Representing NFRs and FRs: A Goal-Oriented and Use Case Driven Approach

Lawrence Chung[1] and Sam Supakkul[2]

[1] The University of Texas at Dallas, Richardson, Texas, USA
chung@utdallas.edu
[2] Titat Software LLC, Euless, Texas, USA
ssupakkul@ieee.org

Abstract. As software systems become more complex and important for business and everyday life, the need to better address non-functional requirements (NFRs) become increasing more crucial. However, UML and particularly the use case modeling–the current de facto standard method for functional requirements elicitation and modeling–lacks equally matured modeling constructs for dealing with NFRs. This paper proposes a framework for representing and integrating NFRs with FRs in the use case model at four association points: subject (system boundary), actor, use case, and communicate association. The NFRs can be implicitly associated with other related use case model elements based on the NFR propagation rules proposed to eliminate the need for redundant NFR specifications. A process is presented to demonstrate how to apply this framework, along with an illustration based on a simplified pricing system.

1 Introduction

As software systems become more complex and important for business and everyday life, the need to better address non-functional requirements (NFRs) become increasing more crucial. However, UML [1], particularly the use case modeling–the current de facto standard method for functional requirements elicitation and modeling, lacks equally matured modeling constructs for dealing with NFRs. In use case driven development, NFRs–if addressed at all–are described informally in the *Special Requirements* section of the use case description. This is intended to provide context for the NFRs, but this approach has several drawbacks: 1) non-intuitive reference points for some NFRs such as maintainability or portability, 2) NFRs not represented and organized due to the lack of modeling constructs, 3) NFRs not traceable to other software artifacts, and 4) redundant and error prone when NFRs textual description is duplicated in many use cases that share the same NFRs.

In this paper, we propose a goal-oriented and use case driven framework to address the problems we described above. Instead of using only use case as the context for all types of NFRs, we propose to associate NFRs at four use case model elements to provide more precise context. These NFR association points

W. Dosch, R.Y. Lee, and C. Wu (Eds.): SERA 2004, LNCS 3647, pp. 29–41, 2005.

include system boundary, actor, use case, and communicate association. To properly represent NFRs, we adopt the NFR Framework [2] to represent NFRs as softgoals to be satisficed. To eliminate the need for redundant specification for common NFRs, we propose NFR propagation rules where equal or more strict form NFRs are propagated to applicable use case elements.

A number of proposals have been proposed to integrate NFRs in the use case model. The Language Extended Lexicon (LEL) driven approach [3] first describes the application domain in LEL to provide context for both FRs and NFRs, which are analyzed separately and then integrated in the use case model. NFRs are analyzed visually using the NFR Framework [2], whose functional solutions (operationalation) are represented by new use cases included by the use cases created for FRs. This approach is of close relevance to our work. The main differences between the two approaches are 1) the use of LEL in this approach 2) this approach integrates NFR operationalizations in the use case model while our approach integrates the NFRs themselves. The cross-cutting quality attributes approach [4] adopts the NFR Framework to textually analyze NFRs for "cross-cutting" relevance to one or more use cases. The NFRs are then represented by unnamed use cases with stereotype indicating the type of NFR. These use cases are included by the use cases found relevant during the analysis process. The goal-driven approach [5] identifies initial set of use cases based on functional goals. Additional use cases are created to represent non-functional goals and extend the related FR use cases. The performance engineering [6] annotates quantitative performance constraints to UML diagrams such as communicate associations in the use case model, messages in the sequence diagram, or states in the state machine diagram.

The content of this paper is organized as follows. Section 2 provides a brief overview of the UML use case model and the NFR Framework, the underlying frameworks for our approach. In Sec. 3, we describe the proposed framework in detail to elaborate how to identify NFRs at four reference points in the use case model and how their effect can be automatically propagated to other part of the use case model. Section 4 presents a process for using this framework with an example illustrating how to apply this framework to a simplified pricing system. Sec. 5 briefly describes case-study and feedback based on using the framework in two industrial projects. We then conclude the paper in Sec. 6 with a summary of the contributions and future directions.

2 A Review of the Underlying Frameworks

This section provides a brief review of two underlying frameworks for our approach: UML use case modeling and the NFR framework.

