A Process Oriented Approach to Model Non-Functional Requirements
Proposition Extending UML

Aneesh Krishna
Department of Computing, Faculty of Science & Engineering
Curtin University of Technology, WA 6102, Perth, Australia
a.krishna@curtin.edu.au

Abstract
Non-functional requirements (NFRs), sometimes termed quality, or quality of service, attributes or requirements, have been a topic of interest within systems engineering, software engineering, and requirements engineering for a considerable period of time. NFRs have been for too long overlooked during the development of software systems. This has led to numerous cases of failure resulting in over budgeting or cancellation of projects. This paper proposes a method to extend the UML diagrams to model NFRs. A car rental case study is provided to illustrate the use of the proposed method.

1. Introduction
Functional Requirements (FRs) determine what functions a software product should offer. Multiple proposals have been developed to analyze and model them, and are nowadays widely used. For Non-Functional Requirements (NFRs), the problem is more complex and due to that they have been ignored for a long time in the early phases of the software development process. NFRs are defined as constrained on the emergent properties of the overall system. Hence, software products are sometimes refined late in the development process to satisfy NFRs, this as a result leads to over budget or cancelled projects [1]. Software engineers now acknowledge the importance of dealing with NFRs in the early phases of the development process [1]; however, there is lack of credible proposals to be used in the software development. In this paper we propose a solution to integrate the analysis and modelling of NFRs into UML [2], which is the most popular language for modelling purposes in object-oriented paradigm. Our goal is to offer a method to analyze, refine and integrate NFRs in existing diagrams to provide justifications for the design.
Chung et al [1] developed a framework to represent and analyze non-functional requirements. This framework is an interesting one; having potential to fully analyze the dependencies between non-functional requirements. This work uses the concept of ‘claim’ in the diagram. Claims are linked to interdependency, and provide a rationale to design decision. However, having multiple claims can make diagrams complicated and harder to read. Cysneiros et al [3] first proposed a method to integrate NFRs into ER and UML data models. This strategy uses the notion of lexicon, namely, the Language Extended Lexicon (LEL), as an anchor to integrate NFRs into OO models. The idea of LEL is to register the vocabulary (symbols) of a project, and determine the relations between the symbols. The concept of LEL is not widely accepted/used in software industry. Imposing its usage would hugely increase the workload of the systems analysts. Therefore, another solution is needed to link NFRs to UML diagrams that would have practical usefulness in industry. This work also considers that functional and non-functional aspects should be carried through two independent cycles with convergence points, while efforts should be carried out to merge the two aspects. Cysneiros’s proposal can be used to integrate NFRs into any UML diagram, but no support is provided to deal with diagrams other than Class Diagrams. These reasons form the basis for our proposal in this paper.

This paper is structured as follow: Section 2 formally formulates the proposal, providing new notations to deal with NFRs through an example. Finally, Section 3 concludes the paper.

2. Incorporation of NFRs into UML
The proposal considers that NFRs and FRs should not be carried out through two independent cycles, but should be part of a common, global strategy. This decision had a huge impact on the process designed. The process can be visualized in Figure 1.

The main steps of the proposal are the following:
1. First, we determine what the FR and NFRs are.
2. From the FRs, we shall draw the use cases, and then associate the NFRs with the different parts of the system they control. When associating them, two cases may arise: either the NFR is specific to an FR (e.g. ‘the authentication must be fast’), or it is global and thus not linked to a single FR (e.g. ‘all screens of the system should look similar’). In the first case, the NFR is linked with a use case, else with the entire system.
3. The NFRs will then be refined using the use cases. In this step, the systems analysts will use their experience to find missing NFRs that should be included
in the diagrams. This is quite important, because the stakeholders do not always think about every single NFR (some might seem obvious, other irrelevant at first), or some of them are simply hard to come up with. This should prevent NFR from emerging at later stages of the project.

