Scope Management of Non-Functional Requirements

M. Kassab[1], M. Daneva[2], O. Ormandjieva[3]
University of Twente[1,2,] Concordia University[3]
[1,2]{m.kassab,m.daneva}@utwente.nl
[3]ormandj@cse.concordia.ca

Abstract

In order to meet commitments in software projects, a realistic assessment must be made of project scope. Such an assessment relies on the availability of knowledge on the user-defined project requirements and their effort estimates and priorities, as well as their risk. This knowledge enables analysts, managers and software engineers to identify the most significant requirements from the list of requirements initially defined by the user. In practice, this scope assessment is applied to the Functional Requirements (FRs) provided by users who are unaware of, or ignore, the Non-Functional Requirements (NFRs). This paper presents ongoing research which aims at managing NFRs during the software development process. Establishing the relative priority of each NFR, and obtaining a rough estimate of the effort and risk associated with it, is integral to the software development process and to resource management. Our work extends the taxonomy of the NFR framework by integrating the concept of the "hardgoal". A functional size measure of NFRs is applied to facilitate the effort estimation process. The functional size measurement method we have chosen is COSMIC-FFP, which is theoretically sound and the de facto standard in the software industry.

1. Introduction

In order to meet commitments in software projects, a realistic assessment must be made of project scope. Such an assessment relies on the availability of knowledge on the user-defined project requirements and their effort estimates and priorities, as well as their risk. This knowledge enables analysts, managers and software engineers to identify the most significant requirements from the list of requirements initially defined by the user. For instance, a requirement deemed critical, but which takes a great deal of implementation effort and poses a high risk, may be a good candidate for immediate resourcing. In most software projects, this scope assessment is performed on the user's functional requirements (FRs), while the non-functional requirements (NFRs) remain, by and large, ignored. The NFR is very important, as it is the key factor that discriminates among competing software products. Empirical reports consistently indicate that improperly dealing with NFRs leads to project failures, long delays or significant increases in final costs [1,2], and hence the need to deal comprehensively with them. While NFRs have a long history in Requirements Engineering (RE), the NFR framework [3] was the first to include a process-oriented and qualitative decomposition approach to handling NFRs. A cornerstone of the framework is the "softgoal" concept for representing the NFR. A softgoal is a goal that has no clear-cut definition or criteria to determine whether or not it has been satisfied. In fact, in this framework, softgoals are referred to as being satisficed rather than satisfied, to underscore their ad hoc nature with respect to both their definition and their satisfaction.

Approaching the specification of NFRs through the NFR framework makes sense for stakeholder' requirements when they describe the system qualities they want built in laymen's terms, qualitatively citing specifications with accompanying verbal descriptions of how the functionality should be used [5]. However, for the satisfaction method to be performed on more concrete basis, stakeholders may agree to identify the NFRs with crisp indicators with defined acceptable values for those indicators to be satisfied; for instance, a scalability requirement may be specified as follows: "The system shall be *capable of providing its functionalities when all 500 business units of our multinational company share it.*" This NFR describes a verifiable criterion (through the acceptable value of 500 business units simultaneously sharing the system) for testing the system's scalability quality. The NFR

33rd EUROMICRO Conference on Software Engineering and Advanced Applications (SEAA 2007)
0-7695-2977-1/07 $25.00 © 2007 IEEE

framework treats all NFRs as softgoals, but, as NFRs tend to be identified with crisp indicators, they are no longer soft and thus they should be modeled as "hardgoals". This is an omission from the NFR framework. Furthermore, there is a tight connection between the use of the NFR's crisp indicators on the one hand, and the quality of the process of acquiring knowledge on effort estimates, priority and risk on the other [8]. Having NFRs concretely defined in terms of the indicators leads to a more realistic assessment. Having performed the assessment, project decision-makers may then call for a revision of the crisp indicators to adjust the acceptable values. For example, if response time is a high priority for the system, then the acceptable value for the response time may need to be adjusted.

The argument of this paper reflects the above discussion and contributes towards achieving the goal of managing the attainable scope of NFRs using the NFR framework. Based on our analysis of the drawbacks to the NFR framework, we recommend improvements to the use of the original concept of the softgoal and to NFR satisfaction. Then, we use the improved NFR framework to acquire knowledge on the effort estimates, priority and risk of NFRs. This knowledge will be instrumental in achieving a proper assignment of project resources, and, as a result, an efficient mapping of NFRs from the requirements domain to the solution space.

The rest of this paper is organized as follows: Section 2 introduces the NFR framework and related work. In section 3, the proposed approach for extending the NFR framework taxonomy is introduced and the characteristics of the goal are provided. Section 4 describes our approach to NFR scope management. Section 5 provides a critical discussion of the approach. Section 6 concludes the paper and discusses our future research agenda.