2.1 UML Use Case Modeling

UML use case modeling is the current de facto standard modeling technique for capturing functional requirements (FRs) [1]. Use case diagram, like the one in

Fig. 1, is used to depict actors, use cases, and relationships among them. An actor, denoted by a stick man icon, represents a role played by one or more external entities, which can be human or machine. A use case, denoted by an ellipse, represents a way to use the system that produces an observable result to an actor. Every use case must be invoked either by a stimuli from an actor or automatically invoked by the system through *include* and *extend* relationships. The actors that invoke use cases are *active actors*, whereas the actors that are passive and stimulated by the system to fulfill certain needs are *passive actors*. The relationship that represents the communication between the actor and the use case is called *communicate association*. It is denoted by a directed line with an arrow indicating the direction of the stimuli either from actor to use case in the case of active actor, or use case to actor in the case of passive actor. The system in question is denoted by a rectangular enclosing the use cases to delineate the boundary between external entities and what functionality to be provided by the system.

Use cases can be organized using *include*, *extend*, and *generalization/ specialization* relationships. When the model shows *use case A and B include use case C*, it means that functionality represented by C is common and is included as part of A and B. *Use case A extends use case B* means that functionality represented by *A* is optional and can be included as part of *B*; however, only when a predefined condition for an *extension point* in use case *B* is met. *Generalization/specialization* is used to categorize similar actors or use cases the same way that classes may be organized in UML class diagram. Figure 1 is a use case diagram that shows system functionality of a simplified pricing system. The pricing system allowed the airlines to collaborate with its suppliers over the Internet to manage prices of in-flight service items such as meals, drinks, supplies, and cleaning activities provided by suppliers.

2.2 The NFR Framework

The NFR Framework is a goal-oriented approach for addressing NFRs [7][2]. In this framework, NFRs are represented as *softgoals* to be *satisficed*. Softgoals are considered satisficed when there is sufficient positive and little negative evidence for the claim. To determine satisficeability, *operationalizing softgoals* representing design decisions for achieving the NFR softgoals are identified and analyzed. Contribution are evaluated and trade-offs are made possibly with rationale recorded. The entire process is recorded in a *Softgoal Interdependency Graph (SIG)*. The selected design decisions are then used as the basis for architecture and design. Figure 2 shows an SIG of *serviceability softgoal* for the pricing system described in Sec. 2.1. The light clouds represent NFR softgoals, denoted by nomenclature ***Type[Topic]*** where *Type* is a non-functional aspect (e.g. serviceability, performance) and *Topic* is the context for the non-functional aspect (e.g. Pricing System).

NFR softgoals may be refined, typically either by *Type* or *Topic* at a time, using either AND-decomposition (denoted by a single arc) or OR-decomposition (denoted by a double arc). For example, `Serviceability[Pricing System]` is

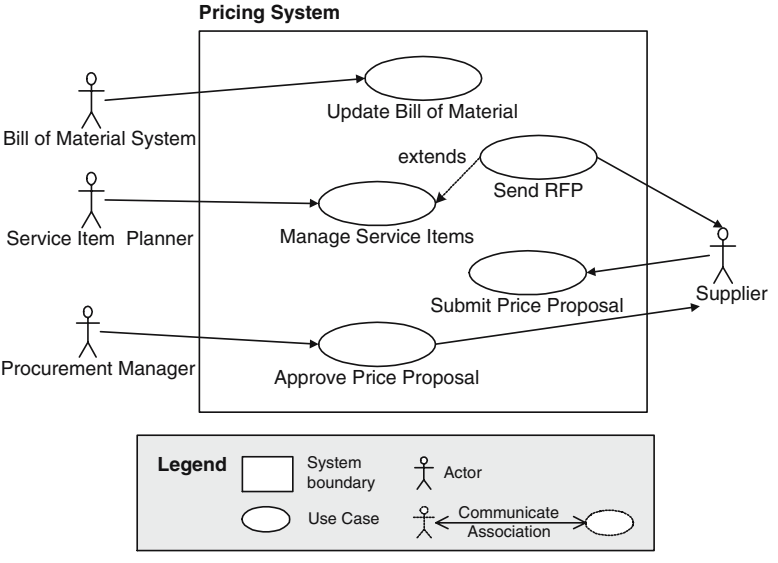

Fig. 1. Use Case Diagram of a Simplified Pricing System

AND-decomposed on the *Type* to `Installation[Pricing System]` and `Tech Support[Pricing System]`. The dark clouds represent operationalizing softgoals. The lines from dark clouds to light clouds indicate the degree the operationalizing softgoal contribute to satisficing the NFR softgoals. The degree of the contribution is indicated as highly positive (denoted by `++` symbol), somewhat positive (denoted by `+` symbol), highly negative (denoted by `--` symbol), or somewhat negative (denoted by `-` symbol). Rationale for any node or link in the SIG contribution are recorded with a dotted cloud called *claim softgoal*.

