

Integrating FRs and NFRs: A Use Case and Goal Driven Approach

Sam Supakkul
Network Surveillance Systems
MCI
ssupakkul@ieee.org

Lawrence Chung
Dept. of Computer Science
The University of Texas at Dallas
chung@utdallas.edu

Abstract

Non-Functional Requirements (NFRs) have been increasingly accepted as crucial to the success of software projects. However, the current state of industrial practice is still focusing mainly on functional requirements (FRs) using UML use cases as the main tool for requirements elicitation and modeling. In order to encourage practitioners to focus more on much deserved NFRs, there is a need for frameworks to provide a smooth transition from the use case modeling. This paper proposes such a framework for integrating NFRs with FRs in the use case model. It proposes that key use case model elements, specifically, actor, use case, actor-use case association, and system boundary, be used as association points to provide intuitive context for the NFRs. The framework specifies the scope of each type of NFR association through the formalization of NFR scope propagation rules that take advantage of relationships between use case model elements (specialization, generalization, extends, includes). A process and illustration are presented to demonstrate how to apply the method to a simplified pricing system.

1 Introduction

Non-Functional Requirements (NFRs) have been increasingly accepted as crucial to the success of software projects. However, the current state of industrial practice is still focusing mainly on functional requirements (FRs) using UML use cases as the main tool for requirements elicitation and modeling. In order to encourage practitioners to focus more on much deserved NFRs, there is a need for frameworks to provide a smooth transition from the use case modeling. Recognizing this need, the research community has gradually focused more on how to integrate NFRs into existing FRs-dominant practice. However, there is a lack of techniques for integrating FRs with NFRs. Some proposed to integrate NFRs in UML class model [1], and some proposed to integrate the requirements in the use case model and interaction model (sequence diagram, activity diagram) [2, 3, 4]. As the use case model is now the established standard for FRs elicitation and modeling; we believe, integrating NFRs in the use case model should provide the simplest and least resistance path for practitioners to adopt a new

framework. For this reason, we consider the use case model as an excellent basis for integration.

Although the proposed methods provide a better starting point for practitioners that may have not addressed NFRs otherwise, improvements should be made in the following areas in order to facilitate the analysis and communication:

Ontological and epistemological. To better represent and reason about NFRs, we need to be able to offer both an appropriate set of ontological concepts-i.e, identify and categorize different NFR related concepts- and a set of appropriate epistemological ways - i.e, organize the large and complex NFRs related aspects.

Preserving the existing principles. For frameworks such as UML that has been accepted by practitioners and researchers, we need to preserve the underlying principles when integrating NFRs into the existing framework. Otherwise, changing their usage and principles could cause confusions and affect existing practices and methods that are based on the accepted principles. For example, use case is traditionality used to represent functional aspect of the system. It should not be used to represent NFR. Also, unnecessarily creating use cases may adversely affect the result of methods such as use case based cost estimation [5].

Traceability to architecture and design. Without ontological and epistemological constructs, it is difficult to establish traceability between requirements and architecture or other design artifacts. We aim to provide modeling constructs and a process to describe how to trace the integrated FRs and NFRs model to architecture and design.

This paper proposes a use case and goal-driven framework for integrating FRs and NFRs. It proposes to associate NFRs at key association points in the use case model-specifically, actor, use case, actor-use case association, and system boundary- to provide an intuitive context for the NFRs. It also takes advantage of the relationship between use case elements (generalization, specialization, extends, includes) to define the scope of the NFR in the use case model to ensure completeness and to make requirements modeling more productive and less error prone. To provide a rich set of ontological and epistemological constructs for modeling and organizing NFR related concepts, we adopt

the goal-oriented NFR framework [6, 7] as it offers extensive modeling constructs for addressing NFRs.

This paper first provides a brief review of UML use case model and the NFR framework in Section 2. The framework is then presented in Section 3. Section 4 presents a process and illustration to demonstrate how to apply the framework to a simplified pricing system based on an industrial project. We then provide a discussion of our experience in using the integration framework at two companies in Section 5. We conclude the paper in Section 6 with a summary of the contributions of the paper and future directions.

2 A Review of UML Use Cases and the NFR Framework

This section provides a brief review of the two underlying frameworks, UML use case modeling [8] and the NFR framework [6, 7].

2.1 UML Use Cases

Use cases capture system functionality or functional goals of the system from the perspective of external entities called actors (human users or automated systems). Use case diagram depicts the use cases with relationship with actors and among themselves.