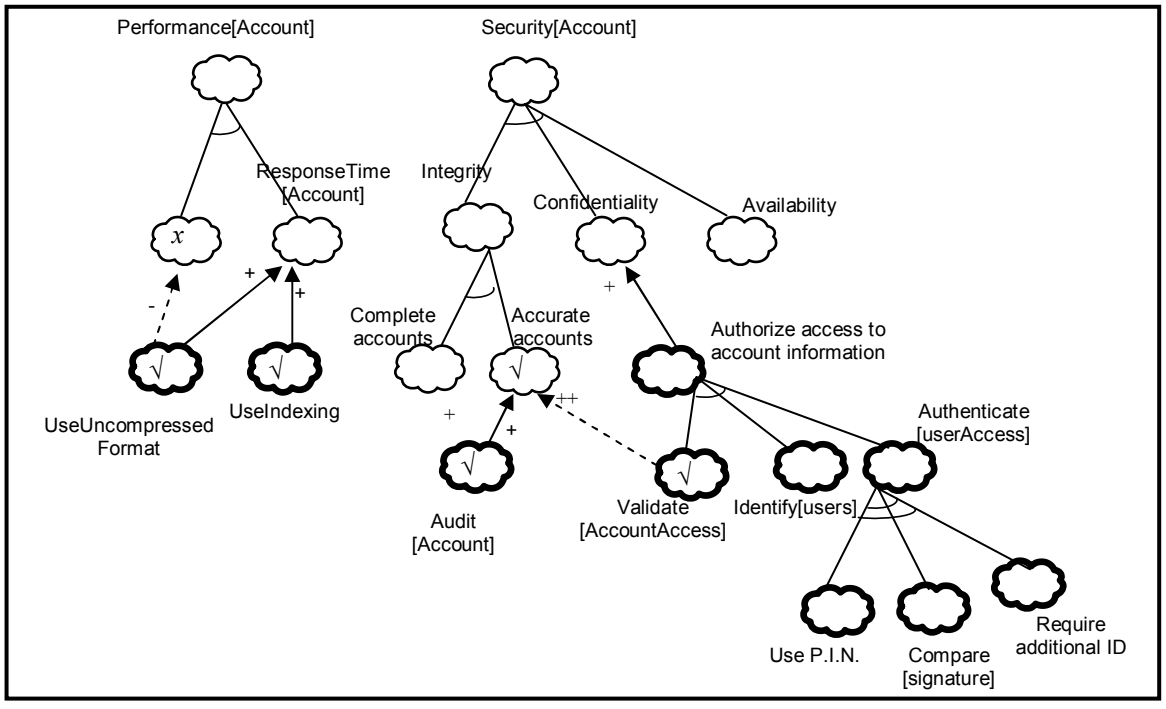

Figure 1: Softgoal interdependency graph for performance and security in a credit card system

2. The NFR Framework

The NFR framework [3] is a process-oriented and goal-oriented approach aimed at making NFRs explicit and placing them at the forefront of the stakeholder's mind. Putting it into practice involves executing the following interleaved tasks, which are iterative:

1. Acquiring knowledge about the system domain, the FRs and the kinds of NFRs associated with the particular system;
2. Identifying NFRs as NFR softgoals and decomposing them into a finer level;
3. Identifying the possible design alternatives for NFRs in the target system as operationalizing softgoals. Operationalizations correspond to

functionalities, operations, processes, data structuring, constraints and agents in the target system;

4. Dealing with ambiguities, tradeoffs, priorities and interdependencies among NFRs and their operationalizations;
5. Selecting operationalizations;
6. Supporting decisions with a design rationale;
7. Evaluating the impact of operationalization selection decisions on NFR satisfaction.

The operation of the framework can be visualized in terms of the incremental and interactive construction, elaboration, analysis and revision of a softgoal interdependency graph (SIG). Figure 1 presents an example of a SIG, with NFR softgoals representing performance requirements and the security of customer accounts in a credit card system. In the SIG, all softgoals are given *Type[Topic1, Topic2,...]* nomenclature. For the NFR softgoal, *Type* indicates the NFR concern and *Topic* indicates the NFR context.