3 The Goal-Oriented and Use Case Driven Analysis and Design Framework

This section describes how to represent NFRs as softgoals in the use case model. It also presents NFR propagation rules that define how the effect of NFRs can be propagated to other parts of the use case model.

3.1 NFR Association Points

We propose four NFR association points in the use case model including actor, use case, actor-usecase-communication, and subject.

Actor. NFRs can be associated with *active actors* to describe the characteristics of the actor. For example, to represent scalability of a web-baled application

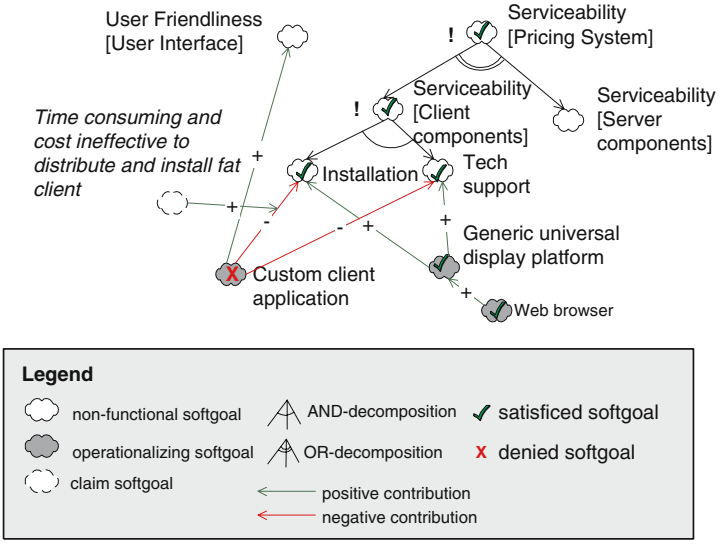

Fig. 2. A Softgoal Interdependency Graph (SIG) for a Serviceability NFR

supporting large number of users, we would associate scalability NFR softgoal with the Customer actor. The actor associated NFR softgoal would then constrain parts of the system that realize all use cases and communication related to this actor. On the other hand, if we associate the NFR with a passive actor, it would place the constraint on the external entity instead. For example, associating a scalability NFR to external Payment Clearing System actor would require the organization responsible for the system to provide a clearing system that meets the NFR. We could also describe characteristics of the actor with NFRs such as *the actor must be a certified network engineer* if domain knowledge is

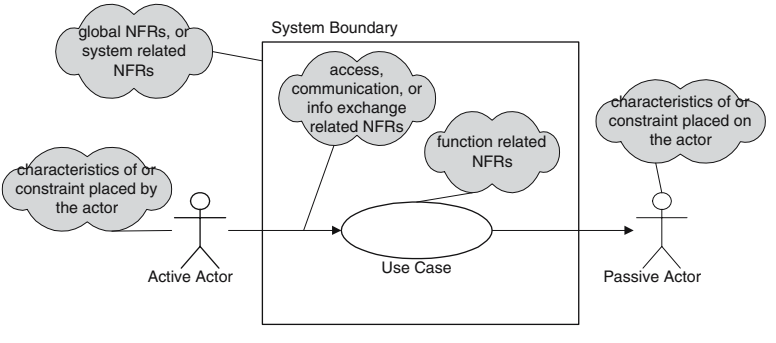

Fig. 3. NFR Association Points and Their Semantics in the Use Case Model

critical to operating a network management system. This provides information for the organization to plan or train the users accordingly.

Use Case. Since use case represents system functionality, it is an ideal reference point for associating function related NFRs such as performance or accuracy. For example, associating *fast response time* NFR to *Withdraw Fund* use case of an Automated Teller Machine (ATM) system would provide a precise context for the designer to pay more attention to the performance aspect of the architecture and software design that realize this particular use case and prevent over design for other use cases.

Communicate Association. Communicate association serves as an association point for NFRs that are related to communication, access, information exchanged with the actor, user interface, or application programming interface (API) with external systems. For example, by associating security NFR to the communicate association between *Customer* actor and *Withdraw Fund* use case, it precisely suggests that the constraint be placed on the architecture and software design that realize the interface to access this use case, but not others.

System Boundary. System boundary serves as an association point for NFRs that are global in nature such as environmental, business process, or software process related NFRs that may be less meaningful to be associated with other association points. Examples of this type of NFRs are cost, maintainability, serviceability, portability, and extendability.