Figure 1 is a use case diagram that depicts a simplified version of functionality provided by a pricing system developed for a major airlines. The pricing system (denoted by a rectangle surrounding the use cases) allows the airlines to collaborate with its suppliers over the Internet to manage prices charged by suppliers for in-flight service items such as meals, drinks, supplies, and cleaning activity. The actor named Service Item Planner (represented by a stick man) is a role play by authorized airlines users to manage the service items specification through Manage Service Item use case (represented by an ellipse). When service items are created or updated, system automatically sends electronic Request For Proposal (RFP) over the Internet to the suppliers through Send RFP use case, which extends the Manage Service Item use case. The suppliers receiving the RFP sends price proposals (via Submit Price Proposal use case) for the airports that they serve. The airlines users (Procurement Manager actor) then approves or rejects the proposal through Approve Price Proposal use case. Suppliers revise the rejected proposals and re-submit until both sides agree on the prices.

2.2 The NFR Framework

The NFR framework is a goal-oriented approach for addressing NFRs. In this framework, NFRs are represented as “softgoals” to be “satisfied”. Softgoals are considered satisfied when there is sufficient positive and little negative evidence for the claim, and they are unsatisfied when there is sufficient negative evidence and little positive support for their satisficeability. To determine satisficeabil-

ity, design alternatives or decisions (called operationalizing softgoals) are considered, design tradeoffs are analyzed, design rationale is recorded and design choices are made. The entire process is recorded in a “softgoal interdependency graph (SIG)”. The selected design decisions (operationalizing softgoals) can then be used as the framework for architecture and design.

Figure 2 depicts a softgoal interdependency graph (SIG) of serviceability softgoal in the context of the pricing system described in Section 2.1. The light cloud indicates an NFR softgoal, denoted with nomenclature *Type[Topic]* where *Type* is a non-functional aspect (e.g. serviceability) and *Topic* is the context for the softgoal (e.g. Pricing System).

Either *Type* or *Topic* of each NFR softgoals can be refined, one at a time, with either AND-decomposition (denoted with a single arc) or OR-decomposition (denoted with a double arc). For example, as shown in Figure 2, *Serviceability[Pricing System]* is AND-decomposed to *Installation[Pricing System]* and *Tech Support[Pricing System]*. For diagrammatic brevity in this paper, *[Pricing System]* is omitted as it is constant. By the same token, if *Topic* is decomposed, the associated *Type* would be omitted. The dark cloud indicates an operationalizing softgoal. The lines between the dark clouds and the light clouds indicate the degree to which the design decisions corresponding to the dark cloud satisfies the NFR represented by a light cloud. The degree of satisficeability is indicated by the color and +/- indicator. Green color indicates positive contribution while red indicates negative contribution. The degree of the contribution is subjectively indicated as highly positive (denoted with a ++ symbol), somewhat positive (denoted with a - symbol), somewhat negative (denoted with a - symbol), or highly negative (denoted with a – symbol). The selected operationalizing softgoals are then used as the justification for selection of design. For example, in this pricing system example, generic display platform is a justification for selection of web browser in the software architecture.

3 The Goal-Oriented Analysis and Design Framework

To provide precise context for NFRs, we propose that NFRs be integrated at certain points in the use case diagram called NFR Association Points. As requirements engineer come across these locations as part of his normal use case modeling, he should also consider any applicable NFRs. This framework specifies NFR scope propagation rules to ensure requirements completeness. NFRs can then be analyzed using the NFR framework, which would determine the design decisions to be used in the design to realized the use cases (FRs).

