Risk-driven Non-functional Requirement Analysis and Specification

Yi Liu[1,2], Zhiyi Ma[1,2], Hui Liu[1,3], Weizhong Shao[1,2]

[1] Key Laboratory of High Confidence Software Technologies, Ministry of Education,

[2] School of Electronics Engineering and Computer Science, Peking University, Beijing, 100871, China

[3] Beijing Lab of IntelligentInformation Technology, School of Computer Science and Technology, Beijing Institute of Technology, Beijing 100081, China

{liuyi07, mzy} @sei.pku.edu.cn, liuhui08@bit.edu.cn, wzshao@pku.edu.cn

Abstract—**The complexity and usefulness of software systems are determined not only by their functionality, but also by non-functional requirements such as accuracy, security, cost, user-friendliness and performance. However, even with the growing interest in dealing with NFRs from early stages of software development, current technology is not adequate for analyzing and representationally expressing these requirements. They mainly focus on refinement of NFRs themselves, neglecting the analysis of factors or risks in the environment that may have adverse impacts on these NFRs, which may lead to incomplete understanding and prevent the effective processing of NFRs. In this paper, we propose a risk driven approach to analyzing and specifying NFRs, in which NFRs are considered to address business risks. We first elaborate the aspects that should be considered when understanding NFRs. We also propose constrained case diagram to model the analysis result and their impacts on target system. This approach provides us an operating procedure and a complete NFR view. It could complement with existing approaches to more completely analyze and refine NFRs.**

Keywords-non-functional requirements; risk driven; analysis; specification; software development

I. INTRODUCTION

The complexity and usefulness of software systems are determined not only by their functionality, but also by Non-Functional Requirements (NFRs) such as accuracy, security, reliability, user-friendliness, performance [1][2]. Due t o the importance of NFRs, researches have been done on how to achieve them in software systems. Generally, two main approaches were defined: curative approach and preventive approach. A cur ative approach is applied after a software system has been developed; the software system is tested and defects associated with NFRs are identified in order to be fixed. In a preventive approach, the software development tasks (e.g. constructing architectures and design models) are planned to prevent defects associated with NFRs, incorporate NFRs into the design and i mplementation, and finally obtain a sys tem which satisfies the NFRs [2].

In both approaches, especially the preventive approach, a concise specification of NFRs is an important premise. Traditionally, NFRs are just simply and textually denoted in a distinctly section of the requirement specifications [3], e.g., "The system shall be available 24×7, give users a one-second response time, and process 200 orders per minute". These specifications are remote from the job of the software and it hard to te ll that how developers are supposed to r eact to them and take wh ich steps to achieve them. Later, goal-oriented approaches [4][5] have been proposed to fu rther analyze and operationalized NFRs, in which NFR s are taken as s oftgoals. Scenario-based approaches [6][7] are also proposed to describe NFRs using scenarios.

However, these approaches mainly focus on refinement of NFRs themselves in the scope of software, neglecting the analysis of the factors/risks in the environment that would have adverse impacts on them, which may lead to incomplete understanding and in appropriate solutions of NFRs. Additionally, corresponding models are just represent NFRs and their relationships with functionalities, the d erived requirements which will be directly designed and implemented to support the NFRs are always ignored or denoted implicitly. These incomplete models may prevent effective handling of NFRs.

In this paper, we propose a risk-driven approach to stepwise analyze and r efine NFRs. Besides breaking down a spec ific kind of broad non-functional goal into clear requirements that are as detailed as developer need, we als o identify origins of NFRs; analyze the external and in ternal challenges for achieving them; and reco mmend tactics based on the p revious analysis. A constrained case diagram is also proposed to model the analysis results, so as to e ffectively deal with them in the following software development lifecycle.