3.2 NFR Propagation

We propose NFR propagation rules in this section so common NFRs that are applicable to multiple use case model elements need to be defined and associated with a use case model element only once. They are then automatically propagated to other related parts of the use case model. This eliminates the redundancy in the requirements model and encourages modular and encapsulation in the design. The rules are specific to different association points as follows.

Actor. When an NFR is associated with an actor, a more or equally strict form of NFRs should be considered for the actors that directly and indirectly specialize the original actor. Fig. 4.a and 4.b show a conceptual and a concrete examples of this NFR propagation. In the diagrams, solid clouds are NFRs that are explicitly associated with the actors, while dotted clouds are the propagated NFRs using this rule. This rule can be defined formally as:

$$(\forall a_1, a_2/Actor)(\forall n_1/NFR)(specialized(a_1, a_2) \ \wedge \ nfr(a_1, n_1))$$
$$\Rightarrow (\exists n_2/NFR)(specialized(n_1, n_2) \ \wedge \ nfr(a_2, n_2))$$

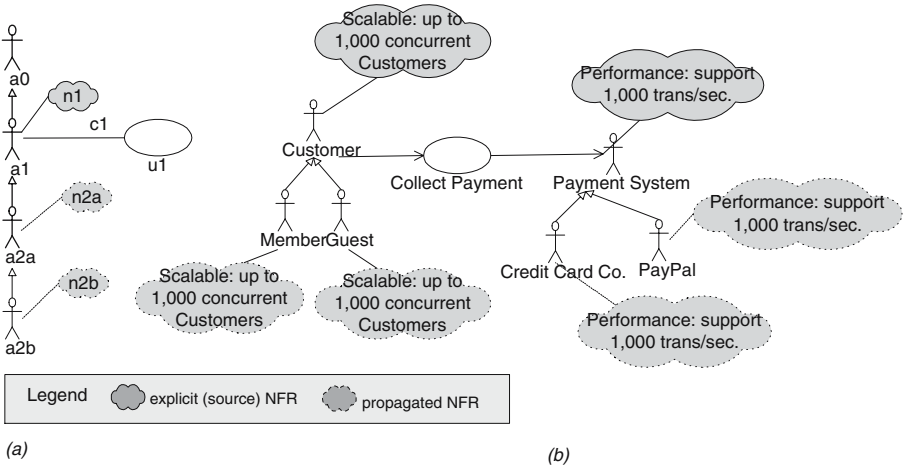

Fig. 4. (a) Conceptual Example and (b) Concrete Example of Actor-Associated NFRs Propagation

Use Case. When an NFR is associated with a use case, a more or equally strict form of NFR should be considered for its directly and indirectly specialized use cases as well as the use cases that are directly or indirectly included or extending the original use case. Fig. 5.a and Fig. 5.b show a conceptual and a concrete examples of this NFR propagation. This rule can be formally defined as:

$$
\begin{aligned}
&[(\forall u_1, u_2/Usecase)(\forall n_1/NFR)(specialized(u_1, u_2) \ \wedge \ nfr(u_1, n_1)) \\
&\Rightarrow (\exists n_2/NFR)(specialized(n_1, n_2) \ \wedge \ nfr(u_2, n_2))] \\
&\wedge \ [(\forall u_1, u_3/Usecase)(\forall n_1/NFR)(include(u_1, u_3) \ \wedge \ nfr(u_1, n_1)) \\
&\Rightarrow (\exists n_3/NFR)(specialized(n_1, n_3) \ \wedge \ nfr(u_3, n_3))] \\
&\wedge \ [(\forall u_1, u_4/Usecase)(\forall n_1/NFR)(extended(u_1, u_4) \ \wedge \ nfr(u_1, n_1)) \\
&\Rightarrow (\exists n_4/NFR)(specialized(n_1, n_4) \ \wedge \ nfr(u_4, n_4))]
\end{aligned}
$$

Communicate Association. When an NFR is associated with an Communicate Association (CA), which links between an actor, say A, and a use case (say U), a more strict form of NFR can be considered and associated with all communicate associations that link between specialized actors of A and specialized use cases of U. Fig. 6.a and 6.b show a conceptual and a concrete examples of this NFR propagation. This rule can be formally defined as:

$$
\begin{aligned}
&(\forall c_1, c_2/CA)(\forall n_1/NFR)(a_1 = actor(c_1))(a_2 = actor(c_2)) \\
&(u_1 = usecase(c_1))(u_2 = usecase(c_2)) \\
&(specialized(a_1, a_2) \ \wedge \ specialized(u_1, u_2) \\
&\wedge \ ca(a_2, u_2) \ \wedge \ nfr(c_1, n_1)) \\
&\Rightarrow (\exists n_2/NFR)(specialized(n_1, n_2) \ \wedge \ nfr(c_2, n_2))
\end{aligned}
$$