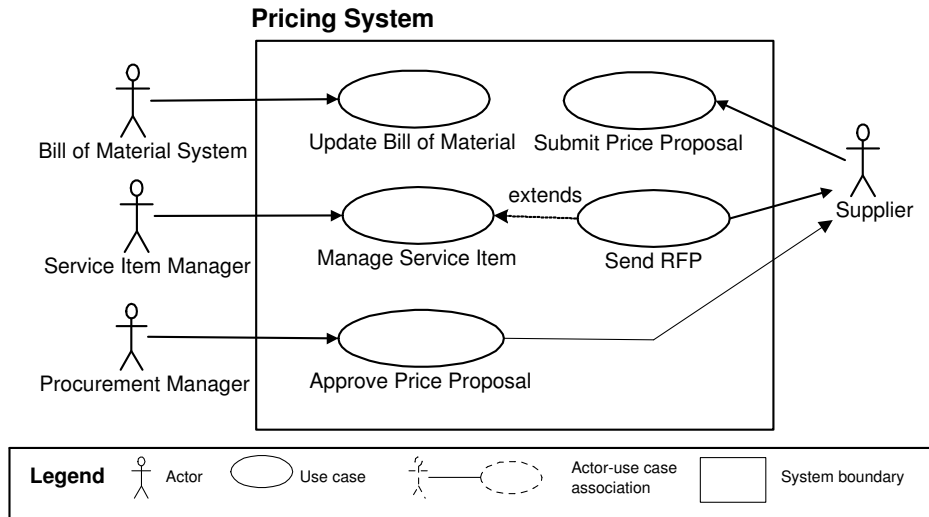


Figure 1: Use Case Diagram of a Pricing System

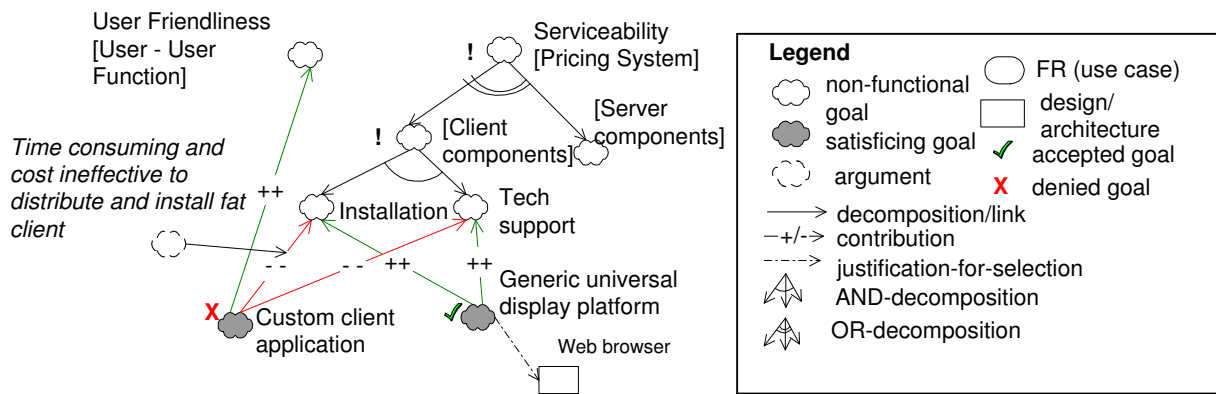


Figure 2: Softgoal Interdependency Graph (SIG) for Serviceability NFR

3.1 NFR Association Points

We propose to associate NFRs with four use case model elements: actor, use case, actor-use case association, and the system boundary as shown in Figure 3.

Actor Association Point. We propose to use actor as an association point for external entity related NFRs such as scalability. For example, associating scalability NFR to Customer actor would indicate that system must handle potentially large number of users accessing system functionality represented by the use cases available to the actor. This provides precise context for the stakeholders; For example, it helps designer to understand where exactly in the system are affected by the NFR, and helps tester to understand where to test the NFR.

Use Case Association Point. Use case can be used as an NFR association point to provide context for function related NFRs. For example, associating fast response time

NFR to Withdraw Fund use case of an Automated Teller Machine (ATM) system would indicate that system must complete the functionality described by the Withdraw Fund use case within an acceptable duration.

Actor-Use Case Association (AU-A) Association Point. Actor-use case association (AU-A) is an association point for NFRs that are related to system access, communication, user/system interface, or information exchanged between system and the actors. For example, associating security NFR to an AU-A between Customer and Withdraw Fund use case would indicate that withdraw fund must be secured, which also precisely implies that user interface for other AU-A not required to be secured.

System Boundary Association Point. System boundary is an association point for NFRs that are global in nature, or not applicable in the context of other use case model elements. For example, associating portability NFR to the

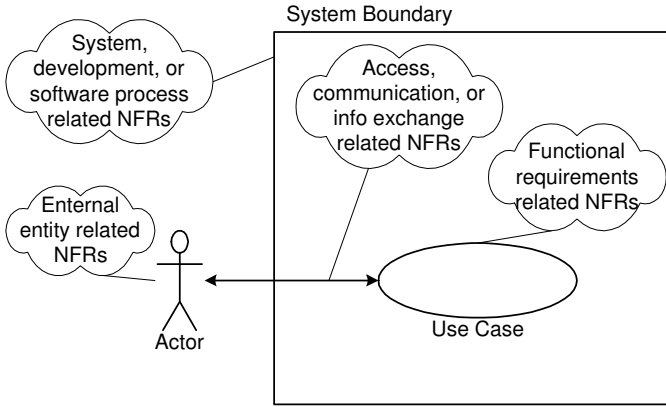


Figure 3: NFR Association Points

system boundary would intuitively specify that the NFR is global that the system must be operational in multiple platforms, which globally affects every part of the system.