High-level softgoals are refined into more specific subgoals or operationalizations. In each refinement, the offspring can contribute fully or partially, and positively or negatively, towards satisficing the parent. In the example in Figure 1, both *space* and *response time* should be satisficed for the performance to be satisficed. The AND contribution is represented by a single arc, and the OR by a double arc. Further discussion on the original concept of the softgoal is presented in [5, 6]. A major drawback in the NFR framework is that "satisficing" a goal is defined in a fuzzy way. It is impossible to make a system satisfy a softgoal like "usability'" because the goal cannot be clearly defined. So, we also cannot use the satisficing process to plan for this situation. But, if we agree with the stakeholders on some definition of the goal in terms of crisp indicators, then we can set acceptable values for those indicators to satisfy them. Thus, we state that "to satisfice a goal x" is to "satisfy the first acceptable values for defined indicators for x". In section 3, we address the need to model NFRs defined with crisp indicators within the NFR framework.

Further work on NFR's satisfaction and the factors contributing to the satisfaction task is presented in [11, 19, 20, 21, 22].

3. Extending the NFR framework

The tendency to treat NFRs as softgoals can often add ambiguity to the requirements specifications. For example, the response time in a user interface is typically soft, whereas response time requirements in real-time systems can be hard. This situation calls for extending the taxonomy of the NFR framework so that it can identify those NFRs that need to be stated in terms of crisp indicators and their acceptable values.

3.1 Hardgoal notation

In response to this need, we propose an extension of the NFR framework and its softgoal notation. The element that is essential to the extended SIG is shown in Figure 2.

Figure 2: NFR hardgoal notation

To illustrate the use of this extension, we refer back to the credit card example we presented in section 2. Suppose there is agreement with the stakeholders that a good system performance implies the following: "*Maintain the response time within 3 seconds.*" This statement represents an NFR hardgoal requirement that is concerned with the quality of the system under development, and, as such, it needs to be absolutely satisfied. Figure 3 shows the updated graph considering a performance softgoal with the new condition on response time.

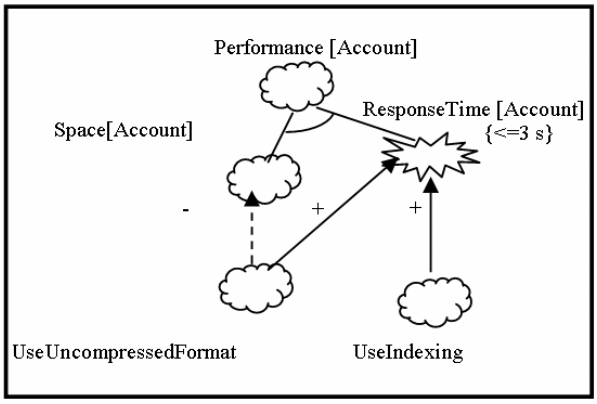

Figure 3: Employing the hardgoal concept in the credit card example.

NFR hardgoals are named using *Type[Topic1, Topic2,...]{Condition1, Condition2,...}* nomenclature. In our extension of the NFR framework's earlier notation, *Condition* indicates a relation to the acceptable values to verify the satisfaction level on NFR achievement. An NFR hardgoal is depicted by a star.

One of the uses of hardgoals is to cope with the need to make explicit the availability of knowledge on the user-defined requirements. For NFRs to be able to be verified when the software is deployed, they have to be clearly defined in terms of crisp indicators. These clearly defined NFRs are hardgoals. Defining the NFR

as a softgoal could serve as an initial step towards our understanding of the NFRs required for the system, but eventually the softgoals will be further defined, and thus presented as hardgoals.

Adopting the new elements suggests that we rename the construction that visualizes the goal-oriented process from SIG to a more expressive title, like Goal Interdependency Graph (GIG). Thus, a SIG is a special type of GIG in which all the stated NFRs are soft.

3.2 Goal characterization

In a GIG, we are stating a goal in compact form, such as described earlier (e.g. "*The response time should be within 3 seconds*"), but it can also be stated in extended form, using a frame-like notation to show more of the goal's characteristics. Below is an example of the long notation form.

Goal Description:	*The response time should be within 3 seconds*.
Author:	*James Xion*
Creation Time:	*15 September, 2006*
Kind:	*NFR goal*
Type:	*ResponseTime*
Topic:	*Account*
Condition:	*<= 3 seconds*
Satisfaction:	*Hard*
Weight:	*0.8*
Label:	√
Functional Size:	*4*

Our approach accounts for the subjective nature of the goals in a relatively straightforward manner by annotating the goal with an identifier indicating its *author* and the *creation time* at which it was suggested.

As stated above, goals have an NFR *type*, which indicates the particular NFR, such as security, reliability, etc., and a subject matter or *topic*. There are three major, and distinct, kinds of goals: NFR goals, operationalizing goals and claim goals. Further definition of the kinds of goals are presented in [3]. NFR goals can be soft or hard. This fact is extracted from the collected requirement statement and specified as the goal *satisfaction*. Operationalizations correspond to functional decisions (e.g. authenticate method) or architectural decisions (e.g. use indexing, use uncompressed forms). Specifying the kind of operationalizations in the *kind of goal* slot is essential for facilitating the functional size calculation, which we will present shortly.