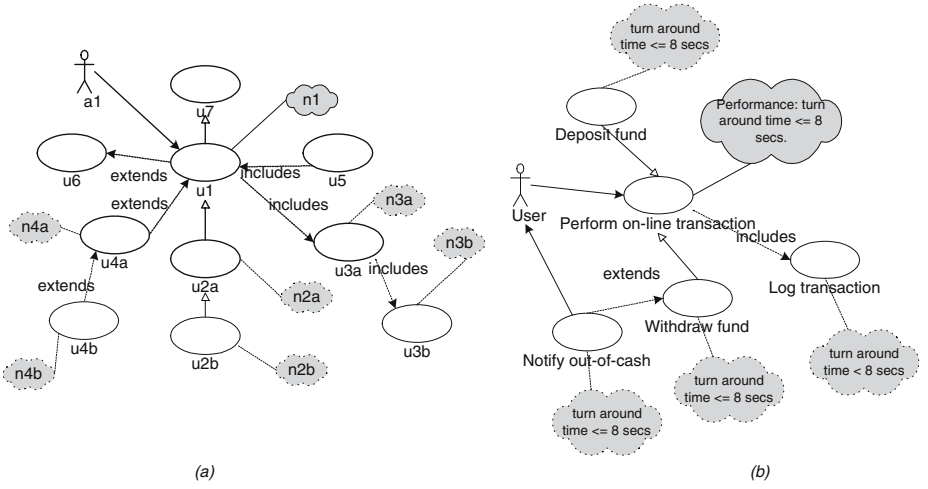

Fig. 5. (a) Conceptual Example and (b) Concrete Example of Use Case-Associated NFRs Propagation

4 A Process for Applying the Framework

This section presents a process for using this requirements representation and integration framework. It is an iterative and interleaving process where refining existing artifacts and repeating previous steps can be performed as needed. Figure 7 is a UML activity diagram depicting the process. The following describes steps in the process using the pricing system presented in sec. 2.1 as an example.

Step 1 - Define System Boundary and Global NFR Softgoals
In this step, we identify the system in question, then define and associate any applicable global NFR softgoals and appropriate criticality. As an example, the customer of the pricing system stated that the new system would support domestic and international users. Therefore, minimizing the distribution of user interface application and field support cost were important. These NFRs were represented by NFR softgoal `Serviceability: minimum client side support for world-wide users` and associated with the system boundary

Step 2 - Identify Actors and Related NFR Softgoals
This step identifies the roles played by external entities. Organize them with generalization/ specialization relationship. Also identify actor related NFR softgoals with appropriate criticality.

Step 3 - Identify Use Cases and Related NFR Softgoals
Identify use cases for each actor. Organize use cases with generalization/specialization, extend, or include relationships. Identify use case or communicate association related NFR softgoals with appropriate criticality. For use cases that are used by multiple actors, a single generalized actor should be introduced to represent the role using those use cases [8]. For the pricing system,

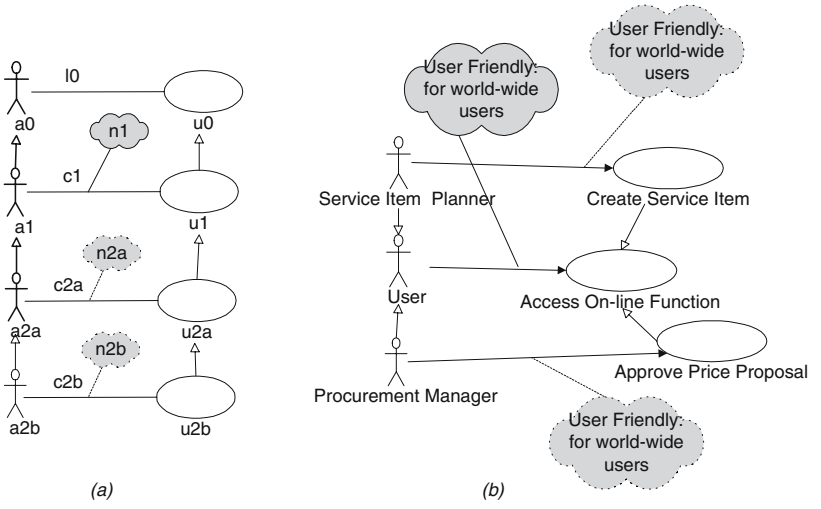

Fig. 6. (a) Conceptual Example and (b) Concrete Example of Communicate Association-Associated NFRs Propagation