Main contributions of our approach include: (1) Support the capture of NFRs based on risks, identify the threats and challenges, which allows the business and developers to understand the origins of NFR and d etermine the appropriate solutions; (2) Provide an op erational guideline to t horoughly analyze NFRs, follow which we could ob tain a more complete understanding NFR origins, challenges need to be tack led and other context; (3) Allow the developers to fully capture the requirements by providing an integrated graphical view about NFRs, the derived requirements, the functional requirements affected by them and relationships among them. (4) Moreover, this approach could complement with existing analysis approaches to better deal with NFRs.

The rest of the paper is organized as follows. Section II introduces related works and our bas ic ideas. Secti on III elaborates the risk driven NFR analysis approach and the proposed constrained cased diagram. Section IV introduces a case study. Section V discusses the pr oposed approach and concludes the paper.

II. Background and Related Works

A. Non-Functional Requirements Analysis Approaches

To effectively deal with NFRs in software development, specifying them in detail and accurately is an important premise. Since initial d escriptions of NFRs are often roughly and non-behavioral, they are hard to tell be rea lized like functional requirements. Therefore, a further analysis to br eak down a sp ecific kind of broad non-functional goal into clear requirements that are as d etailed as developer need, and accurately record and manage these requirements throughout the life cycle of systems development is needed.

Goal-oriented methods, such as th e NFR Framework proposed by Chung et a l [4], the i*/Tropos approach [5], are typical representatives of these works. NFRs are taken as softgoals, gradually refined to lower, more-precise level of detail either by the type or functional topic, and finally refined to operationalizations. However, these approaches mainly focus on the refinement of NFR themselves in t he scope of target software, lack of adequate analysis of the environment entities. Scenario based approaches are also proposed to deal with NFRs [6][7][8]. However, most of them focused on the representation of NFRs not the analysis process. Moreover, nearly all the approa ches are based on the premise that NFRs are already identified, lack of the consi derations about the source and origin of NFRs, which makes it hard to d etermine their priorities and may lead to improper solution recommendations. Thus, besides refining NFRs themselves, a more thorough analysis about the motivations of NFR and the factors in the environment that would have adverse impacts on NFRs are also neede d, in order to effectively dealing with NFRs in software development lifecycle.

In security engineering [9], risk assessment is an important task. It is defined as a process of identifying the risks to system security and determining the probability of occurrence, the resulting impact and a dditional safeguards that would mitigate this impact [10]. This process conducts analysis about security requirement from various aspects, and provides solutions to achieve them. It i s also used i n Software Performance Engineering (SPE) to help analysis of performance [11]. Through analysis in practice and surveying existing literatures, we found the idea of risk assessment could be also applied for analyzing other NFRs such as avail ability, reliability. NFRs could be cons idered as a set of quality statements that th e system must meet in order to manage the risks exposed to system stakeholders, where these business risks may originate from low quality of services under different operating conditions [12]. For ex ample, the business may be at risk of: customer retention if the system is running under poor performance or response times; loss of revenue if the system does not provide sufficient availability of its services, and to mitigate these risks, corresponding performance requirement, availability requirement should put on the ta ble and find solutions to deal with them. Based on this idea, we could more fundamentally know much about the NFRs, thus, conducting more thorough analysis of them.

In this paper, we prop ose a risk driven appr oach to analyzing and refining NFRs. It draws lessons from the idea of risk assessment and management [9][10], and will co mplement with existing approaches to more adequately analyze NFRs.

B. Specifying and Modeling NFRs

Traditionally, during the requirement phase, NFRs are collected and documented in some textual format. They are specified in a separate section of the documentation, as an entry of the functionality description, or recorded in an artifact called supplementary specification [3]. W hen use case models become popular to model functional requirements, approaches have been proposed to e xtend them to address NFRs. Armour and Miller [13] discuss how one may document the 'non-behavioral requirements' using a use case, immediately after the Alternative Flows and Exception Section. Supakkul and Chung [8] represent NFRs as softgoals and associate them with appropriate use case model elements. Abuse case and Misus e case is proposed which extends the use case to define a security requirement [6].