This characterization of NFRs builds the basis for adapting scope management methods to the NFR in a project and adopting them. Scope management, along with the "weight" and "functional size" characteristics, is discussed in section 4, and a set of guidelines for filling in the NFR long notation form is described in section 5.

4. NFR scope management

We discuss NFR scope management in the context of using the NFR framework, and with a strong focus on prioritization, risk and effort estimates.

4.1 NFR prioritization

For design staff to be able to focus their effort on the most important NFRs, stakeholders should provide – at the beginning of the RE process – their input on those NFRs that are critical, and, hence, need to be implemented first. The prioritization decisions can also be biased for social, political or technical reasons. The NFR framework [3] suggests that architects: (i) identify those NFRs that are vital to the system's success as *critical*; and (ii) identify NFRs that deal with a significant portion of the organization's workload as *dominant*. In the framework, NFRs with high priority are identified by an exclamation point (!). The NFR framework deals with prioritization on a qualitative basis.

In our approach, we use the Multi-Attribute Utility Assessment (MAUA) method [4] to manage the prioritization of NFRs in the context of the NFR framework. We selected it because it is intuitive and quantitative, and it has been used in earlier NFR studies. MAUA provides scoring and weighting procedures to evaluate the overall utility of a system. Employing the SIG presented in Figure 1, it is possible to assign weights to each requirement on a scale from 0.01 [wish list] to 1.0 [mandatory] in a top-down process. We start by assigning weights to the top-level NFRs based on the feedback from the stakeholders and/or domain experts. The entire system is deemed to be functioning at 100% when all the requirements at the top level have been satisfied. The offspring for each NFR decomposition are also deemed to be at 100% when they are satisfied. An updated version of Figure 1 with the assigned weights shown for each NFR is presented in Figure 4. The decomposition is depicted by an enclosing box in the figure. A quantitative assignment of the *weight* can be mapped to a qualitative priority later. We would consider that, if the NFR's weight is within the [0.75,1] range, then its priority is High; if it is within the [0.5, 0.75 range,] then the assigned priority is Medium, [0.25, 0.5] Low or [0, 0.25] Negligible.

More discussion on prioritizing NFRs and NFR conflicts is presented in [7, 9].