the customer stated that for confidentiality and competitiveness, Suppliers must not be aware of each other presence and proposed prices. This NFR was represented as `Confidentiality: suppliers may not see each other identity and proposals` and associated with the communicate association between `Supplier` actor and `Submit Price Proposal` use case. Because the system would be used in many countries, the user interface must be user friendly. This NFR was represented as `User Friendly: support international users` associated with the communicate association between `Supplier` actor and `Submit Price Proposal` use case.

Step 4 - Relocate Common NFRs Softgoals

Revisit previously identified NFR softgoals and determine if any of them should also be associated with other use case elements. For those NFRs that are associated with use cases, define a generalized use case for them, then move the NFR to this new use case. For those NFRs that are associated with actors, define a generalized actor, then move the common NFRs to the new actor. For those NFRs that are associated with Communicate Association (CA), define a new set of generalized use case, generalized actor, and a CA between them, then move the NFR to the new CA. For the pricing system, we determined that the user friendliness NFR should also be applicable to all use cases that are accessible to all human users. Therefore, we defined a new generalized actor called `User` to be associated with the new `Perform On-line Function` use case. We then moved the `user friendliness` NFR softgoal to the new communicate association. Figure 8 shows the result of this NFR relocation.

Step 5 - Refine and Satisfice NFR Softgoals

Use the NFR softgoals identified up to this point as the root NFR softgoals of an SIG. To name the NFR softgoals on the SIG, use the use case elements

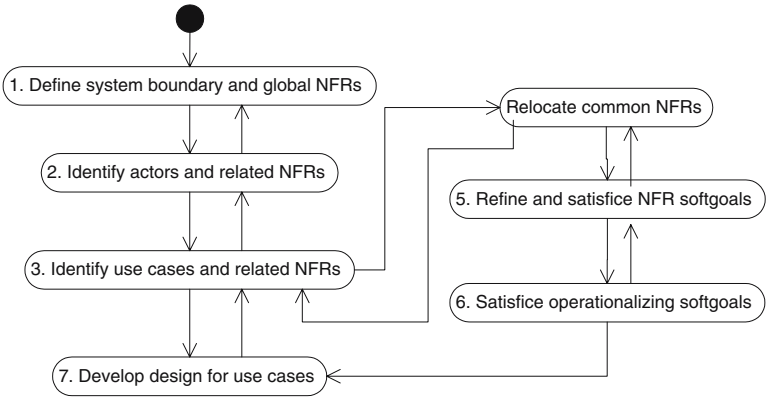

Fig. 7. The Goal-Oriented and Use Case Driven Analysis and Design Process

as *Topic* for the NFRs. If necessary, refine the NFR softgoals using AND or OR decompositions. Then identify operationalizing softgoals, analyze trade-offs, and select the operationalizing softgoals that satisfice the NFR softgoals. We categorize operationalizing softgoals into functional and non-functional operationalizing softgoals. *Functional operationalizing softgoals* are function to be performed by the system or external agents to meet NFR softgoals. For example, authentication operationalizing softgoals may be decomposed to system function to authenticate user at login. *Non-functional operationalizing softgoals* are non-functional decisions such as architectural decisions, personnel or environmental such as using restricted access room or video camera to help satisfice a security NFR softgoal. For the pricing system, we started with the NFR softgoals defined initially in the use case diagram with use case elements serve as the *Topic*. The initial NFR softgoals include `User Friendliness[User-Perform Online Function]`, `Serviceability[Pricing System]`, and `Confidentiality [Supplier-Submit Proposal]`. We then decomposed the NFR softgoals and determined and selected operationalizing softgoals to satisfice the NFR softgoals as shown in Fig. 9. Notice that operationalizing softgoal `Maintain Locale Info` is a functional operationalizing softgoal; therefore, it is mapped to a new use case called `Maintain User Locale`.