3.2 NFR Scope Propagation Rules

One of the problems of the current practice of describing NFR in the special requirements section of the use case description is that requirements engineer must ensure the NFR is described in all use cases to which the NFR is applicable. This is an error prone process as the process is manual and thorough proof read of all use case description is required due to the lack of visual representation of NFR in the UML models.

This framework takes advantage of the generalization/specialization in the standard use case modeling. In order to specify NFR that affects multiple NFR association points, requirements engineer should associate the NFR with a generalized use case element that is specialized by the use case model elements intended to be associated with the NFR. This framework specifies a general property that the NFR would inherently be associated with the specialized elements. In other words, the scope of the NFR is propagated from a generalized element to its specialized elements. However, this property is not applicable to system boundary association point as generalization/specialization is not applicable to this use case model element. Figure 4 shows examples of the scope propagation by NFR association point. The mathematical formalization of the rules are presented in [9].

Actor-NFR Propagation Rule. This framework specifies that NFR associated with an actor is also applicable to all actors that are direct or indirect specialized actors of the generalized actor. For example, if we determined that scalability NFR should be associated with Clerk actor, the scalability NFR would also be associated with Supervisor actor given that it is a specialized actor of the Clerk actor. Figure 4.a shows an example that NFR N_1 , explicitly asso-

ciated with actor A_1 , is propagated to be associated with direct specialized actor A_2 and indirect specialized actor A_3 , but not associated with actor A_0 , which is not a specialized actor of the initial actor A_1 .

Use Case-NFR Propagation Rule. NFR that is associated with a use case is inherently associated with use cases that are specialized use cases of the first use case, and also with use cases that are included by the first use cases. For example, performance NFR, associated with Perform On-line Transaction use case, is inherently associated with Order Product and Cancel Order use cases given that they are specialized use cases of the Perform On-line Transaction use case. If Perform On-line Transaction included Create Audit Trail use cases, the performance NFR would also be associated with the included use case. Figure 4.b shows that NFR N_1 , explicitly associated with use case U_1 , is inherently associated with direct specialized use case U_2 , and indirectly specialized use case U_8 . It is also associated with use case U_3 and U_9 that are directly and indirectly included by the initial use case U_1 .

AU-A NFR Propagation Rule. NFR that is associated with an AU-A is inherently associated with AU-A's related to actors that are direct and indirect specialized actors of the actor that is explicitly associated with the NFR. For example, user friendliness NFR, explicitly associated with AU-A between Clerk actor and Order Product use case, is inherently associated with the AU-A between Supervisor actor and all use cases accessible to the Supervisor actor. Figure 4.3 shows an example of this propagation rule.

4 The NFR Integration Process

This section describes the process of integrating NFRs with FRs and how the integrated requirements may be used later in the analysis and design. Figure 5 is an UML activity diagram depicting the process. It is an iterative and interleaving process where refinement on design artifacts and repetition of previous steps can be performed as needed.

Step 1 - Define system boundary and global NFRs. Identify the system in question and its boundary, which may be organization comprising human performing manual tasks and systems performing automated tasks. The system boundary can also encompass a collection of systems, just one individual system, or even a sub-system. Once the system boundary is determined, define any global NFRs to be associated with the system boundary with appropriate NFR criticality.

Step 2 - Identify actors and related NFRs. Identify roles performed by external entities. Organize them with generalization/specialization relationship. Identify actor related NFRs with appropriate NFR criticality. Actors may need to be re-organized as use cases are determined and appear that they may be accessible by multiple roles.

