Enabling Trade-off Analysis of NFRs on Models of Embedded Systems

Mehrdad Saadatmand, Antonio Cicchetti, Mikael Sjödin
Mälardalen Real-Time Research Centre (MRTC)
Mälardalen University, Västerås, Sweden
{mehrdad.saadatmand, antonio.cicchetti, mikael.sjodin}@mdh.se

***Abstract*—Satisfaction of Non-Functional Requirements (NFR), is a key factor in successful design of embedded systems. This is mainly due to the constraints and resource limitations in these systems. A design that cannot achieve functionality of the system under these limitations is actually a failure. Therefore, NFRs in design of embedded systems deserve special attention. However, one big issue is that NFRs are interconnected and cannot be considered in isolation; especially that they can have direct impacts on each other such as security and performance. This means that a careful balance and trade-off analysis among NFRs is necessary. In this paper, we focus on this need and identify what information about NFRs is required in order to perform trade-off analysis. We propose and explain our in-progress approach to incorporate this information into system models in order to enable trade-off analysis. Our approach is based on UML profiling method to annotate model elements with necessary information.**

***Index Terms*—Non-Functional Requirements, Trade-off Analysis, UML, Embedded Systems, MDE.**

I. Introduction

Successful design of embedded systems depends heavily on satisfaction of their non-functional requirements. This is mainly due to the constraints and limitations of these systems in terms of available resources [1]. Therefore, an embedded system needs to achieve its functionality under these limitations and NFRs. The problem is that each design decision has impacts on the system's NFRs and the system designer should be able to identify and evaluate these impacts. For example, if there is an execution time constraint on a component responsible for sorting some numbers then choosing a slower algorithm than a more optimized one can lead to violation of system requirements and failure of its desired functionality. This situation becomes more complicated when we realize that NFRs not only crosscut different parts of the system, but also have impacts on each other. For example, choosing an time-optimized and faster sorting algorithm might help with satisfying timing requirements but may require more memory and thus violate memory constraints. Also, an NFR such as security crosscuts different parts of a system and affects some other NFRs like performance [2].

In this paper, we propose a UML profile [3] for trade-off analysis of NFRs by focusing on the needs of embedded systems and the information and answers that trade-off analysis of NFRs in these systems should be able to provide. Enabling trade-off analysis of NFRs through a UML profile has several key benefits. UML is a standard modeling solution already used by industry and there are many design and analysis tools based on it. By offering a UML-based solution for trade-off analysis of NFRs, it becomes possible to make use of currently available tools. Also, the learning curve for developers already using UML will be less which implies both cost and time savings for companies [3]. Moreover, the approach we propose here tries to extend design models with necessary information to enable trade-off analysis of NFRs, and thus enables using already designed system models and saves modeling efforts. We propose stereotypes in our profile by which model elements can be decorated and related with necessary information that enable designers to query models for specific information about NFRs and trade-off analysis of them.

The remainder of the paper is structured as follows. In section 2, we discuss the required characteristics and information that a solution for managing trade-offs of NFRs should be able to offer considering the complications and needs of embedded systems. In section 3, profile structure and concepts are explained. Section 4 shows an application of the profile in a portion of a mobile phone design and a trade-off analysis is performed on the annotated model. In section 5, we will have a look at other related works and finally in section 6, future works are explained.

II. Characteristics of Possible Solutions

In this section we identify the key characteristics that a model-based solution for representation and trade-off analysis of NFRs should have.

Traceability of design decisions related to an NFR: considering that NFRs usually crosscut different parts of the system, the designer should be able to understand which parts of the system contribute (both positively and negatively) to a specific NFR. For example, an encryption component is intended to satisfy security requirements. With this information, after trade-off analysis, the designer can identify parts of the systems that should be removed, replaced, improved or kept.

Traceability among NFRs: throughout the whole design process of a system, higher level NFRs are refined and broken down into more concrete ones, particularly in a top-to-bottom approach. For example, a high level and abstract NFR such as security can be refined into access control or encryption requirements at lower levels. Therefore, in order to check satisfaction of security in the system, it is necessary to keep track of its refinements and lower level requirements that cover different aspects of security along with information on how much each contributes towards its parent NFR.

Satisfaction level of an NFR: it should be possible to evaluate the total satisfaction of an NFR in the system. This is necessary to compare current design against system specification and customer requirements as well as different design alternatives. In the end, the goal is that the designer gets an idea to what degree each NFR is satisfied.