Step 6 - Satisfice Selected Operationalizing Softgoals

Map functional operationalizing softgoals to new use cases to represent the new system functions. Iterate to Step 3 to analyze the new use cases and related NFRs. For selected non-functional operationalizing softgoals, map them to concrete architectural or environmental decisions. For example, to satisfice the `Localized input/output[Language]` NFR softgoal for the pricing system, we identified `User-defined localization` operationalizing softgoal, which in decomposed to `Maintaining locale info`, `Appl info stored with locale info`, and `Display info in user locale` operationalizing softgoals. More specific design decisions, including *Maintain user profile* use case, `Using user`

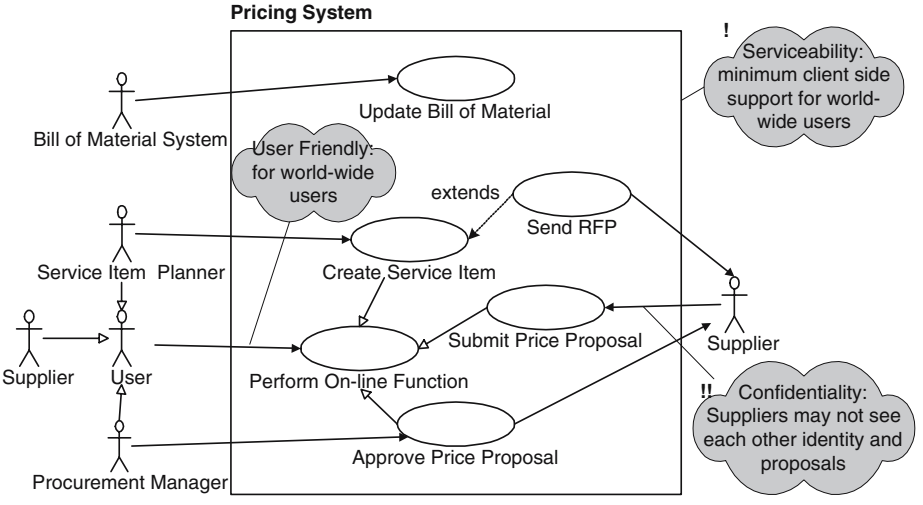

Fig. 8. The Integrated Use Case Diagram with NFRs and FRs

`profile sub-system`, `Multi. prog. lang.`, and `Multi. lang. database` are identified to satisfice the general ones.

Step 7 - Develop Architecture and Design for the Use Cases

Develop a software architecture and design based on the operationalizing softgoals and their satisficing to realize the use cases. For the pricing system, we developed a UML component diagram [9] to identify the sub-systems of the pricing system and their dependency as shown in Fig. 10.a. For each use case, we developed interaction diagram(s) (e.g. sequence diagrams), to envision how the use case would be realized through the interaction among system components. Figure 10.b shows a sequence diagram of the `Submit Price Proposal` use case.

5 Industrial Feedback

We have applied the NFR Framework at a telecom company where we analyzed the existing requirement statements of an ongoing project to determine whether they were correct and complete for meeting the business goals. We developed an SIG to identify business level NFR softgoals and the corresponding operationalizing softgoals. These operationalizing softgoals are the ideal system requirements. We then reverse-engineered existing system level requirements to operationalizing softgoals. The result showed that the two set of operationalizing softgoals did not match, which could indicate missing, wrong, or gold plating requirements. The joint architecture team that reviewed the analysis result appreciated how the decision process and rationale were clearly documented in the SIG. We also introduced the technique for representing NFRs and FRs in the use case model to a software vendor that was developing a pricing system for a

