements are required to be dealt with since the early stages of software development. But there are few research works, which have focused on them during the requirement analysis and design phase of the software development process. In this paper, we show a strategy to integrate elicited NFRs in use cases along with the functional requirements of the software. We propose the extension of UML, by enhancing the Use Case Profile, with the new modeling notations, to express the NFRs. These NFR notations will be used to depict the non-functional aspects of the software system, such as security, reliability, confidentiality, etc. The detailed description of these new Use Cases includes stereotypes and graphical notation. We have described the process to integrate NFRs into Use Case Diagram. Our proposed strategy will help developers to build more complete software specification and thus, to a shorter time-to-the-market. Therefore, expressing these NFRs in use cases will provide a solution, by facilitating the stakeholders, a NFR view of the software.

Keywords: Non-Functional Requirements, UML, Meta-model, Use Case, notations, Security.

I. INTRODUCTION

Modelling only Functional Requirements and overlooking other crucial requirements which are non-functional, such as security, reliability, portability, is one of the top most risk of Requirements Engineering. Although Non-Functional Requirements are present in many software development methods, they have not been considered, and are often left assuming them as second or third class of requirement. NFRs are mostly hidden inside some documents and therefore they are neglected or forgotten. Errors caused due to improper handling of NFRs are among the most expensive and difficult to correct. Therefore, NFRs must be considered seriously and should be dealt from the early stages of software development. NFR has received little attention by research, there are few works that focus on NFR as important requirements and they are not well understood. Not paying attention to these requirements has resulted in quite number of software failures in software development that include the deactivation of software just one day after its deployment. NFR play very dominant role in acceptability of software, but they have been neglected, by industry for long, until they realized the fact that NFR cannot be neglected further. The unsatisfied NFR, results into low acceptability of the product due to increasing competitive market, expectations of stakeholders and failures of various critical system.

UML Use Cases are helpful in the elicitation and documentation of functional requirements, when designing software systems, but they are not used for elicitation and documentation of NFRs. Use Cases are easy to understand, and therefore they are very well-suited for the communication and discussion of requirements with the stakeholders. It is very important to capture qualities and constraints of software, which are nothing but NFRs, in the initial phase of software development, that is the Use Case analysis, so that they will be focused throughout the development process. But there is no strategy to capture NFRs in Use Cases.

We have proposed an extension of UML 2.0 metamodel by defining new Use Case Notations for NFRs. All the new UCs defined for the NFRs are represented as a single package, called NFR-UC, for the identification and elicitation of non-functional requirements of a software system. This paper also describes the process of applying this UML extension, for capturing the NFRs in Use Case diagrams, with the help of the proposed NFR UCs and relationships defined in this profile. The rest of the paper is organized, as follows: In Section 2, Literature Survey is presented, in

which the previous methods to handle NFR at various stages of development is given. Section 3, describes the Proposed Work, section 4, states the conclusion of the proposed work and in section 5 Future Work is stated.

II. LITERATURE REVIEW

Many researchers proposed extensions of UML model for the integration of NFRs in the analysis and design phase. Lawrence Chung et al. [1] proposed to integrate the NFRs with FRs in UML use case model. They implemented the NFRs by a NFR framework and associated those NFRs with four use case model elements: use case, actor, actor-use case association and system boundary. They named these associations as "Use Case Association Point", "Actor Association Point", "Actor-Use Case Association (AU-A) Association Point" and "System Boundary Association Point", respectively. Chung's work has represented NFRs and their conflicts, but it does not focus on conflict detection with functional requirements.

Ana Moreira et al. [2] proposed a model for integrating quality attributes with FRs in use case diagram and interaction diagram. They proposed a template for quality attributes with some specific fields, such as description, source, focus, decomposition and they integrated those quality attributes with FRs by using standard UML notations with extended special notations.

Luiz Marcio Cysneiros et al. [3] proposed a systematic approach, through which the conceptual models will reflect the elicited NFRs. They used a vocabulary anchor called Language Extended Lexicon (LEL) to build functional and nonfunctional aspects of a software system. They have also described, how to integrate NFRs into UML by extending some of the UML sublanguages, and also presented a systematic way for integration of NFRs into the functional models.

Evgeni Dimitrov et al. [4] described three approaches for UML-based performance engineering. The three approaches are: 1) Direct representation of performance aspects using UML, 2) Expanding UML to deal with performance aspects and, 3) Combining UML with formal description techniques. They proposed some extensions to UML use case and state transition diagram.

