Integrating the NFR framework in a RE model

Isabel Brito
Departamento de Engenharia,
Escola Superior de Tecnologia e Gestão
Instituto Politécnico de Beja,
7800-050 Beja, Portugal
+351 284311540
isabel.sofia@estig.ipbeja.pt

Ana Moreira
Departamento de Informática,
Faculdade de Ciências e Tecnologia,
Universidade Nova de Lisboa,
2829-516 Caparica, Portugal
+351 212948536
amm@di.fct.unl.pt

Abstract

In this paper we build on our work already produced on advanced separation of concerns for requirements engineering by adding two main ideas: (i) the integration of catalogues to help identifying and specifying concerns and (ii) improve the composition rules by informally defining some new operators.

1. Introduction

Separation of concerns refers to the ability of identifying, encapsulating and manipulating parts of software that are crucial to a particular purpose [5]. Separation of concerns reduces software complexity and enhances understandability and traceability throughout the development process. It minimizes the impact of change promoting evolution.

Non-functional concerns, such as response time, accuracy, security and reliability, are properties that affect a system as a whole. Existing approaches to deal with non-functional concerns have some limitations due to the diverse nature of these concerns. Moreover, most approaches handle non-functional concerns separately from the functional requirements of a system. This shows evidence that the integration is difficult to achieve and usually is accomplished only at the later stages of the software development process.

Another problem is that the current approaches fail in addressing the crosscutting nature of some concerns, i.e. it is difficult to represent clearly how these concerns can affect several requirements simultaneously. Since this is not supported from the requirements stage to the implementation stage, some of the software engineering principles, such as abstraction, localization and modularisation, can be compromised. Furthermore, the resulting system is more difficult to maintain and evolve.

This paper builds on our work already produced on advanced separation of concerns for requirements engineering. In [2], and improved version of [3], we have presented an approach for requirements engineering to identify, specify and integrate concerns, including crosscutting concerns. In those papers, we proposed the concepts of match point, dominant concern and composition rule. The novelty introduced here is, on the one hand, the use of catalogues, such as the NFR framework [4] to help identify and specify concerns and, on the other hand, a refinement of the composition rules by using some new operators inspired in the LOTOS operators [1].

The rest of this paper is organised as follows. Section 2 gives an overview of our model, and discussing its main activities. Section 3 applies the approach to a subway system. Finally, Section 4 draws some conclusions and points out directions for future work.

2. A model for separation of concerns at RE

A generic model to handle separation of concerns during requirements engineering is depicted in Figure 1. It is composed of four main tasks, as: identify concerns, specify concerns, identify crosscutting concerns and compose concerns.
Task 1: Identify concerns. A concern is any matter of interest in a system, i.e., it is a goal or a set of properties that the system must satisfy. The identification of concerns is accomplished by undertaking a complete and exhaustive analysis of the documentation and any other information provided by the system’s stakeholders. (Stakeholders are all the people that have a direct or indirect influence in the system under study. They may be users, clients, managers, developers, etc.)

We believe that requirements elicitation can be accomplished by using any of the existing techniques. However, so far we have only studied the use of viewpoints [8] and UML [7]. While some techniques emphasize the main functions that the future system should implement (e.g. use cases [10]), others emphasize constraints and certain properties that affect the whole system (e.g. the NFR framework [4]). To alleviate the onus of the identification of the broadly scoped requirements we propose the use of existing catalogues, such as the NFR framework [4].

Each concern should be registered in a template as depicted in Table 1.

| Classification | Helps the selection of the most appropriate approach to specify the concern. For example: functional, non-functional, goals. |
| Contribution     | Represents how a concern can be affected by other concern. This contribution can be positive (+) or negative (-) |
| Priority         | Expresses the importance of the concern for the stakeholders. It can take the values: Very Important, Important, Medium, Low and Very Low. |
| Required concerns | List of concerns needed by this concern. |

The outcome of Task 1 is the completion of the rows Name, Source, Stakeholders and Description. The remaining rows will be filled during Task 2.

Task 2: Specify concerns. This task is divided into four subtasks: apply the approach that better specifies each concern; identify contributions between concerns so that conflicts can be detected; identify priorities; the list of concerns needed by which concern to accomplish its behaviour. Besides completing the remaining unfilled rows in each template, these subtasks will produce auxiliary documentation generated by the chosen specification techniques (e.g. use cases, viewpoints, interdependency graphs).

Task 2.1: Specify concerns using the best approach. We may use any of the existing techniques for requirements specification. In certain cases a choice may have been done during Task 1, especially if particular techniques have been used to help the elicitation process. If so, we should build any models or other documentation proposed by those techniques. Based on this we can fill the Classification row in Table 1.