However, these approaches mainly focus on NFRs themselves and th eir constraint relationships with functionalities. The der ived requirements identified to achieve or support NFRs ar e often neglected or just implicitly represented in t he models. Since NFRs are non-behavioral in nature, corresponding tactics for achieving them should be identified first in order to make them operational. Thus, just graphically representing FRs and NFRs together is not enough and relatively incomplete. In this paper, based on the proposed risk driven NFR analysis approach, we propose a constrained case diagram to specify NFRs. Besides associating NFRs with related use case, it also represents the derived requirements for achieving the NFRs, and attaches them to affected functional requirements. This di agram could be considered as a complement to the use case, a nd provide an integrated view for NFRs.

III. The Proposed Appraoch

The proposed approach in t his paper for analyzing and refining NFR is risk-driven which we got inspiration from risk assessment and management used in security engineering and the SPE process [9][11]. NFRs are identified based on possible business risks for potential low quality of service of the target system. External threats and internal vulnerabilities are then analyzed to find the possible reasons for low quality of service. Furthermore, tactics for achieving NFRs are recommended based on the risks, threats and vulnerabilities. In the following, we first introduce the aspects that needed to be considered when analyzing and refining NFRs. Then, based o n these aspects, the risk driven analysis process is elaborated. Lastly, we present the constrained case diagram which used to model the analysis results.

A. Understanding Non-Functional Requirements

For best achie vement of an NFR, a concise understanding of NFR itself and its environment is very important. Basically, we need to find out answers for the following questions in the analysis process [14]: (1) what exactly the NFR is about; (2) why need the NFR; and (3) how to achieve it. The first question could roughly correspond to a basic description of the NFR, including its category, related business goal (functionality) and the target value. The answer of the second question is risk related information since NFRs are proposed to address business risks. To achieve NFRs, we should identify the factors that may have adverse impact on them, and propose solutions according to the factors. Table I shows the main aspects that should be considered in NFR analysis. It could also be considered as a template to specify NFRs.

TABLE I. ASPECTS FOR UNDERSTANDING NFR

	Aspect	Description
what	Name	Name of the NFR
	NFR Category	Category of the non-functional requirement. E.g., performance, security, availability, reliability.
	Related business goals	Which part of the system does this NFR target apply to? An NFR can apply to a single function, a group of functions, a single interface, and so on, or it can apply to everything (all functions).
	Target (NFP value)	The quality target that the system need to deliver. It might be some quantitative or qualitative measure. E.g., System should operate 24 *7
	Exceptional cases	Is the quality target likely to be unachievable in some cases? E.g., functions that involve intensive processing will be slower, so it' s unfair to judge them against the same response time goal.
why	Motivation (risks to be addressed)	Specify the potential risk and analyze its impact on system's business value. It is the reason for proposing the NFR. . E.g., the motivation for a fast download of software by a Web visitor might be so that they don't give up impatiently halfway through.
	Criticality /Priority	Description of impact if the NFR is not achieved, and specify the impact grade,such as:high, medium or low.
How	Challenges, Threats or Vulnerabilities	Any circumstance or event that can have negative impacts on end-user experience of the quality, or a list of all those pieces relevant to your environment that have a bearing on the non-functional goal of your system.
	Impact rating/ Likelihood / Risk rating	Level of the impact resulting from successful exploitation of a vulnerability; likelihood of the probability that a potential vulnerability might be exploited in the context of the associated threat environment/ the level of risks to the system.
	NFR Tactics (risk mitigation recommendations)	Things to be done to help achieve this non-functional goal. They could be any specific features we want our system to have to help to achieve the goal. They are considered as a set of new derived requirements. It will be better if each of these requirements includes a statement estimating the extent to which it contributes.
	Extra requirements	Explains what sorts of extra requirements should be considered and in what circumstances. It provides guidance on what to do beyond simply specifying the main requirement.E.g.: E.g., Functions for measuring response times and letting administrators see them.
	Acceptance criteria	Could be taken as the criteria for verifying that the NFR has been met (may be used to assist definition of system test cases).