4. Using the FRs and NFRs that have been extracted, a requirement graph will be drawn. This graph will show the decompositions of the NFRs into subgoals, how to satisfy the sub-goals with operationalizations, and the way all these elements interact with each other. The focus here is solidly on FR that can impact NFRs, so not every FR will be included in this graph.

5. Study the interdependencies between the different requirements, their subgoals and their operationalizations. If necessary, perform trade offs (for example, using uncompressed format is good for performance, but not for space usage), and more generally, refine the requirements graph by studying the impact of each interdependency on the related requirements (see subsection 3.4).

6. Finally, use all the information gathered to draw other UML diagrams. This paper highlights only the application of this to sequence, class and deployment diagrams.

**Use cases controlled by NFRs:** Incorporation of NFRs into use case diagrams is done by using a solution similar to [4]. A prerequisite for this is a complete or near complete list of the system requirements. FRs are subsequently used to produce use cases. Each NFR is documented with three pieces of information: the interested stakeholder, the NFR’s type and NFR identification code (ID). Representing an NFR in the diagram, these three pieces of information are particularly important. First, we need a non ambiguous way to link a use case to an NFR, which is done by stating the ID of the NFR. Then, including both the stakeholder and the type of the NFR in the representation provides the systems analysts’ with useful information without overloading the schema.

The general representation of such a diagram is presented on Figure 2.

**Construction of the NFRs graph:** The aim of this step is to decompose an NFR into sub-softgoals [1]. Then, sub-softgoals that can’t be subdivided find alternatives to operationalize it. Therefore, the following process has to be completed for each NFR represented in the use cases/control cases diagram. Add the object that the NFR is related to. Given that the NFR is not linked to a use case in this diagram, it is important to know on what the NFR acts (e.g. if we consider the informal high-level NFR ‘Good Performance for accounts’, the object is ‘Account’). Next, the analyst has to specify the priority of the NFR. This is important to algorithmically determine what subset of the NFRs can actually be satisfied. The mark 1 means that the NFR can easily be ignored, while 5 mean that the NFR has to be satisfied no matter what. With the conclusion of the process, the NFR is renamed ‘Formulated NFR’ as shown in Figure 3.

Sub-softgoals are subdivided until the analysts decide that no more decomposition is feasible or interesting. If possible, the analysts must try to avoid dividing a subgoal using only ‘helps’ contributions. If a softgoal does not have a sub-softgoal (or a group of sub-softgoals) which ‘satisfies’ it, it will be hard to determine if the higher level softgoal can actually be met. Once no more division is needed, possible operationalizations are associated to each leaf of the constructed graph. To do that, one must think about the most intelligent solutions to respond to the need expressed by a softgoal. All these alternatives will be represented on the graph using a unique ID and linked to the corresponding softgoal using the same connectors than before.

At the end of this phase, the analyst should have a graph...
of formulated NFR, each with its own set of sub softgoals and operationalizations. The next step is to study the interactions between graphs and to incorporate the necessary FR.

**Studying interdependencies in the requirements diagram:** At this point of the process the analysts concentrate on the interdependencies between the different requirements. First all the NFRs graphs which are related are grouped. Obviously, the NFRs controlling the entire system must appear on each diagram. For the others it is quite easy: the NFRs linked to the same use cases, or having the same 'object' (Figure 3) can interact or be in conflict. Once all the related NFRs are on the same diagram, one uses the use case diagram and the associated documentation to add all the related FRs to the diagram. FR can also have an ID.

What remains is the selection of the operationalizations which will be used to design the system. In this work, we don’t propose any algorithm (this forms part of the future investigation on this topic) to solve this problem, but such an algorithm should:

- analyze the impact of choosing each operationalization (which requirements are satisfied or helped, which operationalizations or requirements are hurt or broken)
- assure that every FR (and NFR of priority 5) is satisfied (one of its operationalization is chosen)
- give a mark to each alternative subset (using the weight of the contributions and the priority of the NFRs)
- return the highest-ranked solutions

The analyst then chooses the most appropriate subset for the design of the system. The chosen operationalizations are marked with the letter “C” in the ‘Expected status’ box, while the others with the letter “R”, “Rejected”. The ‘Expected status’ of the Formulated NFRs and softgoals will then be updated according to this choice.