Task 2.2: Identify contributions between concerns. A contribution relationship between two concerns defines the way in which one concern affects the other. This contribution can be collaborative (or positive) and is 

Table 1: A template to describe concerns

| Name | The name of the concern. |
| Source | Source of information, e.g. stakeholders, documents, domain, catalogues and business process. |
| Stakeholders | People that need the concern in order to use the system. |
| Description | Explains the intended behaviour of the concern. |
concerns have the same priority, a trade-off must be negotiated with the user. To guide the negotiation among users we propose the identification of the dominant crosscutting concern for a given match point. Therefore, if two or more crosscutting concerns exist in a given match point, with negative contribution and the same priority, we start by analysing two concerns first to identify the dominant, then take the dominant and analyse it with a third concern and so forth until we have taken into consideration all the crosscutting concerns. The result is the concern with higher priority between them all. Next we need to identify the second dominant crosscutting concern among the remaining concerns, and so on until we have a dependency hierarchy between all the concerns.

The identification of the dominant concern is important to guide the composition rule.

**Task 4.4: Define composition rules.** The composition rule defines the order in which the concerns will be applied in a particular match point. It takes the form: `<concern> <operator> <concern>`. The operators we propose are inspired on the LOTOS specification language [1]. In the descriptions below C<sub>i</sub> denotes the <sup>i</sup>th Concern.

- Enabling (denoted by C<sub>i</sub>>C<sub>j</sub>): This is a sequential composition and means that the behaviour of C<sub>j</sub> begins if and only if C<sub>i</sub> terminates successfully.
- Disabling (denoted by C<sub>i</sub>|>C<sub>j</sub>): This is the disabling composition and means that C<sub>2</sub> interrupts the behaviour of C<sub>i</sub> when it starts its own behaviour. This allows the representation of interruptions.
- Pure interleaving (denoted by C<sub>i</sub>||C<sub>j</sub>): This is a parallel operator and means that C<sub>i</sub> evolves separately from C<sub>j</sub>. It represents concurrent composition without interaction.
- Full synchronization (denoted by C<sub>i</sub>|||C<sub>j</sub>): This is another parallel operator and means that the behaviour of C<sub>i</sub> must be synchronized with the behaviour of C<sub>j</sub>. It represents concurrent composition with interaction.

A composition rule can be defined based on simpler composition rules, separated by one of the above operators. We will use brackets, “(“ and “)”, to attribute priorities to the operators.

### 3. Applying the approach to a case study

The case study we chose is based on the Washington subway system:

“To use the subway, a client has to own a card that must have been credited with some amount of money. A card is bought and credited in special buying machines available in any subway station. A client uses this card in an entering machine to initiate her/his trip. When s/he reaches the destination, the card is used in an exit machine that debits it with an amount that depends on the distance travelled. If the card has not enough credits the gates will not open unless the client adds more money to the card. The client
can ask for a refund of the amount in the card by giving it back to a buying machine.”

**Task 1: Identify concerns.** Based on the requirements given above, we could think that the system has to offer the following concerns: BuyCard, LoadCard, RefundCard, EnterSubway and ExitSubway.

Different types of methods are used to specify functional requirements. Use case driven approaches describe “the ways in which a user uses a system” that is why use case diagram is often used for capturing functional requirements [10]. The concerns listed above are not too difficult to identify, if we think in terms of use cases, for example. Other concerns can be identified based on the NFR catalogue [4]. For each entry in the catalogue, we must decide whether it would be useful in our system or not. For example, (i) Response Time, as the system needs to react in a short amount of time to avoid delaying passengers; (ii) Accuracy, as only right amounts should be debited from, or credited to a card; (iii) Multi-access, so that several passengers can use the system concurrently; (iv) Availability, as the system and the machines must be available when the subway is open; (v) Security, as card information must be protected against illicit actions.

For each of these concerns we fill rows Name, Source, Stakeholders and Description in a template. Here we show only the templates that will help us illustrating concerns that are crosscutting and a situation of possible conflict. The concerns chosen are Enter Subway (Table 3), Response Time (Table 4) and Availability (Table 5).

**Task 2: Specify concerns.** Concerns are specified using the following three subtasks.

**Task 2.1: Specify concerns using the best approach.** Apply the techniques that best suite the specification of the concerns encountered. In Task 1 we have already made a choice when identifying concerns. Therefore, we should specify the first five concerns using use cases (through scenarios, for example) and the rest of the concerns using Softgoal Interdependency Graphs (SIG) [4]. SIG is a hierarchy graph of softgoals (i.e. non-functional concerns) that shows the interdependencies between them.