Among above aspects, the knowledge about threat, vulnerabilities, NFR tactics and relationships among them is especially important for finally implementing NFRs in the target system. The threat and vulner abilities explains the possible reasons why the system may provide poor quality of service, and the tactics provide countermeasures for the threats and vulnerabilities, so as to achieve NFRs. Fortunately, these types of knowledge could be accumulated based on previous experience, such as the development experience of similar systems, journal articles and conference papers, technical reports etc.

B. *Risk-driven NFR Analysis Process*

Formulating a detailed and accurate analysis of NFRs is not a straightforward task, because various aspects have to be considered. In this subsection, we present a process that guides the analyst to analyze and refine NFRs. It starts with characterizing the system and identifying the initial NFRs by recognizing the associated risk to the system stakeholders. Then, threats and vulnerabilities that may do harm to the NFR, and possible tactics to mitigate the risks are identified so as to achieve NFRs in the target system. This stage is accomplished through following steps, which need the participation of both the stakeholders and developers.

1) *Preparation:characterizing the target software system*

In this step, the analyst defined the boundaries of the IT system, along with the resources that constitute the system, its connectivity, and any other elements necessary to describe the system. The aim is to provide a context for NFR analysis.

2) *Risk analysis and NFR identification*

Analyze potential business risks, assess their impacts; and neatly specify the concerned non-functional properties for mitigating the risks, including the category, a brief summary on the target value and the reasons why stakeholders concern about them. This step could be assisted by an existing risk assessment process.

3) *Related functionality identification*

Based on system characterization, identify which functionality is related to the non-functional requirements or what does this NFR apply to. Is it for a specific function, a collection of functions, all functions for a class of users, or something else?

4) *Threats identification*

Towards the identified NFR and system characterization, draw up a list of the external threats or risks relevant to the environment that have a bearing on the non-functional properties of the system. Record and specify their impacts to corresponding non-functional properties. This knowledge could be collected from trusted and proven journal articles, conference papers, books, software companies' brochures and some websites [9][15].

5) *Vulnerability idenification*

In this step, the analysts developed a list of internal system vulnerabilities (flaws or weaknesses) that could be exploited by the potential threat vectors or that would have negative impacts on the achievement of NFRs. It often needs a deeper analysis of system characteristics.

6) *Risk determination*

In this step, the analysts determined the degree of risk to the system, including the likelihood determination, impact rating and the risk level. Existing assessment methods [9][10] could be directly used in this step.

7) Exceptional cases identification

Is the non-functional requirement likely to be unachievable in some cases? For example, functions that involve intensive processing will be sl ower, so it's unfair to judge them against the same goal.

8) Tactic recommendations

Based on the identified threats and risks, identify the tactics that could be used to improve any of these threats or risks. It would be better if each tactic include a statement estimating the extent to which it contributes. Since tactics for N FRs are relatively fixed, knowledge about the tactics could be collected in advance from previous experiences or d ependable documents [6][15].

9) Extra requirement identification

Besides the tactics recommended mitigating the threats and vulnerabilities, some additional requirements may be introduced. For example, functions for measuring response times and letting administrators see them. In this step, we identify and explain what sorts of extra requirements should be considered and in what circumstances.

10) Result documentation

Presented to the stakeholders to check an d confirm the derived requirements and to t he developers for the following design and implementation. We could use the template derived from table II and constrained case diagrams defined i n the following subsection.

Noticing that this is a n iterative and incremental process, and not all steps must be performed in a NFR analysis process. Analyzers and user s could customize above steps and form more appropriate process.

C. Constrained Case Diagram for NFR Modeling

To provide a precise context to NFR design and implementation, we propose a constrained case diagram to graphical model the NFRs, the derived requirements, the original functional requirements and relationships among them. This diagram could be roughly considered as a complement to the use case diagram since we refer to the use cases to specify the functionality affected by the NFRs.