Step 3 - Identify use cases and related NFRs. Identify

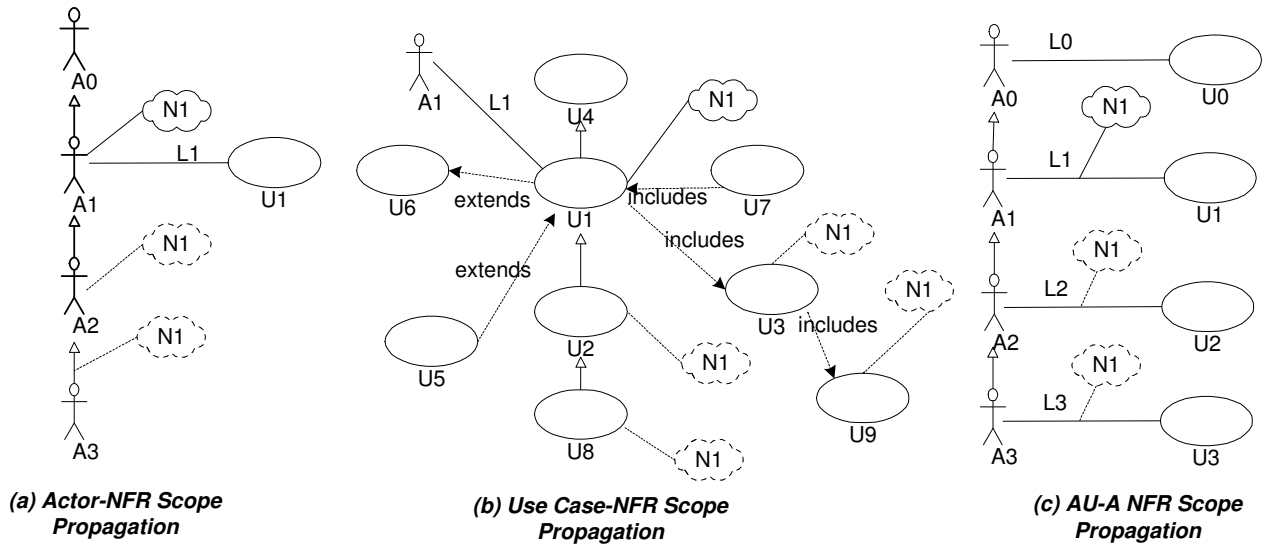


Figure 4: Examples of NFR Scope Propagations

functionality to be performed by the system and represent the functions as use cases, and link them to actors interacting with the use cases. Organize use cases with generalization/specialization, extends, or includes relationships. Identify use case or AU-A related NFRs with appropriate criticality. It is also recommended that problem domain class model be developed to capture real-world entities and their relationship.

Step 4 - Relocate common NFRs. Revisit previously identified NFRs and determine if any of them should also be associated with other use case model elements. For each use case related NFR that should be associated with other use cases, define one or more generalized use cases to be specialized by those use cases, then move the NFR to be associated with the newly defined generalized use case(s). With the NFR scope propagation rules, the NFR would inherently be associated with all specialized use cases. For each AU-A related NFR that should be associated with other AU-A's, define generalized one or more actors that link the actors of the AU-A's to share the NFR. Then move the NFR to be associated with the AU-A of the appropriate generalized actors so that the NFR scope propagation rule would propagate the NFR to the intended AU-A's.

Step 5 - Refine and satisfy NFR softgoals. Use the NFRs associated with the NFR association points in the use case model as the starting NFR softgoals on the SIG. Based on the NFR framework, refine the NFR softgoals using AND or OR decomposition. Identify architectural or design alternatives for each leaf-node NFR softgoal, and their contribution to the NFR softgoals. Evaluate the contributions, then select the operationalizing softgoals that best satisfy the corresponding NFRs softgoals.

Step 6 - Satisfice operationalizing softgoals. With the selected operationalizing softgoals, identify the implementation decisions or solutions that would implement the operationalizing softgoals. Select the solution that best satisfy each operationalizing softgoal. For example, operationalizing softgoal "Using generic display platform" may be realized by the use of either HTML capable web browser or a high performance display engine requiring a proprietary mark up language. Existing NFRs' criticality and other operationalizing softgoals may influence the selection of HTML web browser.

Step 7 - Develop design for use cases. Based on the problem domain class diagram, operationalizing softgoals, and the justification-for-selections identified in the SIG, develop a software architecture that includes sub-systems modeled in UML package dependency diagram. Develop behavioral models with interaction diagrams (sequence diagram or collaboration diagram) to show interaction among sub-systems to realize the use cases.

Illustration

This section illustrates how to apply the framework to the pricing system described in Section 2.1.

Step 1-3. Suppose that the following NFRs are applicable: "Serviceability: minimum client side support for world-wide users" associated with the system boundary, "User Friendly: support international users" associated with AU-A between Supplier actor and Submit Price Proposal use case, and "Confidentiality: suppliers may not see each other identity and proposals" associated with AU-A between Supplier actor and Submit Price Proposal use case.