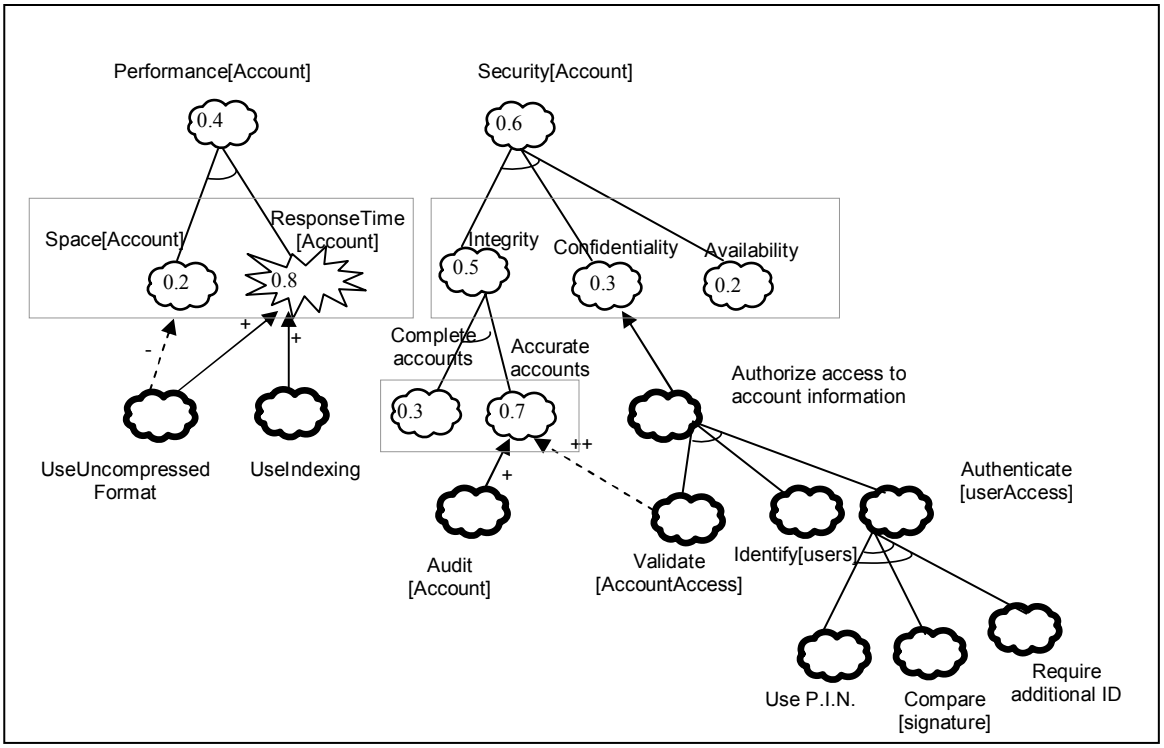

Figure 4: Employing the MAUA in an NFR framework

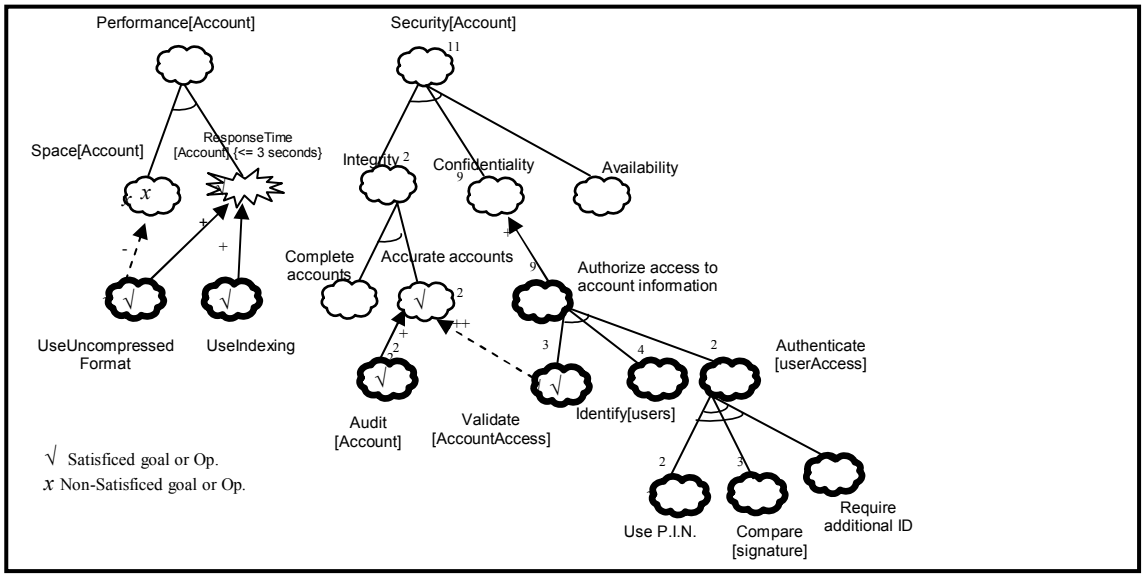

Figure 5: Calculating the functional size of NFRs

4.2 Risk management

The software industry has recognized risk management as a best practice for reducing the surprise factors in software projects [8]. This has to do with how to act early before a concern evolves into a major crisis.

In this paper, we propose a quantitative risk assessment based on the mutual dependency of

NFRs in terms of identifying potential risk in dealing with conflicting NFRs. During software execution (that is, when the executable version of the software is running), hardly any requirements manifest in isolation. Normally, the provision of one NFR may well affect the level of provision of another (e.g. increasing security may increase response time). We refer to this mutual dependency as non-orthogonality.

We propose a function M to map each pair of the identified NFRs to values "+", "-" or " " to indicate the constructive interaction, damage, or no interaction between two NFRs at a certain functionality respectively. M: $\{(NFR_i, NFR_j)\} \rightarrow \{\text{"+"}, \text{"-"}, \text{""}\}$. We state that the negative interaction at runtime poses a risk which has to be considered. This is the case when two or more NFRs affect the same functionality and interact in a negative way among themselves during execution time. This happens mainly because the effort required for the integration process would highly depend on the level of interdependency between the NFRs, and, more specifically, on the defined conflicts between them. To objectively assess NFR conflicts, we propose to use the local conflict measure [23], which reports on the level of conflict (LLC: Local Level of Conflict) for each piece of functionality, based on the list of NFRs that interact at this functionality:

$$LLC(f) = \left| \{(NFR_k,NFR_l) \bullet NFR_k,NFR_l \in NFRs_{at\ f} \wedge M(NFR_k,NFR_l) = \text{"}-\text{"}\} \right| / n$$