Luiz Marcio et al. [5] provides an extension of previous work [6] and presents a strategy for dealing with NFRs and a process to integrate them into conceptual models. Some part of their work has been strongly influenced by Chung's work [1], resulting in a slight adaptation of his NFR graph to be used in earlier stages of software development. The authors consider that the systematic integration of NFRs into conceptual models will be helpful in identifying such conflicts, which were not focused by Chung et al [1]. Their integration strategy is a lightweight method which is simple and does not require a large investment to be put into practice. They have shown that dealing with NFRs from the very beginning of software development and integrating them with the functional conceptual models leads to cost savings and higher customer satisfaction.

Supakkul et al. [7] proposed an NFR Framework to address NFRs. It is a a goal-oriented approach in which, authors represented NFRs as softgoals to be satisficed. NFR softgoals are satisficed when there is sufficient positive and little negative evidence, otherwise they are unsatisficed. NFR softgoals are identified by nomenclature "Type[Topic]", where Type denotes a non-functional aspect and Topic represents the context for the Type. The criticality of NFR softgoals is either neutral, critical or very critical. Softgoals may be refined into offspring softgoals with more specific Type or Topic using AND- or OR-decompositions. During Topic refinement, the Topic of parent softgoal is inherited by offspring softgoals. To determine satisficeability, they identified one or more operationalizing softgoals, and their corresponding degree of contribution indicating how well they achieve NFR softgoals (MAKE (++), HELP (+), HURT (-), or BREAK (--)). They then labeled the desirable leaf-node solutions with √ and labeled the undesirable ones with ×. They manually or via a CASE tool, propagated these leaf-node labels upstream the goal hierarchy. All of these modeling activities are recorded in a diagram call Softgoal Interdependency Graph (SIG).

III. PROPOSED WORK

UML 2.0 provides an extensibility mechanism to extend the modeling language using well-defined extensibility constructs that are packaged in a UML Profile. In our work, we use stereotypes to define new types of model elements into UML metamodel. To define NFR Use Cases it is necessary to extend the UML 2.0 metamodel and define stereotypes. A stereotype is an extension of the UML vocabulary that allows us to create new modeling elements, derived from the existing ones. In this section we present the UML Use Case profile extension. It will be possible to represent the Non Functional aspects for UC diagram, such as security, reliability and portability, thus obtaining more complete Use Case diagrams that will reflect both, the Functional and Non- functional requirements. This extension has been built, as meta-class of NFR use cases.

In Figure 1, we briefly represent the NFR-UCs. There are two elements in the definition: 1) Name of NFR: This indicates the purpose and significance of the element. 2) Notation: This is a graphical icon associated with the NFR. Figure 2, depicts the advanced relationships that will be used to represent relationships between FR- NFRs, and NFR-NFR.

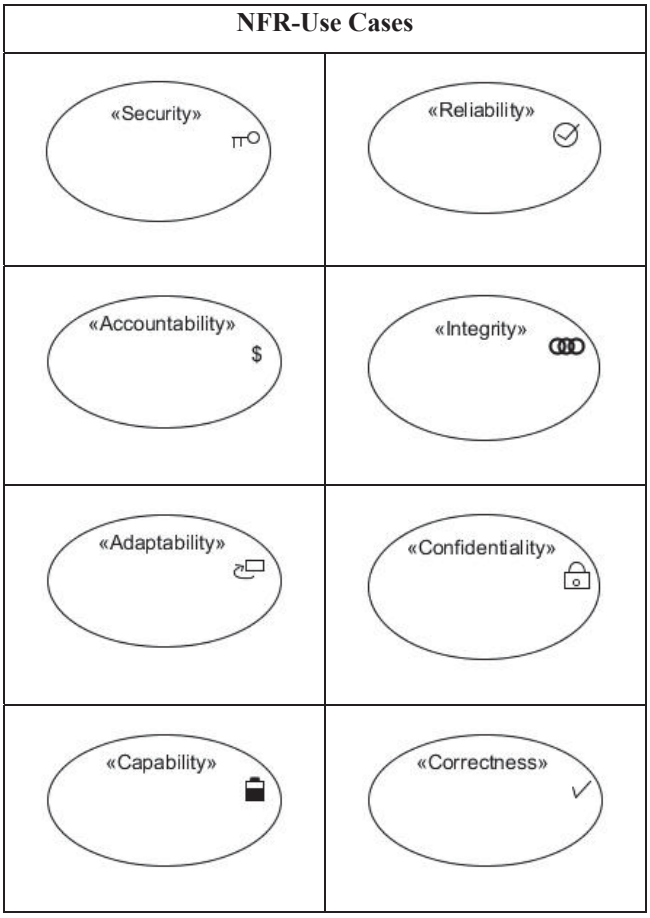

Fig 1: NFR-Use Cases

NFRs Relationships	
«threatens»	«mitigates»
«presumes»	«aggravates»
«exploits»	«prevents»
«detects»	«harms»
«negates»	«constraints»

Fig 2: Advanced Relationships

3.1 PROCESS FRAMEWORK

We proposed a process framework that will identify and integrate the FRs and NFRs, in use case analysis, as shown in Figure 3 below.