Step 4. We revisit the previously identified NFRs and determine that the user friendliness NFR should be appli-

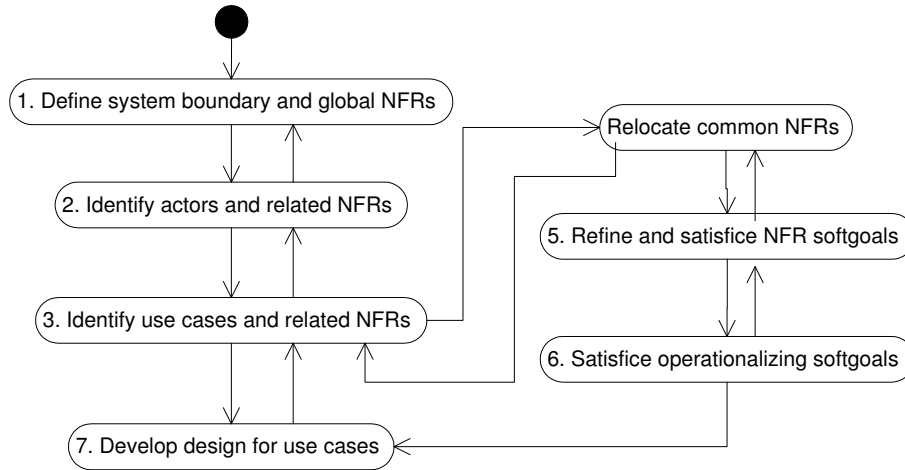


Figure 5: Activity Diagram Describing the Integration Process

cable to all human accessible use cases. Therefore, we define a new use case called Access On-line as a generalized use case of Submit Price Proposal, Create Service Item, and Approve Price Proposal use cases. Because these three use cases are not accessible by the same actor, we define a new actor called User as a generalized actor of Service Item Planner, Supplier, and Procurement Manager actors. The new User actor is associated with the new generalized Access On-line Function use case. We now move the user friendliness NFR to the AU-A between User actor and Access On-line Function use case. After the use case re-organization and NFR relocation, the user friendliness NFR is now inherently associated with the three intended use cases based on the AU-A NFR scope propagation rule described in Section 3.2. Figure 6 shows the use case diagram after this step.

Step 5. To develop a SIG, we use the NFRs integrated in the use case model as the starting point. NFRs would be the Type of the NFR softgoals with Topic defined from the associated use case model elements. We then refine and satisfy the NFR softgoals. Figure 7 shows the resulting SIG from this step. Notice that operationalizing goal Maintain locale info is determined to be necessary, but we do not currently have a use case to perform the maintaining of the user locale. Therefore, this would trigger a need for a new use case called Maintain User Locale use case.

Step 6. We selected User-defined localization operationalizing softgoal that best satisfies the Localized input/output[Language] NFR softgoal in step 5. We now determine the solutions and design decisions to satisfy this operationalizing softgoal. The SIG in Figure 7 shows that “Maintaining locale info”, “Appl info stored locale info”, and “Display info in user locale” operationalizing softgoals have “justification-for-selection” links to design for Access On-line Function use case, which is decomposed to “Using

user profile sub-system”, “Multi. prog. lang.”, and “Multi. lang. database”. This provides design outline for the use case with traceability back to NFRs.

Step 7. For each use case, we develop interaction diagram(s), such as sequence diagrams, to envision how the use case would be implemented through the interaction between sub-systems outlined in the SIG and the relevant sub-systems.

5 Industrial Feedback

We have applied the goal-oriented NFR framework in combination with the agent oriented i^* framework [10] at MCI in a major ongoing project. We used the i^* framework to first identify stakeholders and important agents, and high level functional and non-functional goals to be achieved by the system. We then use the NFR framework to refine the goals, evaluate, and select their satisficing. The final Softgoal Interdependency Graph (SIG) clearly showed what needed to be done in order to achieve the fully automated fault management objectives, and what were being planned in the requirements, and the gaps between them. The result was presented to the joint architecture team. Many expressed great interest in the analysis methods, with one even suggested that this kind of analysis should be performed in all projects.

We also introduced the FRs and NFRs integration framework presented in this paper at a software vendor developing a pricing system for a major airlines. A simplified version is used for the illustration in Section 2.1 and 4. By associating NFRs to the use case model, it made the requirements elicitation sessions effective as it clearly indicates the context for the NFR’s being discussed. One of the software engineers even envisioned that it would be more effective if there was a use case modeling tool that could prompt a list of applicable NFRs when an NFR association point is