In this formula, *n* is the cardinality of the set of all pairs of NFRs at functionality f (the order is ignored to avoid duplication). We can relate the complexity of an arbitrary functionality to other functionality complexities in the system using the following formula:

$$Complexity\ (f) = \left| \{(NFR_k,NFR_l) \bullet NFR_k,NFR_l \in NFRs_{at\ f} \wedge M(NFR_k,NFR_l) = \text{"}-\text{"}\} \right| / \sum_{j=1}^{n} \left| \{(NFR_k,NFR_l) \bullet NFRq,NFRr \in \{NFR_j\} \wedge M(NFR_q,NFR_r) = \text{"}-\text{"}\} \right|$$

The proposed measurements help in obtaining quantitative data that are supposed to direct the effort towards better design strategies and decisions. For example, high complexity values identify those pieces of functionality that pose more risk to the project; these pieces can be closely reexamined by architects to see which combinations of possible architectural options provide the best match; consequently, project managers may decide that more human resources, time or money needs to be dedicated to developing those pieces of functionality.

The collected quantitative data on NFRs lets the stakeholders plan for actions on how to minimize the likelihood or impact of these potential problems. Like the risk management due to FRs, NFR risk assessment makes it possible to focus on controlling the most serious risks first, thereby achieving better scope management of the requirements.

4.3 Effort estimation

The effort needed to develop a hardgoal NFR clearly depends on the size of its operationalization. To the best of our knowledge, we are the first to attempt to measure the size of NFR operationalizations. In our approach, we apply the COSMIC FFP method for functional size measurement [10], which does the following: (i) it takes as input those FRs that result from NFR decomposition; and (ii) yields as output the contribution of the NFR to project size, and, ultimately, to the estimated effort to build the system. COSMIC FFP sees each software functional process as a sequence of events, starting from the trigger and consisting of four data movement types: Entry, Exit, Read or Write. An Entry moves a data group, which is a set of data attributes, from a user across the boundary into the functional process, while an Exit moves a data group from a functional process across the boundary to the user requiring it. A Write moves a data group lying inside the functional process to persistent storage, and a Read moves a data group from persistent storage to the functional process. The unit of measurement is the data movement denoted by the symbol Cfsu (Cosmic Functional Size Unit). To illustrate the use of COSMIC FFP within the NFR framework, we refer to the credit card system presented in section 2 and we measure the functional size for those operationalizations that correspond to functional processes/functions. In this example, suppose that the "Compare Signature" operationalization will be automated and will thus correspond to a functional operation. This operation requires three data movements: (i) Entry, for the signature to be compared; (ii) Read, for the signature to be compared with the one on the disk; and (iii) Exit. So, the functional size for "Compare Signature" is 3 Cfsu. Similarly, we carry out the measurement for all non-decomposable operationalizations that correspond to functional operations/functions.

Calculating the functional size of NFRs is a bottom-up measuring process in which the selected operationalizations are aggregated to calculate the sub-NFRs and their respective NFRs until we obtain the final value. The aggregation of the size values is

theoretically valid because the COSMIC-FFP unit of measurement, the Cfsu, is on the ratio scale [17]. Figure 5 illustrates this process.

To make sure we properly integrate the functional size measurement process into the NFR framework, we suggest that size and effort estimation be performed right after task 3 and before task 4 in the NFR framework process presented in section 2. The measurement data will provide the rationale required for selecting the appropriate operationalizations. For example, suppose that the two operationalizations "Compare Signature" and "Use P.I.N" are 3 Cfsu and 2 Cfsu in size respectively. We have to choose one operationalization to satisfy "Authenticate". We then may well consider choosing "Use P.I.N", as it has a smaller functional size. Bearing in mind that effort is a function of size [8], choosing the smaller operationalization in terms of functional size implies that less effort needs to be invested in the implementation of the particular NFR. More on using the COSMIC-FFP for measuring the functional size of NFRs is presented in [18].

Furthermore, we use the functional size numbers to obtain a rough estimate of the effort needed to implement the NFRs. Effort is usually measured in person-months; earlier experiments suggest that one COSMIC-FFP size unit (Cfsu) requires approximately 2-3 person-months; this number varies from organization to organization and from project to project. Rough effort estimation, then, is obtained from size measurement by using a simple mathematical operation. For example, to implement "Use P.I.N" (an NFR with size 3) would take between 6 and 9 person-months. However, we make a note that this is an early estimation only, and, as such, it should be treated with great care [Jon98]. To obtain a more precise effort estimate would require collecting industrial statistical data on the NFR size and the effort required to incorporate the NFRs into the solutions. This is our most important topic for our future research.

Effort estimation data can be seen as quantitative feedback which design staff need in order to plan for and control the achievement of the NFRs. We therefore take it as an additional criterion to consider in NFR scope management.

