Improving the Separation of Non-Functional Concerns in Requirements Artifacts

Geórgia Sousa and Jaelson Castro
Centro de Informática – Universidade Federal de Pernambuco (UFPE)
P.O. Box 7851, CEP: 50.732-970, Recife – PE – Brazil
{gmcs, jbc}@cin.ufpe.br

Abstract

One of the most important principles in Software Engineering is the separation of concerns. When this principle is correctly applied, it helps to promote comprehensibility, maintainability and reusability of software system artifacts. However, often, the specification of non-functional requirements is scattered and tangled with functional artifacts they affect. Hence, in order to produce better requirements specifications, this paper presents an approach to improve the separation of concerns in requirements artifacts, providing a way to represent non-functional concerns apart from the requirements they affect and to specify the composition between them in a noninvasive way.

1. Introduction

The growing complexity of software and the demand for rapid development have increased the importance of comprehensibility, maintainability and reusability of software system artifacts. Separation of Concerns (SoC) is one of the Software Engineering principles often used to support these quality attributes [1].

Separation of concerns means dealing with different issues of a problem individually so that it is possible to concentrate on each one separately. The main advantages of applying this principle are: (i) decrease of the software development complexity by concentrating on different issues separately; (ii) reduction of efforts and separation of responsibilities [1] as well as (iii) improvement of the modularity of software systems artifacts. Furthermore, when the SoC principle is correctly applied, software artifacts tend to be cohesive and loosely coupled [2].

Nevertheless, due to the intrinsic relationship among some requirements, especially between non-functional and functional ones, sometimes references to a non-functional requirement (NFR) is scattered across multiple functional artifacts (scattering) and one functional artifact includes references to multiple non-functional requirements (tangling). For instance, each non-functional requirement is normally repeated in every use case that the NFR affects [3]. This kind of representation makes it difficult to evolve, maintain, reuse and understand the requirements [4].

Hence, in order to improve comprehensibility, maintainability and reusability of requirements specifications, this work describes an approach to provide a way to represent non-functional requirements apart from the requirements they apply to as well as means to specify the composition among them in a noninvasive way.

2. Proposal Outline

The purpose of our approach is to improve the description of non-functional requirements. It is worth noting that in most development approaches, requirements are specified in a scattered and tangled fashion.

Figure 1 presents an outline of our proposal. We consider separately both the analysis and the specification of functional and non-functional requirements. Furthermore, we document the relationships between them in a different artifact, named composition table. Doing so, we improve the comprehensibility, maintainability and reusability of requirements specifications.

![Figure 1 – Outline of our proposal](image)

We rely on the well known use case approach [5] to capture and represent functional requirements.

We chose the NFR Framework [6] to systematically deal with non-functional concerns. In this approach, abstract and subjective NFRs (e.g. security, performance)
are treated as goals to be achieved. In summary, our approach each non-functional goal is iteratively decomposed into more specific ones. At some point, when the non-functional goals has been sufficiently refined, it is possible to operationalize it, i.e. provide more concrete and precise mechanisms (e.g. operations, processes, data representations, constraints) to realize it.

However, there are some non-functional requirements that need to be related to functional requirements. The common approach to model this situation is to insert, inside each affected use case, references to the NFR operationalizations that that need to be applied to them [3]. For complex systems, this kind of representation makes it difficult to keep all the requirements updated and in the correct place. Moreover, scattering and tangling in requirements artifacts makes it difficult to evolve, maintain, reuse and understand the requirements [4]. Nevertheless, according to the separation of concerns principle, in order to preserve the reuse and maintainability of the artifacts, one artifact should only have references to artifacts that share the same concerns.

Hence, we advocate that this composition should be represented separately, i.e. in a noninvasive way in relation to the affected artifacts. To achieve this goal, we provide a composition table (see Table 1).

Table 1 - Composition Table for a Crosscutting Artfact

<table>
<thead>
<tr>
<th>Affected Artifact</th>
<th>Condition (optional)</th>
<th>Composition Rule Operator</th>
<th>Affected Point</th>
</tr>
</thead>
<tbody>
<tr>
<td>UC 4N&lt;name&gt;</td>
<td>condition of the composition</td>
<td>(overlap.after</td>
<td>overlap.before</td>
</tr>
</tbody>
</table>

This table should be specified for each NFR operationalization that is crosscutting, describing which artifacts it will affect (first column), in which point of the artifact the crosscutting should be applied (third column) and how the composition should be done (second column). To determine how a crosscutting requirement is applied in a particular point that it affects, we use the following composition rule operators [7]:

- **Overlap**: indicates that the crosscutting requirement should be applied before or after the step of the scenario it transverses;
- **Override**: indicates that the crosscutting requirement superposes the scenario’s step it transverses. This means that the behaviour described by the crosscutting requirement substitutes the behaviour defined by the step.
- **Wrap**: indicates that the crosscutting requirement “encapsulates” the scenario’s step it transverses. This means that the behaviour described by the scenario’s step affected is enveloped by the behaviour described by the crosscutting requirement.

3. Conclusion and Future Work

In software development it is important to specify system artifacts with a clear separation of concerns. However, sometimes it is difficult to apply the separation of concerns principle in the specification of requirements artifacts due to the strong relationship and the interdependencies among some requirements. This fact is especially true in the specification of non-functional requirements that are naturally crosscutting. Very often, they are scattered and tangled in functional specification.

Our proposal provides an approach to solve this problem, allowing that: (i) the analysis and specification of functional and non-requirements to be accomplished separately; and (ii) the composition of these requirements to be done in a different artifact.

Hence, crosscutting requirements are specified separately from the requirements they affect. As a consequence, we advocate that using a quite simple composition mechanism, the maintainability and the reusability of functional requirements specification can be improved. The benefits of our approach will be greater as the number of relationships between functional and non-functional requirements is increased.

Our future work will focus on applying this proposal in case studies and evaluating its use in the context of Aspect-Oriented Software Development.

3. References