**UML diagrams:** This section demonstrates how to integrate NFRs and their operationalizations into UML diagrams. The focus is on sequences, deployment and classes, but the techniques can work on other diagrams.

The following links can be used in these diagrams, as shown in Figure 4

- The «implements» link means that the element has been added to the design of the product in order to satisfy the operationalization. If the operationalization is not considered anymore, this element can be removed.
- The «controlled by» link means that the element previously existing in the diagram is used by the operationalization. It is important to highlight such an element: if it has to be modified to comply with its primary function, the systems analysts must study the impact of the change on the realization of the operationalization.
- The «controlled by» link defines that one knows that the implementation has to be guided by a particular NFR but no specific solution has been chosen yet. Most of the time, such a link must come with an annotation.

**Sequence Diagram:** Once the sequence diagrams are ready, the analysts will incorporate the NFRs into them. Usually, a sequence diagram will be related to one of the use cases that has been designed earlier, and this use case might be related to one or more NFRs. If this is the case, then for each of them, the analyst has to ask himself: where does this NFR impact the sequence diagram, and how? The analyst can choose the level of detail for the diagram as shown on Figure 5. Therefore the analyst can:

- state that a subsequence is «controlled by» an NFR,
- link some specific previously existing messages to a operationalization («contributes to»),
- add new messages (and maybe new classes) to the diagram which «implements» an operationalization.

**Class Diagram:** The approach to incorporate NFRs into class diagram is quite simple and is shown in Figure 6.
**Verification:** Once the NFRs or their operationalizations have been included in every necessary diagram, one has to verify that everything planned on the NFR graphs is actually considered in the design of the product. This justifies the existence of the box ‘Current status’ at the bottom of the representations of NFRs and operationalizations in the NFR graphs. For each operationalization chosen, one has to check if it appears on diagrams with links of type «implements» or «contributes to». If it is the case, then the operationalization has been ‘satisfied’, or else it was ‘not handle’. Then update the ‘current status’ of all NFR soft-goals that the operationalization links to. After considering the operationalization, if an NFR softgoal that was expected to be ‘satisfied’ or ‘weakly satisfied’ is not, it has to be verified if this NFR is linked to any other diagram with relation «controlled by». If so, the NFR has not been forgotten, however, no technical solution has been indicated on the design. If it is the case, it is up to the analyst to decide if this is good enough to mark to softgoal as ‘weakly satisfied’/’satisfied’ or if the design needs to be refined. These verifications should in fact be done automatically by a framework implementing our proposal. After these verifications, the operationalizations which were chosen but are not satisfied and the NFR softgoals where ‘expected status’ is different from ‘current status’ should be highlighted for the analyst to know immediately what is missing.

**Car Rental Case Study:** This case study demonstrates the application of the method for designing an e-commerce application for online car rentals. The example concentrates on core aspects of such a system with emphasis on business processes that relate to rental and provision of information to customers regarding the company. The complete explanations of the approach (along with case study) can be had by visiting http://www.computing.edu.au/~aneesh/NFR (details withheld due to space limitations).

Use cases enriched with NFRs are shown in Figure 7. The sequence diagram of the system is depicted in figure 8. The functionality depicted is limited to the searching process for a car.

3 **Conclusions and Future Work**

This paper presented a novel approach to dealing with Non-Functional Requirements and to model them in UML diagrams. This work constitutes an important contribution given that it allows system analysts to consider NFRs in their models without requiring additional learning of new software. Part of our future plans include the automation of elements of the proposal in order to utilise its full usefulness in industrial settings. We accept that the full proposal is very difficult to automate, as there is a need to provide manual support by systems analysts at certain steps of the approach. However, this is the case with almost all methods proposed so far in this challenging area.

---

**References**