5 Discussion

To properly manage the scope of NFRs in the RE process within a project, a revised set of guidelines needs to be provided. Their purpose is to build the basis for a systematic process of transforming the vague stakeholders' NFRs statements into more precise and objective-enabled requirements definitions. This motivated us to revise the list of the seven tasks discussed in section 2, as explained below.

5.1 Revising the set of guidelines

Our revision includes the addition of one new task and the enhancement and augmentation of a few existing tasks. Our revised task list includes the following:

(i) Perform NFR tasks 1 to 3, as discussed in section 3. While doing so, use the NFR long notation form to describe the attributes from Goal description to Weight.

(ii) Apply the functional size measurement method on the operationalizations, and calculate the functional size of the NFRs, as discussed in section 4. The functional size is to be reported in the NFR long notation form.

(iii) Perform NFR framework tasks 4 to 7, as discussed in section 2. Use the long notation form to fill in the label characteristic indicating the decision on the level of satisfaction.

5.2 Viewpoints to adopt when managing NFR

State-of-the-art functional size measurement (FSM) practices [14,15,17] suggest that a choice be made between two perspectives which estimators may adopt when approaching the NFRs in a project. The first (and still predominant) viewpoint on how to quantify the NFRs implies: (i) that an NFR be first decomposed into FR; and then (ii) an FSM method be used to size both FR and NFRs. We demonstrated how this works in section 4.3, with the use of COSMIC FFP to size the operationalization of one NFR.

The alternative viewpoint, which has most recently attracted attention in the FSM literature [8], assumes that in each project there are always some NFRs which should not be decomposed into FRs when sizing. According to this viewpoint, these NFRs are considered to be criteria for making architectural design decisions. Thus, instead of decomposing them into FR, it makes more sense to treat them as contextual factors expected to introduce uncertainty into the estimation process [8]. To judge how significant these uncertainties (due to NFRs) are, we need to identify and report these NFRs in the final project estimate. So, the final estimate would include two components: (i) the Cosmic FFP due to FR; and (ii) the sizing numbers of the NFRs.

Our paper agrees with this second viewpoint on quantifying NFRs. It is also consistent with the position taken by RE researchers regarding NFRs [1,11,12,16], according to which not all of them should be decomposed into FRs. If NFRs serve as norms [11] and as criteria for making architectural design choices, then they should not be decomposed into FRs. Examples are global NFRs like survivability, multi-currency reporting and customizability. Of course, it is possible to decompose an NFR which says that "the system shall support multiple currencies" into an FR like "each user is offered the functionality to select a currency, to select all documents that should use this currency, to generate reports in this currency," and so on. However, it makes more sense to consider this NFR as a criterion for exploring and making choices over alternative architectural options. Our motivation to do so rests on the following observations: (1) the above decomposition into FR (which is needed for effort estimators and is used for size counting) refers to functionality that the users did not ask for at the time of RE; (2) the NFR is a norm to which the system must conform [12]; (3) the currency in an application tells us about the project context, hence it may point to a contextual factor that may well be a source of risk [8] to obtaining realistic size and effort estimates. Moreover, global business information systems which typically produce currency-specific reports for specific user groups should be able to prepare reports according to the accounting standards adopted by the user group. The design architects must then choose a way to set up this multi-currency NFR.

6 Conclusion

Existing NFR approaches fall short when characterizing and quantifying hardgoal NFRs. These approaches primarily only adequately address the softgoal NFR. They also lack quantitative support for objective analysis and decision-making. We propose a solution to these issues and elaborate an extension to the NFR framework to allow modeling and analysis of hardgoal NFRs. We have used the extension to devise a new approach to the scope management of NFRs which takes into account three criteria, namely, customer-defined priority, size and risk. Drawing on the revisited conceptual foundations, we have also proposed guidelines as to the techniques to be present in those requirements modeling approaches which are likely to employ the extension of the NFR framework. Our research activities planned for the immediate future include: (i) carrying out case studies at company sites to extend our understanding of the problems and solutions in managing the scope of NFRs; (ii) exploring how NFRs impact the total cost of projects; and (iii) defining a process for NFR conflict resolution based on objectively assessed priorities. Each new case study is expected to shed light on the problem of estimating hard goal NFRs and help form "the bigger picture" of industrial NFR practices, the characteristics of NFR conflict resolution processes and the problems of the size and effort estimation areas.

References

[1] Lindstorm, D. R., Five "Ways to Destroy a Development Project", *IEEE Software*, September 1993, pp. 55-58.

[2] Breitman, K. K, Leite J. C. S. P. and Finkelstein A. "The World's Stage: A Survey on Requirements Engineering Using a Real-Life Case Study". *Journal of the Brazilian Computer Society* No 1 Vol. 6, Jul. 1999, pp.13-37.