Figure.1 shows the metamodel of the constrained case diagram. The gray elements are borrowed from UML specification [14], and the white elements are whi ch we proposed to model NFR related elements. This diagram mainly consists of four types of classifiers and four types of relationships. The classifiers include: (1) the *Constrained Case* which is us ed to represent the refined non-functional requirement; (2) the constrained artifact that constrained by the NFRs which refers to specific *Use Case*; (3) the *NFR Tactic* which recommended to deal with the threats and vulnerabilities so as t o achieve NFR; (4) the *Extra Requirement* which identified to support the NFR. The relationships include: (1) the relationship *Constrain* which connect NFR and r elated functionality together; (2) the relationship *Satisfy* which connect the *NFR tactic* to the NFRs ; (3) the relationship *Support* which connect the *Extra Requirement* to the NFR; (4) the relationship *ActUpon* which specifies that a d erived requirement will act upon the constrained artifact.

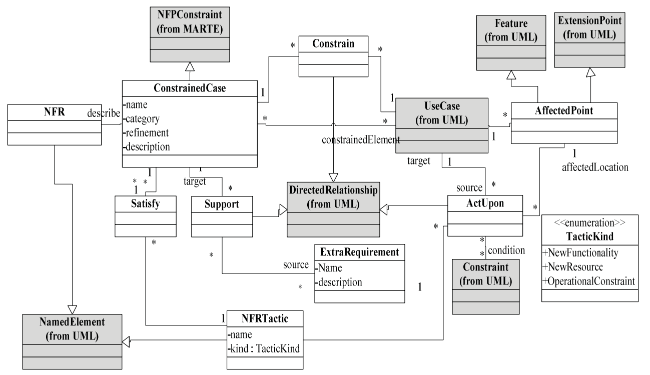


Figure 1. Metamodel of the Constrained Case diagram

For *ConstrainedCase*, the attribute *refinement* denotes the concrete NFRs refined from the high-level NFR. The attribute *constrainedElement* refers to the artifact that constrained by the NFR. In our metamodel, it mainly refers to use cases. For example, associating fast response time NFR to Withdraw Fund use case of an Automated Teller Machine (ATM) system. Moreover, to specify relationships between the NFRs and the constrained functionality, we introduce a r elationship *Constraint* in this metamodel.

The element *NFRTactic* in this metamodel represents the countermeasure adopted to make the NFR satisfied. It might be a new functionality, new resource, operational constraint, etc. It will be realized into some software entities and interact with the constrained functionality. The *Extra Requirement* represents the requirement derived to support certain NFR but not to achieve them, such as monitoring related non-functional property and letting administrators see them. The r elationship *ActUpon* is proposed to specify the relationship between a NFR derived requirement and the affected use case. Moreover, since sometimes use case i s further described by interactions, activities or state machines, we could specify the sp ecific location that the NFR tactics will act upon. Using the element *affectedPoint*. For example, for a use case described by an activity diagram, the affected point may be one of the actions.

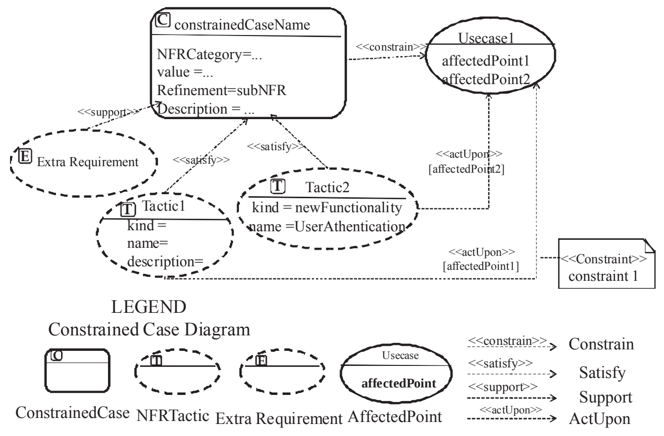


Figure 2. Graphical notation of Constrained Case diagram

Figure 2 s hows the grap hical notation of the constrained case diagram. Using this diagram, the NFRs, the derived requirements (NFR tactics and extra requirements), the affected use cases, and relationships among them could be modeled explicitly in an integrated view, which provides the designers a precise context for NFR r ealizations and a source of the transformation from NFRs into architecture.