Impact of an NFR on other NFRs: as mentioned before, NFRs cannot be considered in isolation in a system without taking into account their impacts on each other. Therefore, it is required to identify and evaluate effects that a model element and design decision that is used to satisfy one NFR, has on another NFR in the system. For example, performing heavy computations by an encryption component in an embedded system can also mean consuming more battery. Therefore, the side effects of such design decision should be identifiable for the designer.

Priority of an NFR: all NFRs do not have the same importance in the system. In order to increase overall satisfaction of NFRs and also to resolve conflicts among NFRs (reduce impact of one NFR in favor of another), it is necessary to know the importance of each NFR and compare them.

Coherent terms for NFRs: one subtle problem with NFRs that is more often noticed in large enterprises, is that different (sub)departments may have different interpretations for an NFR term or use different terms to refer to an NFR. Therefore, it is important to provide a coherent and consistent notation for defining NFRs and relating them to design elements.

Coherent measurements of NFRs: to compare different NFRs and perform trade-off analysis among them, evaluation of the satisfaction degree and impacts of NFRs should follow a coherent representation. In other words, the criteria or metrics used, should be such that to allow pair-wise comparison of NFRs (e.g. using the same types, scales and units, or a convertible format).

III. Suggested Profile

Figure 1 shows the structure of our profile to include the necessary information mentioned in the previous section in models to enable trade-off analysis of NFRs.

`System`: this stereotype is used to annotate the system which is the context of the analysis and to which different NFRs belong. It includes satisfactionValue tagged value to represent quantitatively the total satisfactions of the system's NFRs.

`NFR`: each NFR is identified using this stereotype. A higher level NFR (in terms of abstraction level) can be refined into one or more other NFRs. Therefore, there is an association relationship to itself (reflexive aggregation).

`Feature`: this stereotype which is an equivalent of *Operationalization* concept in NFR framework and Softgoal Interdependency Graph (SIG) [4], represents a feature in the system that contributes to the satisfaction of an NFR.

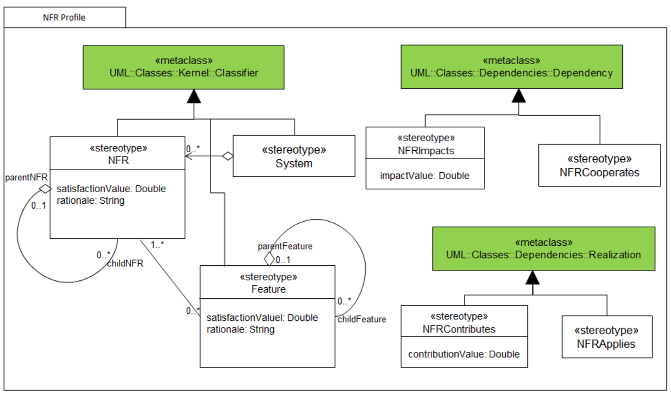
NFR Profile
«metaclass» UML::Classes::Kernel::Classifier
«metaclass» UML::Classes::Dependencies::Dependency
«stereotype» NFR
satisfactionValue: Double
rationale: String
parentNFR
0..1
0..*
childNFR
1..*
0..*
«stereotype» System
«stereotype» NFRImpacts
impactValue: Double
«stereotype» NFRCooperates
parentFeature
0..1
«stereotype» Feature
satisfactionValuel: Double
rationale: String
0..*
childFeature
«metaclass» UML::Classes::Dependencies::Realization
«stereotype» NFRContributes
contributionValue: Double
«stereotype» NFRApplies

Fig. 1. Profile for Trade-off analysis of NFRs

`NFRContributes`: this stereotype which is used on relationships between model elements shows that an element (NFR or Feature) contributes to the satisfaction of another element. The contributionValue of this stereotype is used to specify the degree of this contribution.

`NFRImpacts`: this is similar to NFRContributes stereotype, but is used to include the impact of a design decisions and model element on other NFRs in the system in a quantitative manner. ImpactValue attribute of this stereotype shows the degree of the impact.

`NFRCooperates`: is used to relate two or more elements that cooperate together to satisfy an NFR. This is similar to the AND relation in NFR framework and SIG.

`NFRApplies`: the relation between NFR related model elements and functional ones can be modeled using NFRApplies stereotype (e.g. an NFR that applies to a component).

`Rationale`: this property and tagged value in NFR and Feature stereotypes can be used to include the description and rationale for an NFR, its refinements into other NFRs or Features implementing that.