[3] Chung, L., B. A. Nixon, E. Yu and J. Mylopoulos, *Nonfunctional Requirements in Software Engineering*. Kluwer Academic Publishing, 2000.

[4] Ryan, A. J., "An Approach to Quantitative Non-Functional Requirements in Software Development", *Proceedings of the 34th Annual Government Electronics and Information Association Conference*, 2000, pp.13-20.

[5] Mylopoulos, J., Chung, L. and Nixon, B., "Representing and Using Nonfunctional Requirements: A process Oriented Approach". *IEEE Trans. S.E.* 18, 6, 1992, pp. 483-497.

[6] Rosa, N. S., Cunha, P. R. F. and Justo, G. R. R., "Process NFL: A language for Describing Non-Functional Properties". *Proc. 35th HICSS*, IEEE Press, 2002, pp.3676-3685.

[7] Moreira, J. Araujo and I. Brito, "Crosscutting Quality Attributes for Requirements Engineering", *In 14th Int. Conf. on Soft. Eng. and Knowledge Eng.*. 2002. pp. 167-174.

[8] Pfleeger, S. L., F. Wu and R. Lewis, *Software Cost Estimation and Sizing Methods: Issues and Guidelines*, RAND Corporation, 2005.

[9] Daneva, M. M. Kassab, M. L. Ponisio, R. J. Wieringa and O. Ormandjieva, "Exploiting a Goal-Decomposition Technique to Prioritize Non-functional Requirements". *Proc. of the 10th Workshop on Req. Engineering*, Toronto, Canada, May' 07.

0-7695-2977-1/07 $25.00 © 2007 IEEE

[10] Abran, A., Desharnais, J.-M., Oligny, S., St-Pierre, D. and Symons, C., *COSMIC FFP – Measurement Manual (COSMIC implementation guide to ISO/IEC 19761:2003*), École de technologie supérieure – Université du Québec, Montréal, Canada, 2003, URL: http://www.gelog.etsmtl.ca/cosmic-ffp/manual.jsp .

[11] Mylopoulos, J., "Goal-oriented Requirements Engineering", Keynote at the *14th IEEE Int. Conf. on Req. Eng.,* IEEE Comp. Society Press, 2006.

[12] Wieringa, R., The Declarative Problem Frame: "Designing Systems that Create and Use Norms", *Proc. of the 10th IEEE Int. Workshop on Software Specification and Design*, IEEE Computer Society Press, 200, pp. 75-85.

[13] Jones, C., *Software Assessment, Benchmarks, and Best Practice*, Addison-Wesley, 2000.14. ISBSG, Practical Software Estimation, 2nd Ed., Int. Software Benchmarking Standard Group, 2006.

[14] ISBSG, *Practical Software Estimation*, 2nd Ed., Int. Software. Benchmarking Standard Group, 2006.

[15] FISMA, *Experience Situation Analysi*s, Finnish Software Metrics Association, Helsinki, 2001.

[16] Glinz, M., "Rethinking the Notion of Non-Functional Requirements", *Proc. of the 3rd World Congress for Software Quality*, Munich, Germany, 2005, Vol. II, pp. 55-64.

[17] Fenton N.. and, S. L. Pfleeger, *Software Metrics: A Rigorous and Practical Approach*, PWS Publishing, 2nd edition, revised printing, 1998.

[18] Kassab M., Ormandjieva, O., Daneva M. and Abran, A., "Size Measurement of Non-functional Requirements and Their Testing with COSMIC-FFP". Submitted to the *International Conference on Software Process and Product Measurement*, Mallorca, Spain, November 2007.

[19] Azar, J., R. K. Smith and D. Cordes, "Value Oriented Prioritization*"*, *IEEE Software,* Jan, 2006.

[20] Lehtola, L., M. Kauppinen and S. Kujala, "Requirements Prioritization Challenges in Practice", *Proc. of the 5th Int'l Conf. on Product Focused Software Process Improvement (PROFES)*, April 2004, pp. 497-508.

[21] Berander, P. and A. Andrews, "Requirements Prioritization", in Engineering and Managing Software Requirements, ed. Aurum, A., and Wohlin, C., Springer Verlag, Berlin, Germany, pp. 69-94.

[22] Davis A., "The Art of Requirements Triage", *IEEE Computer*, 36 (3), March, 2003, pp. 42 – 49.

[23] Kassab, M., O. Ormandjieva and C. Constantinides, "Providing Quality Measurement for Aspect-Oriented Software Development". *Proceedings of the 12th Asia-Pacific Software Engineering Conference,* December 15-17, 2005, pp. 769-775.