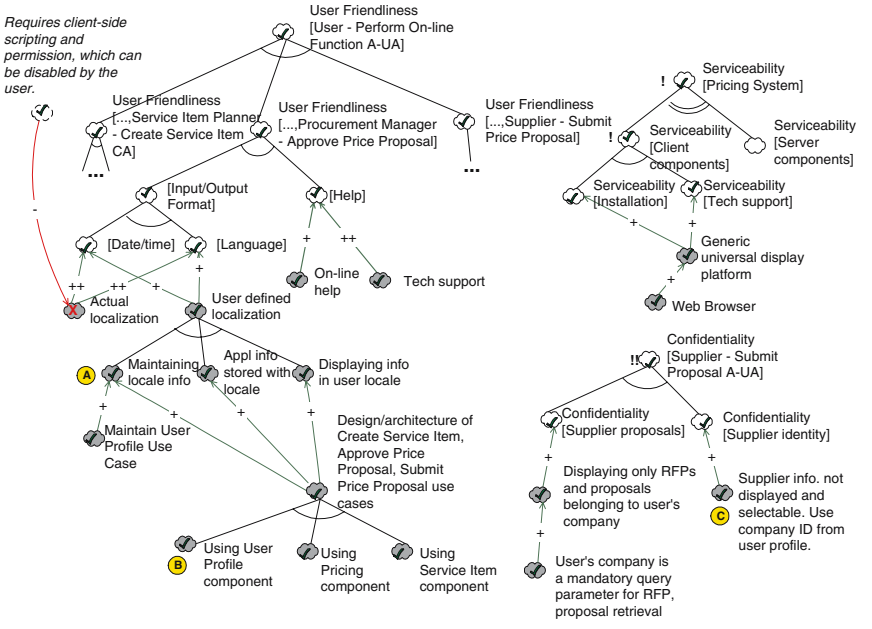

Fig. 9. Softgoal Interdependency Graph (SIG) of the Pricing System

major airlines (its simplified version is presented in this paper). We found that the association points in the use case model were intuitive and useful during requirement elicitation. Some of the works presented in this paper were the result of these feedbacks.

6 Conclusions

In this paper, we have proposed a goal-oriented and use case driven approach for representing NFRs and FRs. Using the use case model as the basis for NFRs identification and integration is important as it is the current de facto standard method for requirements elicitation and modeling. The contributions of this framework include: 1) an intuitive approach for using use case model elements to provide context for NFRs. 2) NFR propagation rules to eliminate redundant specification for common NFRs; and 3) a process for representing and integrating NFRs and FRs. Much remains to be done in this research. It needs to go through more usage to validate and help refine the framework such as in the area of the organization of NFRs along different relationship types in the use case model. We also aim to develop a metamodel to provide precise definition of the relevant concepts, which is an important basis for tool support.

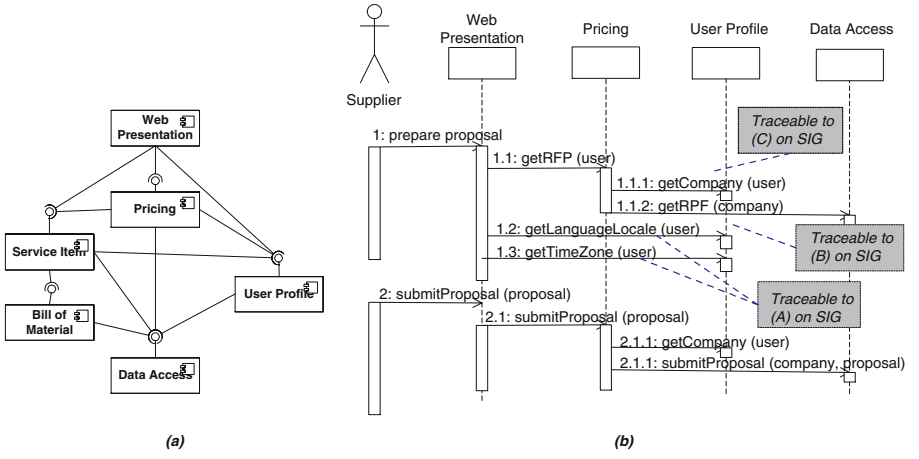

Fig. 10. (a) Component Design of the Pricing System, (b) A behavioral model for Submit Price Proposal Use Case

References

1. Booch, G., Rumbaugh, J., Jacobson, I.: The Unified Modeling Language User Guide. Addison-Wesley (1999)
2. Chung, L., Nixon, B.A., Yu, E., Mylopoulos, J.: Non-Functional Requirements in Software Engineering. Kluwer Academic Publishers (2000)
3. Cysneiros, L., do Prado Leite, J.: Nonfunctional requirements: from elicitation to conceptual models. IEEE Transactions on Software Engineering **30** (2004) 328–350
4. Moreira, A., Brito, I., Arajo, J.: Crosscutting quality attributes for requirements engineering. The fourteenth International Conference on Software Engineering and Knowledge Engineering (SEKE'02) (July 15-19, 2002) 167–174
5. Lee, J., Xue, N.: Analyzing user requirements by use cases: A goal-driven approach. IEEE Software (July/August 1999) 92–100
6. Dimitrov, E., Schmietendorf, A.: UML-based performance engineering possibilities and techniques. IEEE Software (January/February 2002) 74–83
7. Mylopoulos, J., Chung, L., Nixon, B.A.: Representing and using nonfunctional requirements: A process-oriented approach. IEEE Transactions on Software Engineering **18** (1992) 483–497
8. Jacobson, I., Booch, G., Rumbaugh, J.: The Unified Software Development Process. Addison-Wesley (1999)
9. Rumbaugh, J., Jacobson, I., Booch, G.: The Unified Modeling Language Reference Manual. 2nd edn. Addison-Wesley (2005)