There are also several rules on the elements and their relationships described above:

- The allowed range of values is between -1 and 1. For example a negative value on the NFRImpacts relationship shows the negative impact of the source element on the target element.
- The satisfaction value for each leaf node is considered to be 1.
- The sum of contribution values of the links connecting children nodes (refinement/lower level elements) to their parent should be less or equal to 1 (maximum is 1).
- Contribution of a child node to its parent is calculated as the satisfactionValue of the child node multiplied by its contributionValue or impactValue.
- The satisfactionValue of a node is, therefore, calculated as the sum of the contributions from all of its children nodes.

IV. Usage Example

The profile concepts described in the previous section were implemented in MDT Papyrus [5]. In figure 2 an example usage of the profile is shown on parts of a mobile phone system. Two NFRs are defined. One for quality of the taken camera picture and the other one for the battery life. There are two features that contribute to the quality of taken photos: usage of a flash and type of the lens. Also two features are considered for battery life: adjustment of screen's brightness level and auto standby feature. The contribution of each feature to its parent NFR is annotated using NFRContributes stereotype and its contributionValue attribute. In this system, use of flash consumes lots of battery. The impact of Flash feature on battery life is therefore specified using NFRImpacts stereotype and its impactValue attribute which is -0.8.

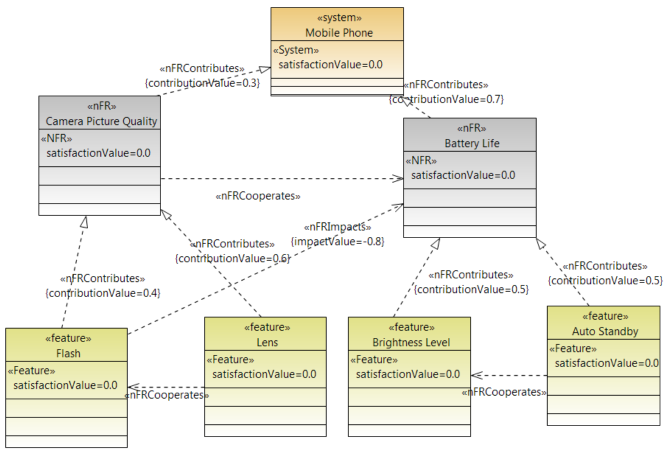
«system» Mobile Phone
«System» satisfactionValue=0.0
«nFRContributes» {contributionValue=0.3}
«nFRContributes» {contributionValue=0.7}
«nFR» Camera Picture Quality
«NFR» satisfactionValue=0.0
«nFR» Battery Life
«NFR» satisfactionValue=0.0
«nFRCooperates»
«nFRImpacts» {impactValue=-0.8}
«nFRContributes» {contributionValue=0.6}
«nFRContributes» {contributionValue=0.4}
«nFRContributes» {contributionValue=0.5}
«nFRContributes» {contributionValue=0.5}
«feature» Flash
«Feature» satisfactionValue=0.0
«feature» Lens
«Feature» satisfactionValue=0.0
«feature» Brightness Level
«Feature» satisfactionValue=0.0
«feature» Auto Standby
«Feature» satisfactionValue=0.0
«nFRCooperates»
«nFRCooperates»

Fig. 2. Example usage of NFR profile for a Mobile Phone

The contributionValue attributes for Battery Life and Camera Picture Quality NFRs to the Mobile Phone system actually imply the priorities the customer/designer has given to each. For example, Battery Life has 0.7 as its contribution value to the system while Camera Picture Quality has 0.3 which means Battery Life is more than two times more important for the customer. One point here is that, although these values are assigned subjectively by the designer, there are methods such as sensitivity analysis [6] that help to increase the confidence in the chosen decision. Now what we need to understand is the impact that having the Flash feature has on the system.

By having the necessary information in the model, it is now possible to perform analysis on the model to determine impacts of each design decision on the system and perform trade-off among them. To traverse the model and perform calculations, we have developed a model-to-model (M2M) transformation using QVT Operational language (QVT-O) [7] in Eclipse [8]. It reads as input a UML model annotated with our profile, traverses the nodes and calculates satisfaction values and writes results in the same model. In other words, we use an in-place transformation (i.e. input and output models are the same).

To calculate the satisfaction values, a recursive algorithm is used in the model transformation based on the rules mentioned in the previous section. For each node, all the incoming links that have NFRImpacts or NFRContributes stereotypes are retrieved. If a node does not have any such links (meaning that it is a leaf node), its satisfaction value is set to 1. Otherwise, the source of the link, which is another node, is retrieved and the algorithm starts calculating the satisfaction value of the source node; hence the recursion. The satisfaction value of each node is therefore calculated as follows: if s_i is the satisfaction value for each child node of a node, and l_i is the value on the link that connects the child node to its parent node, the satisfaction value of the parent node is calculated as: $\sum s_i * l_i$.