Let us considerer the simple situation where only the actor Client is handled. The corresponding use case diagram is illustrated in Figure 2, where the use cases EnterSubway, ExitSubway and LoadCard have been refined to factorize the common functionality ValidateCard.

**Figure 2.** The use case diagram of the Subway System

Nodes of a SIG are (non-functional) concerns, and are represented by clouds, and the lines represent decompositions (of a given concern into its sub-concerns). When all sub-concerns of a given concern are needed to achieve that concern, an AND relationship is defined with an arc connecting the lines (see Figure 3). (A situation where not all the sub-concerns are necessary to achieve the concern, an OR relationship, is defined with two arcs linking the decomposition lines.)

**Figure 3:** Security specification for EnterSubway

In this case Integrity and Availability are both needed to achieve Security, and therefore an AND relationship is required. Furthermore, Accuracy and Completeness are needed to achieve Integrity. Notice that a subject matter is added under the concern name. A subject matter is the topic addressed by the non-functional concern. This means that in Figure 3, Security is being handled for EnterSubway, Integrity, Completeness and Accuracy must be guaranteed.
for Card Data and Availability needs to be guaranteed for Machine. Additionally, Card Data Accuracy may be satisfied with a Validation Card Data procedure, i.e., a possible design solution to satisfy the concern. This is known as operationalization [4] and is represented with a thick dark cloud and an arrow with a positive sign. Other operationalizations can be proposed for the other concerns.

Task 2.2: Identify contributions between concerns. For each concern we must identify its contribution to other concerns and fill the Contribution row in the template. Based on the NFR catalogue, we could, for example, say that Availability has a collaborative (+) contribution with Multi-access and a damage (-) contribution with Response Time (See Table 5).

Task 2.3: Identify priorities. In this task we identify the importance of each concern to the system. This information is added to the Priority row.

Task 2.4: Identify required concerns. Lists all the other concerns required to accomplish the behaviour of the concern being specified. This information is added to the Required concerns row.

At the end of this task, the template of each concern is complete.

### Table 3: Enter Subway template

<table>
<thead>
<tr>
<th>Name</th>
<th>EnterSubway</th>
</tr>
</thead>
<tbody>
<tr>
<td>Source</td>
<td>Stakeholders, set of initial requirements, knowledge of the system</td>
</tr>
<tr>
<td>Stakeholders</td>
<td>Client, owner of the system</td>
</tr>
<tr>
<td>Description</td>
<td>To initiate the trip the client inserts his/her card in an entering machine. The system checks the card and registers an entrance.</td>
</tr>
<tr>
<td>Classification</td>
<td>Functional</td>
</tr>
<tr>
<td>Contribution</td>
<td>&lt;none&gt;</td>
</tr>
<tr>
<td>Priority</td>
<td>Very Important</td>
</tr>
<tr>
<td>Required concerns</td>
<td>Response Time, Accuracy, Security, Availability, ValidateCard</td>
</tr>
</tbody>
</table>

### Table 4: Response Time template

<table>
<thead>
<tr>
<th>Name</th>
<th>Response Time</th>
</tr>
</thead>
<tbody>
<tr>
<td>Source</td>
<td>Stakeholders, set of initial requirements, NFR catalogue, knowledge of the system</td>
</tr>
<tr>
<td>Stakeholders</td>
<td>Client, owner of the system</td>
</tr>
<tr>
<td>Description</td>
<td>The system has to react in time in specific situations, for instance to enter the subway, exit the subway.</td>
</tr>
<tr>
<td>Classification</td>
<td>Non-functional</td>
</tr>
<tr>
<td>Contribution</td>
<td>(-) Security, (-) Multi-access</td>
</tr>
<tr>
<td>Priority</td>
<td>Very Important</td>
</tr>
<tr>
<td>Required concerns</td>
<td>&lt;none&gt;</td>
</tr>
</tbody>
</table>

### Table 5: Availability template

<table>
<thead>
<tr>
<th>Name</th>
<th>Availability</th>
</tr>
</thead>
<tbody>
<tr>
<td>Source</td>
<td>Stakeholders, set of initial requirements, NFR catalogue</td>
</tr>
</tbody>
</table>