The constrained case diagram could be considered as a complement to the functionality-focused use case diagram, highlighting in each view that designer needs to consider both the functional and non-functional requirements to present a complete picture for the following software development stages. For example, it could be leveraged in the 4+1 view [15], together with the use case view to assist the software architecture design; or it could be lever aged by the tester to validate the requirements. Additionally, it can also be used in planning project activities, resources, cost and timeframe.

IV. CASE STUDY

In this section, we p resent a case study o n a simplified Online Shopping System.

A. *System Characterization and Identfiied NFRs*

Basic functionalities of an online shopping system include account management which assists customer to maintain their account information and; allowing customers to bro wse the product; adding, removing items to shopping carts and to reviewing their orders; especially, to at tract customer, the system is required to provide a flash sale/spike area function, where in a certain period of time, a pr oduct is sold in an incredible price. This may result in a big visit amount on the website. Figure 3 shows the functionality-focused use case diagram of the system.

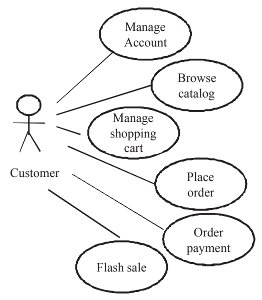


Figure 3. The usecase diagram

By analyzing potential risks from users' perspective, three types of i mportant NFRs are identified, one is the *Security* for account, the other is the *Availability* of the sho pping, and the last one is the *Performance* for flash sale. Table II(a),II(b),II(c) shows the initially identified NFRs using our template.

TABLE II. INITIALLY IDENTIFIED NFRS DESCRIPTION

Constrained Case: Security constraint
Description: the system shall have high security.
Motivation: Account information is very important for online shopp ing system, poor security may injure customers' benefits and affect the customer retention.
NFR Category: Security

(a) Initially identified Security constraint

Constrained Case: Availability constraint
Description: the system shall be available to users 24 hours every day, every day of the year, especially the functionality Place Order.
Motivation: most online shopping systems are available 7*24, if the target system does not provide similar availability, it might lead the ri sk of losing many customers for the business.
NFR Category: Availability

(b) Initially identified Availability constraint

Constrained Case: Performance constraint
Description: the various f unctionalities shall have an appropriate and acceptable response time.
Motivation: Under multiple concurrent user load, the system's response time may become unacceptable to the en d users which lead to the ri sk of losing customers.
NFR Category: Performance (Response time)

(c) Initially identified Performance constraint

B. *Risk Driven Analysis and Refinement*

Towards these high-level NFRs, the next to do is r efining them, analyzing threats and vulnerabilities, and deriving the tactics and extra requirements. Table III s hows the an alysis results guided by our risk driven analysis approach.

TABLE III. REFINED NFRS DESCIPTION

Constrained Case: Security constraint		
Related Business Goal: Manage Account, Order payment		
Challenges or threats: unauthorized access		
Impact Rating: High	Likelihood: Medium	Risk Rating: Medium
Recommended tactics: Data Encryption, Password Authentication, select the third-party payment platform		

(a) Refined Security constraint

Constrained Case: Availability constraint		
Related Business Goal: Place Order, Order payment		
Challenges or vulnerabilities: Unreliable network links, hardware failure, service interruptions of the pay platform		
Impact Rating: High	Likelihood: Medium	Risk Rating: Medium
Recommended tactics: Ping/echo, Replicate Server, Multiple payment mechanism		
Extra Requirement: System unavailable page(when the system is unavailable to users, any attempt by a user to access the system shall result in the display of page informing them that it is unavailable),		