For example, in figure 2, the satisfaction values for `Auto Standby` and `Brightness Level` features are set to 1, since they are leaf nodes. The satisfaction value of `Battery Life` is then calculated as the satisfaction value of `Brightness Level` (1) multiplied by value on the link that connects it to `Battery Life` (0.5), plus the same multiplications for `Auto Standby` (1*0.5) and `Flash` (1*-0.8) which results in 0.20. The discrepancy observed between this calculated value (0.20) and the one in figure 3 (0.199...) which is calculated automatically through the in-place model transformation on the model is due to the OCL implementation of real numbers that QVT-O uses.

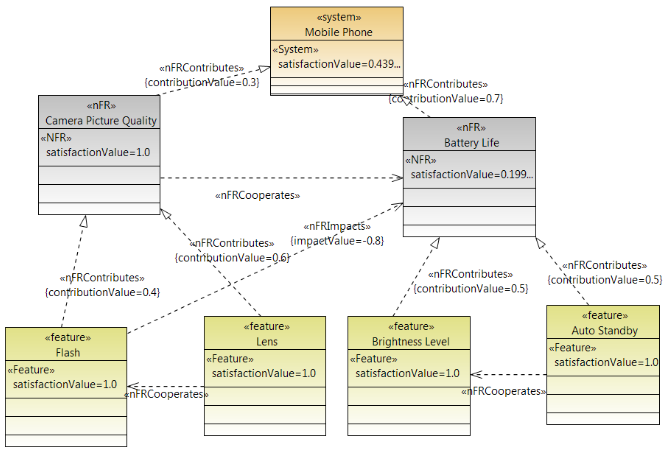
«system» Mobile Phone
«System» satisfactionValue=0.439...
«nFRContributes» {contributionValue=0.3}
«nFRContributes» {contributionValue=0.7}
«nFR» Camera Picture Quality
«NFR» satisfactionValue=1.0
«nFR» Battery Life
«NFR» satisfactionValue=0.199...
«nFRCooperates»
«nFRImpacts» {impactValue=-0.8}
«nFRContributes» {contributionValue=0.6}
«nFRContributes» {contributionValue=0.4}
«nFRContributes» {contributionValue=0.5}
«nFRContributes» {contributionValue=0.5}
«feature» Flash
«Feature» satisfactionValue=1.0
«feature» Lens
«Feature» satisfactionValue=1.0
«feature» Brightness Level
«Feature» satisfactionValue=1.0
«feature» Auto Standby
«Feature» satisfactionValue=1.0
«nFRCooperates»
«nFRCooperates»

Fig. 3. Calculated satisfcation value of the system having the Flash feature

By performing the transformation on the whole model, the satisfaction values are calculated for each node and propagated upwards toward the system element as shown in figure 3:

(1*0.4+1*0.6)*0.3+((1*0.5+1*0.5)-0.8)*0.7=0.44

Therefore, the total satisfaction value in this case will be 0.44. Trying the procedure again on the same model but without having the flash feature results in:

(1*0.6)*0.3+(1*0.5+1*0.5)*0.7=0.88

The introduced modeling concepts and calculations above, provide for several interesting features. One feature is to optimize the total satisfaction of the system considering different design alternatives. For example, in the mobile phone system described, if the designer needs to select among several possible solutions that contribute to better image quality, he/she can find the ones that lead to the highest satisfaction value of the system. Another possibility is to have run-time adaptability or re-configuration based on different quality of service levels. For example, if the battery level goes low beyond a certain limit, the system can go into a power-saving mode using features that incur minimum impact on battery consumption or replacing active components with back-up/standby ones which may provide lower quality/fewer services but consume less battery (e.g. in design diversity techniques [9]). Without the trade-off analysis introduced here, such a decision may not only be hard, but also will be blind in the sense that the side effects of a feature/component replacement on other aspects of the system will be unknown.