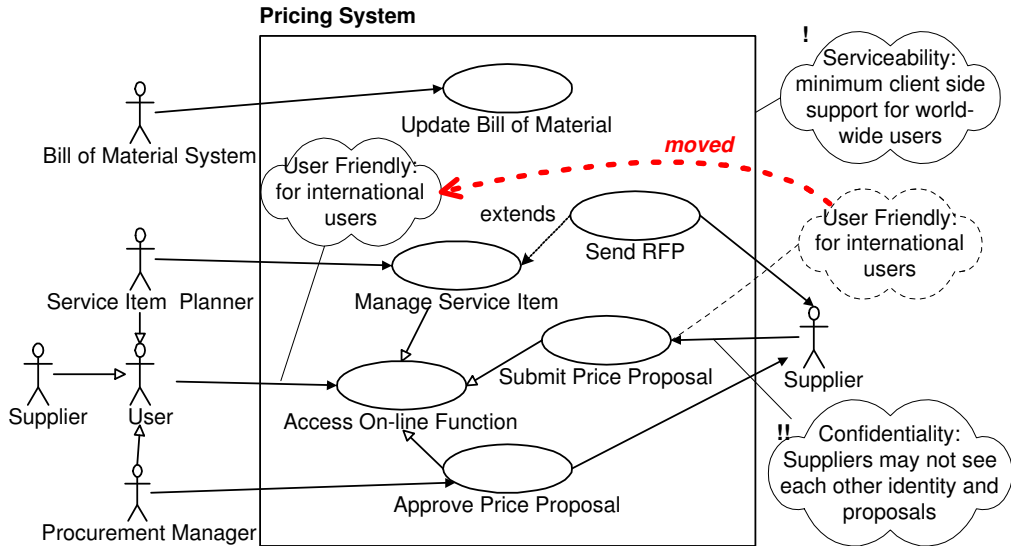


Figure 6: Use Case Diagram After the Requirements Integration

selected. This would remind what NFRs should be considered for the use case element at hand. We envision that for modeling tools to provide such capability, they must maintain a database of NFRs identified by the NFR association points.

Later observation found that while the new combined notation from use case modeling and the NFR framework provides a rich set of modeling constructs, it does require requirements engineer to be familiar with those new constructs and their usages. This may be necessary due to the lack of existing modeling constructs in UML for modeling NFRs and their analysis. Additionally, the formalization of the NFR association propagation rules would be too technical for casual users. But in fact, the intended use of the formalization is for a precise specification, if really necessary (eg, for safety critical systems), but more importantly, for interpreting the scope of the NFRs within the use case model, which may be of interest more to researchers and modeling tool developers. In practice, practitioners need only understand the semantic of the rules in order to develop or understand requirements modeled with the proposed use case modeling extension.”

6 Conclusion

We propose a use case and goal-driven approach for integrating FRs and NFRs. The use case model [8] is used as the basis integration as it is the natural progression from existing use case dominant practices. The contributions of this framework include: 1) intuitive NFR elicitation by using use case model elements (actor, use case, actor-use case association, and system boundary) as the context for NFRs; 2) NFR associated to a use case model element con-

sistently propagated to other elements based on the proposed NFR scope propagation rules to ensure completeness and integrity; 3) providing a rich set of ontological and epistemological modeling constructs through the adoption of the NFR framework [6, 7]; 4) FRs and NFRs traceable to the architecture and design through the goal satisficing and justification-for-selection concepts from the NFR framework combined with the traditional use case driven design.

But much remains to be done in this research. The framework needs to go through more actual usage in practice to gain more experience with the framework strengths and weaknesses, and to fine tune the proposed process and test the effectiveness and validity of the NFR scope propagation rules. The iterative and interleaving nature of the process provides great flexibility, however, at a cost that it requires organizations to depend on skilled modelers to know when to iterate and when to stop the process. The refinement of the process is aimed to provide more precise guidelines so it can be used more effectively by a larger group of practitioners.

Nevertheless, the initial feedback from the sample companies was positive and encouraging. The framework appears to have achieved, to a certain degree, its objectives of providing intuitive and expressive framework for integrating functional and non-functional requirements.

References

- [1] L. M. Cysneiros, J. C. S. P. Leite, and J. M. S. Neto. A framework for integrating non-functional requirements into conceptual models. *Requirements Engineering*, 6(2):97–115, 2001.