(b) Refined Availability constraint

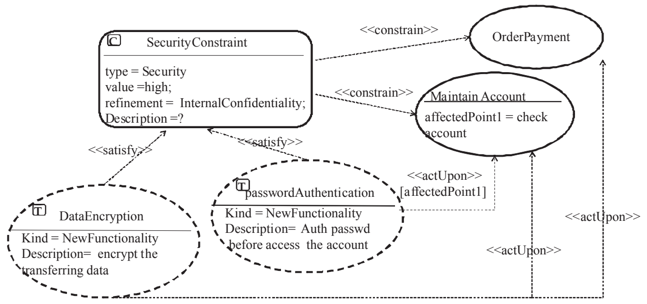
Constrained Case: Performance constraint		
Related Business Goal: Place Order, Flash sale		
Challenges or vulnerabilities: Large amount of concurrent users,		
Impact Rating: High	Likelihood: Medium	Risk Rating: Medium
Exceptional case: at the ti me of flash sale, system load are inevitable high, allow more response time for flash sale		
Recommended tactics: Fixed Scheduling Policy, Cache mechanism, Temporarily disable functions that cause intensive processing		
Extra Requirement: Response time monitoring (Functions for measuring response times and letting administrators see them)		

(c) Refined Performance constraint

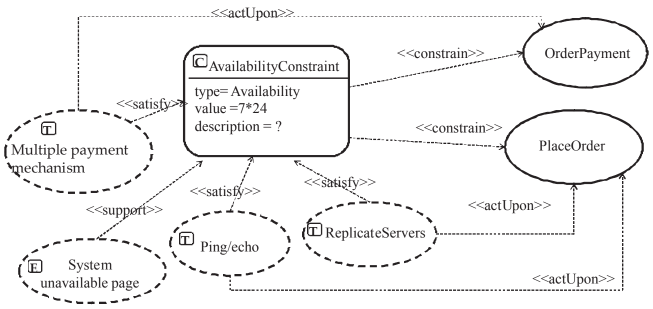
C. *Model NFRs in Constrained Case Diagrams*

To provide a clear graphical input to the following software development stages, we model the NFRs and their analysis results using graphical constrained case diagrams as shown in Figure 4.

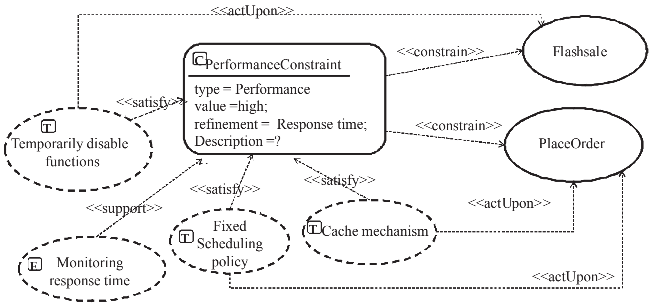
From this example, we cou ld see that our risk driven approach could provide an operational guide to effectively analyze various aspects of NFRs, and the constrained case diagram could give developers an integrated view for NFRs, derived requirements and related functionalities.



(a) Constrained case diagram of security



(b) Constrained case diagram of availability



(c) Constrained case diagram of performance

Figure 4. Constrained Case diagrams for NFRs

V. Discussions and Conclusions

Non-functional requirement is important for building user-satisfied software. However, for its non-behavioral nature, it is relatively difficult to directly implement them in the target software systems. A co mplete analysis of its context, source, related threat and vulnerabilities is very useful and helpful for identifying the tactics to achieve them. In this paper, we propose a risk driven approach to analyzing and refining NFRs. A constrained case di agram is also proposed to provide an integrated graphical view for modeling the analysis results.