V. Related Work

There are versatile research works that try to target different issues regarding NFRs. [10] focuses on the problem of informal and separated documentation of design decisions and NFRs. To alleviate the problem, it introduces two profiles to model design decisions and generic NFRs to treat them as first-class entities in software architectures and maintain the traceability of design decisions and architectural elements. NFR framework proposed in [4] is one of the fundamental works in the area of NFR. It uses Softgoal Interdependancy Graph (SIG) to represent NFRs, their refinements and entities that NFRs are applied to (termed as *Operationalization*), and the interdependencies among them to include their impacts and relations. The dependencies and contributions of NFRs are specified using *make*, *hurt*, *help*, *break* and *undetermined* types. It provides notations to mark critical NFRs in the graph and also an evaluation procedure to determine satisfaction and conflicts of NFRs. The approach suggested in NFR framework is basically a qualitative approach. Moreover, the criticality concept in NFR framework seems more suitable for developers and does not convey enough information for prioritization of NFRs particularly from the customer's perspective and also for performing trade-off analysis [6]. [11] offers a UML profile for modeling SIG and concepts of NFR framework to represent NFRs as UML elements in order to integrate them with functional parts of the system (that are modeled in UML). The article [6] introduces Q-SIG which is a quantified version of SIG that enables quantitative evaluation of impacts and trades-offs among NFRs. In this paper, we introduced modeling concepts that enable designers to apply the Q-SIG approach in the form of UML models, and provided a tooling solution for evaluation and trade-off analysis of NFRS on these models using this approach.

VI. Summary and Future work

In this paper, a UML profile for modeling NFRs and their dependencies was introduced to enable performing trade-off analysis among them. As continuation of this work, we plan to develop an analysis tool as an Eclipse plug-in that can read as input, models annotated with our NFR trade-off profile and provide total satisfaction values for different NFRs, identify parts contributing negatively to an NFR, and perform calculations for overall optimization of NFRs in the system considering different design alternatives and scenarios. With the help of the tool, when some parts of the system need to be changed and updated, the user can identify the side effects of such changes on other parts and the system as a whole in terms of NFRs, at model level and before implementing the intended changes. The defined profile depicted in this paper is the first step towards enabling such features. Using the introduced modeling concepts here along with a back-annotation mechanism in model-based development of embedded systems will also be an interesting topic to further investigate and work on. Having such a mechanism, it would be possible to monitor the system for preservation of the specified NFRs and provide feedbacks to the design model about possible violations in the system (or any of its subsystems) in terms of satisfaction levels of their NFRs. Also, usage of the suggested approach for run-time adaptability and re-configuration of systems is another direction for further investigation.

References

[1] T. Henzinger and J. Sifakis, "The embedded systems design challenge," in *FM 2006: Formal Methods*, ser. Lecture Notes in Computer Science, J. Misra, T. Nipkow, and E. Sekerinski, Eds. Springer Berlin / Heidelberg, vol. 4085, pp. 1–15.

[2] J. Lee, K. Kapitanova, and S. H. Son, "The price of security in wireless sensor networks," *Comput. Netw.*, vol. 54, pp. 2967–2978, December 2010.

[3] B. Selic, "A systematic approach to domain-specific language design using uml," in *Object and Component-Oriented Real-Time Distributed Computing, 2007. ISORC '07. 10th IEEE International Symposium on*, may 2007, pp. 2 –9.

[4] L. Chung, B. A. Nixon, E. Yu, and J. Mylopoulos, *Non-Functional Requirements in Software Engineering*, ser. International Series in Software Engineering. Springer, October 1999, vol. 5.

[5] MDT Papyrus , http://www.eclipse.org/modeling/mdt/papyrus/, Accessed: February 2011.

[6] T. Marew, J.-S. Lee, and D.-H. Bae, "Tactics based approach for integrating non-functional requirements in object-oriented analysis and design," *The Journal of Systems and Software*, vol. 82, pp. 1642–1656, October 2009.

[7] QVT Operational Language, http://www.eclipse.org/m2m/, Accessed: February 2011.

[8] Eclipse Modeling Framework Project (EMF), http://www.eclipse.org/modeling/emf/, Accessed: February 2011.

[9] J. P. J. Kelly, T. I. McVittie, and W. I. Yamamoto, "Implementing design diversity to achieve fault tolerance," *IEEE Softw.*, vol. 8, pp. 61–71, July 1991.

[10] L. Zhu and I. Gorton, "Uml profiles for design decisions and non-functional requirements," in *Proceedings of the Second Workshop on SHAring and Reusing architectural Knowledge Architecture, Rationale, and Design Intent*, ser. SHARK-ADI '07. Washington, DC, USA: IEEE Computer Society, 2007, pp. 8–.

[11] S. Supakkul, "A uml profile for goal-oriented and use casedriven representation of nfrs and frs," in *In Proceedings of the 3rd International Conference on Software Engineering Research, Management and Applications*, 2005, pp. 112–121.