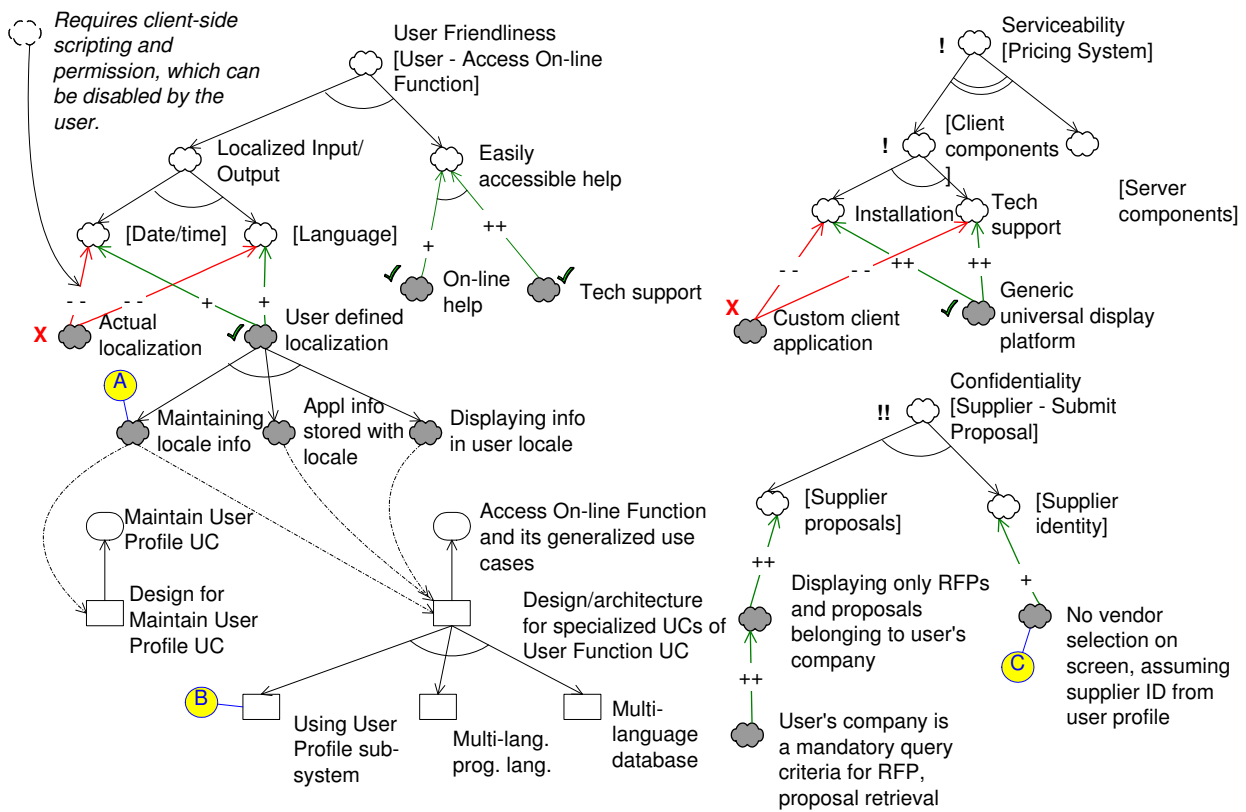


Figure 7: Softgoal Interdependency Graph (SIG) of the Pricing System

- [2] J. Lee and N. Xue. Analyzing user requirements by use cases: A goal-driven approach. *IEEE Software*, pages 92–100, July/August 1999.
- [3] A. Moreira, I. Brito, and J. Arajo. Crosscutting quality attributes for requirements engineering. *The fourteenth International Conference on Software Engineering and Knowledge Engineering (SEKE'02)*, pages 167–174, July 15-19, 2002.
- [4] E. Dimitrov and A. Schmietendorf. Uml-based performance engineering possibilities and techniques. *IEEE Software*, pages 74–83, January/February 2002.
- [5] B. Anda, H. Dreiem, D. I.K. Sjberg, and M. Jrgensen. Estimating software development effort based on use cases - experiences from industry. *Fourth International Conference on the Unified Modeling Language*, Oct. 1-5, 2001.
- [6] J. Mylopoulos, L. Chung, and B. A. Nixon. Representing and using nonfunctional requirements: A process-oriented approach. *IEEE Transactions on Software Engineering*, 18(6):483–497, 1992.
- [7] L. Chung, B. A. Nixon, E. Yu, and J. Mylopoulos. *Non-Functional Requirements in Software Engineering*. Kluwer Academic Publishers, 2000.
- [8] G. Booch, J. Rumbaugh, and I. Jacobson. *The Unified Modeling Language User Guide*. Addison-Wesley, 1999.
- [9] S. Supakkul and L. Chung. Use case and goal-driven integration of FRs and NFRs. *Working Memo*, 2004.
- [10] E. S. K. Yu. Towards modelling and reasoning support for early-phase requirements engineering. *Proceedings of the 3rd IEEE Int. Symp. on Requirements Engineering (RE'97)*, pages 226–235, Jan. 6-8, 1997.