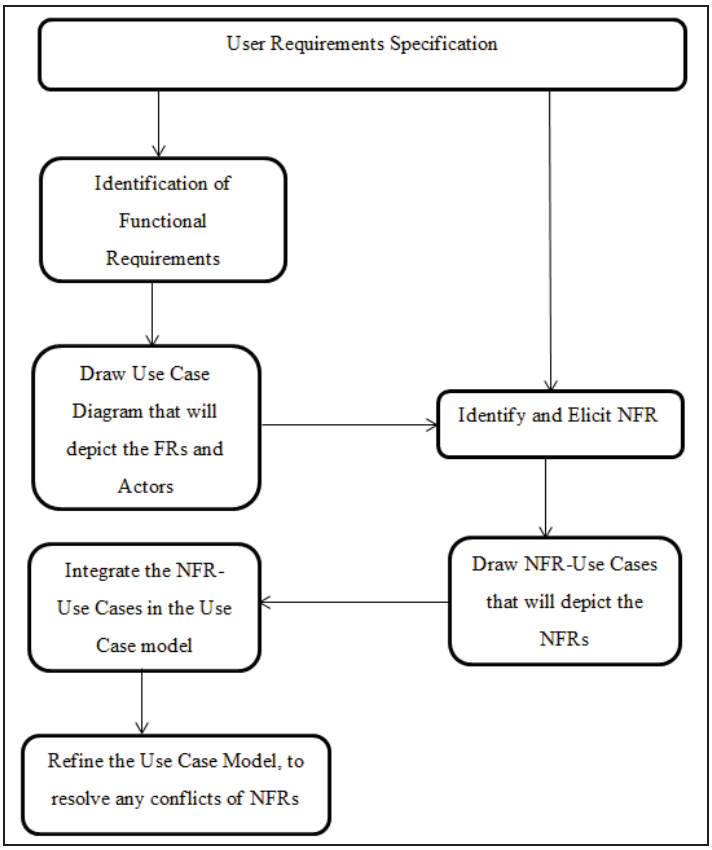

Fig 3: NFR Integration Process

The process is described below:

1. The proposed framework starts with the extraction of Functional Requirements from the user requirements specification of a software system.
2. Identify and specify the Actors of the system.
3. Transform the Functional Requirements to Use Cases in Use Case Diagram.
4. Identify and Extract the Non Functional Requirements from the User requirements of the software system, and from the FRs.
5. Represent the elicited NFRs, by using the appropriate proposed NFR Use Case Notations, and augment it in the same Use Case Diagram. Thus the FRs and NFRs will be depicted in a single integrated Use Case Diagram.
6. Represent the relationship between a FR Use case and NFR Use case, by using the appropriate proposed advanced relationships, such as prevents, aggravates, threatens, conflicts, etc.
7. Identify and resolve conflicts of NFR use cases.