The effectiveness of this approach is dependent on the strength and validity of the NFR related knowledge, including the risks, threats, vulnerabilities and corresponding NFR tactics. A deep understanding of the system characteristics and its environment are also i mportant for analyzing NFRs. Ano ther factor is the e xperience of the so ftware engineer. Although there some NFR related knowledge for reference, it also needs software engineer to u nderstand the target system and make decisions during analysis. In addition, for graphical modeling NFRs, the constrained case views are more appropriate to represent the NFRs that associated with specific functionality that denoted by functional use case. For system-wide NFRs, such as maintainability, and portability that are not normally associated with a particular functionality, we could use the template to describe them, and use the constrained case view to document the policy or tactics that help the project achieve the softgoal. In the future work, we will entail validation of the efficacy of the risk driven NFR analysis approach and collect more specific NFR related knowledge to support the process.

Acknowledgment

The work supported by Beijing Natural Science Foundation (4122036); the National Basic Research Program of China under Grant No.2009CB320703; the Scien tific Research Foundation for the Returned Overseas Chinese Scholars, State Education Ministry; the National Natural Science Foundation of China under Grant No.61121063, No.61003065; Specialized Research Fund for the Doctoral Program of Higher Education (No.20101101120027); Excellent Young Scholars Research Fund of Beijing Institute of Technology (No.2010Y0711).

References

[1] L Chung, J do Prado Leite, "On Non-Functional Requirements in Software Engineering", Conceptual Modeling: Foundations and Applications, 2009, pp.363-379.

[2] A.Matoussi, R.Laleau, "A Survey of Non-Functional Requirements in Software Development Process", Technical report,TR-LACL-2008-7.

[3] IEEE Std 830-1993. IEEE Recommended Practice for Software Requirements Specifications.

[4] L. Chung, B. A. Nixon, E . Yu, and J. Mylopoulos, "Non-Functional Requirements in Software Engineering".1st ed. Springer 1999.

[5] Y.Eric,"Towards modeling and reasoning support for early-pahse requirements engineering", Proc.Third IEEE Inl.Symposium on Reqirement Engineering, 1997.

[6] I.Alexander, "Misuse cases help to elicit non- functional requirements", Computing & Control Engineering Journal, 2009, pp. 40-45.

[7] Joe Zou, Christopher J.Pavlovski, "Control case approach to record and model non-functional requirements", Information Systems and E-Business Management, 2008, Volume 6, Number 1, p49-67, 2008.

[8] L.Chung, S.Supakkul, "Representing NFRs and FRs: A Goal-Oriented and Use Case Driven Approach", SERA 2004, LNCS 3647, pp.29-41.

[9] J.Allen,S.Barnum,R.Ellison,G.McGraw,N.Mead, "Software security engineering: a guide for project managers", Addison-Wesley,2008.

[10] Stoneburner G, Goguen A, Feringa A, "Risk management guide for informaion technology systems", Natiaonal Institute of Standards and Technology (NIST), U.S.Department of Commerce, Publication 800-300.

[11] C.U.Smith, L.G.Williams, "Performance Solutions: A practical Guide to Creating Responsive, Scalable Software", 1st E, Addison-Wesley, 2011.

[12] M. Glinz, "On Non-functional Requirements", Proc.5th IEEE International Conference on Requirements Engineering, 2007, pp.21-26 .

[13] F. Armour and G. Miller, "Advanced Use Case Modelling", Addison-Wesley, 2001.

[14] C.Harris,M.Davis,M.Pritchard,M.Rabins,"Engineering Ethics: What? Why?How? And When?", Jounal of Engineering Education, April 1996.

[15] M.Barbacci,M.H.klein,T.A.Longstaff,C.B.Weinstock,"Quality Attruibutes",Technical Report,CMU/SEI-95-TR-021,1995

[16] Malik Hneif, Sai Peck Lee, "Using Guidelines to I mprove Quality in Software Nonfunctional Attributes", IEEE Software 28(6), p72-77,2011.

[17] Object Management Group, "Unified modeling language: Superstructure version2.0", OMG Document,formal/05-07-04,2005.

[18] P.Kruchten, "Architectural Blueprints – The '4+1' V iew model of Software Architecture", IEEE Software, 12(6), 1995, pp42-50.