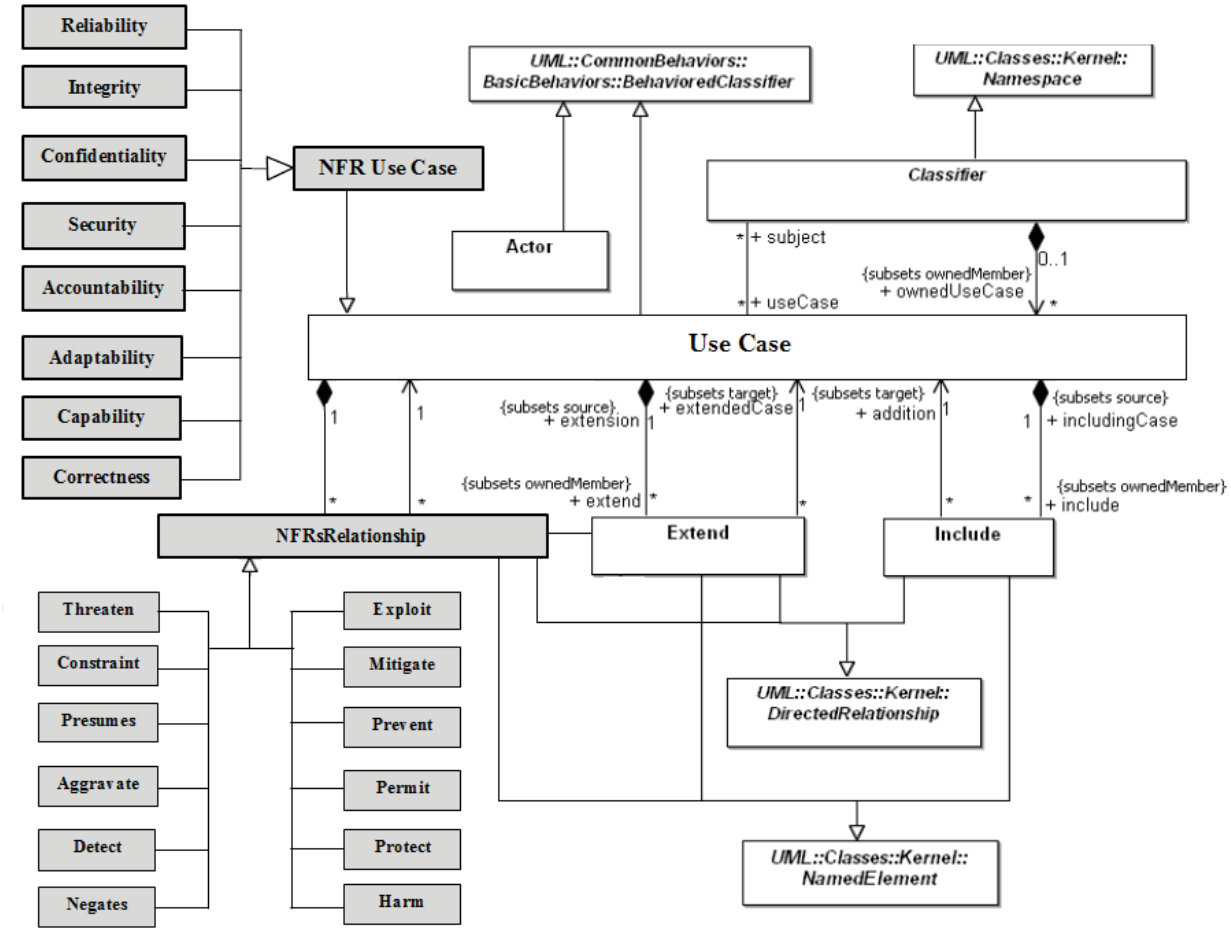

Fig 4: The concept used for modeling Use Cases and NFR-Use Cases

IV. CONCLUSION

This paper presents a use-case driven approach to capture NFRs of the system that extends the use case model to include non-functional requirements. The UML 2.0 meta-model is extended as shown in Figure 4, with new modeling elements for NFRs. The benefits of this approach, is that, stakeholders will have an integrated view of the system that will not only reflect the functional requirements but also the important non-functional aspects of the system, We, have also proposed new relationships, to be included in use case modeling, to connect the FRs and NFRs. Thus a more complete use case analysis is achieved using this approach.

V. FUTURE SCOPE

One of the problems, with the proposed method, is that some NFR may conflict with each other, such as security and efficiency by negative influences. These conflicts can be identified and resolved by assigning priorities to NFR, such as Low, Medium and High. The method to handle such drawback and prioritization of the requirements can be included in the future work.

ACKNOWLEDGEMENT

I would like to express gratitude to all who have helped me with their valuable guidance for this work. I would like to thank my guide Prof. M.R. Dube, Vishwakarma Institute of Technology, Pune, for guiding me, in ways and means by which I was able to complete this paper. I also thank the researchers for publishing their research work as a guideline.

REFERENCES

[1] S. Supakkul and L. Chung, "Integrating FRs and NFRs: A Use Case and Goal Driven Approach".
[2] A. Moreira, I. Brito, and J. Arajo, “Crosscutting quality attributes for requirements engineering”, in The fourteenth International Conference on Software Engineering and Knowledge Engineering (SEKE’02), pages 167–174, July 2002.
[3] Luiz Marcio Cysneiros, Julio Cesar Sampaio do Prado Leite and Jaime de Melo Sabat Neto, "A Framework for Integrating Non-Functional Requirements into Conceptual Models," in Springer, 2001.
[4] E. Dimitrov and A. Schmietendorf, “Uml-based performance engineering possibilities and techniques”, IEEE Software, 19:74–83, January/February 2002.
[5] Cysneiros, L.M., Sampaio do Prado Leite, J.C, "Nonfunctional requirements: from elicitation to conceptual models," Software Engineering, IEEE Transactions, vol. 30, no. 5, pp. 328-350, 2004.
[6] Cysneiros LM, Leite JCSP, “Integrating non-functional requirements into data model”, in Proceedings of the 4th international symposium on requirements engineering – Ireland, IEEE Computer Society Press, Los Alamitos, 1999, pp 162–171
[7] S. Supakkul and L. Chung, "A UML Profile for Goal-Oriented and Use Case-Driven Representation of NFRs," in Third ACIS International Conference on SERA'05, IEEE, 2005.