<table>
<thead>
<tr>
<th>Stakeholders</th>
<th>Client, owner of the system</th>
</tr>
</thead>
<tbody>
<tr>
<td>Description</td>
<td>All the machines of the system need to be accessible from 6am to 2am. For instance, to enter subway, exit subway, buy card, load card and refund card.</td>
</tr>
<tr>
<td>Classification</td>
<td>Non-functional</td>
</tr>
<tr>
<td>Contribution</td>
<td>(+) Multi-access, (-) Response Time</td>
</tr>
<tr>
<td>Priority</td>
<td>Very Important</td>
</tr>
<tr>
<td>Required concerns</td>
<td>&lt;none&gt;</td>
</tr>
</tbody>
</table>

Task 3: Identify crosscutting concerns. Looking at rows Required concerns and Description we can identify crosscutting concerns. For example, Response Time is crosscutting because it is required in EnterSubway and ExitSubway. ValidateCard, on the other hand, is also crosscutting, as it is required in EnterSubway, LoadCard, and ExitSubway (see Figure 2). Other crosscutting concerns are: Availability, Multi-access, Security and Accuracy.

Task 4: Compose concerns. The goal here is to compose all concerns to obtain the whole system. During this process conflicts may be identified.

Task 4.1: Identify match points. This is accomplished by building the match points table (see Table 6).

### Table 6: Identification of match points (AC: Accuracy; MA: Multi-access; RT: Response Time; S: Security; AV: Availability, V:ValidateCard)

<table>
<thead>
<tr>
<th>Concern</th>
<th>EnterSubway</th>
<th>BuyCard</th>
<th>ValidateCard</th>
<th>...</th>
</tr>
</thead>
<tbody>
<tr>
<td><strong>Client</strong></td>
<td>RT, S, AC, AV, V (MP_A)</td>
<td>S, AC, AV (MP_B)</td>
<td>AC</td>
<td></td>
</tr>
</tbody>
</table>

Task 4.2: Identify conflicts. When the same cell lists more than one crosscutting concern we must verify the contribution between them. For example, in our case study the MP_A match point has five concerns. As we identified in Task 2.2, Availability and Response Time, for example, contribute negatively to each other, i.e., they constrain each other’s behaviour, and they may generate a conflict.

Task 4.3: Identify dominant concern. To solve the possible conflicts identified above, we need to verify the priority of each concern involved. Let us only deal with Availability and Response Time. As both have the same priority (Very Important) we need to negotiate a trade-off with the system’s stakeholders. Supposing that the priority of Response Time is decreased, Availability becomes the dominant concern in MP_A. However, as Availability is a sub-concern of Security, we need to make clear if Security is also dominant with respect to Response Time. What makes sense in this situation, and because Security is composed of two sub-concerns (Availability and Integrity), is to guarantee Availability first and only at the end guarantee Integrity. On the other hand, Response Time...
concern is needed during, i.e. it wraps, Accuracy and EnterSubway concerns.

Task 4.4: Define composition rules. A composition rule for MP could be defined using the operators described above as follows:

```
( Security.Availability
  >> ( (ValidateCard >> EnterSubway)
    ||
    ResponseTime
    ||
    Security.Integrity.Accuracy
  )
) >> Security.Integrity
```

> ErrorHandling

Noticed that this expression should not be looked at as a formal LOTOS behavioural expression, since at this level of abstraction we cannot really talk about the “behaviour” of a concern. Our main goal is basically to give the idea of the order in which the concerns should be satisfied. With this in mind, we could then informally say that after the successfully satisfaction of Availability, Response Time, EnterSubway and Accuracy must synchronize and be satisfied in parallel. Notice that EnterSubway can only be satisfied after the successful satisfaction of ValidateCard. Only after this will Integrity be satisfied. If something goes wrong during the composition process, the system must be able to handle the errors raised. Here we represent this situation by defining the concern ErrorHandling that has the capability to interrupt any of the concerns involved by the disable operator. (Notice that we are using the “.” notation to represent a sub-concern of a given concern, e.g. Security.Integrity.)

4. Conclusions and future work

This paper presents a model to handle advanced separation of concerns during requirements engineering. It extends the work presented in [2] and [3] in two ways. First, it proposes the use of catalogues to help in the identification and specification of concerns. The catalogue we have mainly used is the NFR framework. Second, it explores a refinement of the composition rules, by using a set of operators based on the LOTOS major operators.

There are still several problems we need to address in our approach. The major problem we need to investigate is related to the composition rules. LOTOS is a formal specification language with a well-defined semantics. In this paper we started exploring the idea of LOTOS-like operators to specify composition rules. However, at the moment we cannot rigorously talk about a concern’s behaviour, in order to then be able to write a valid behavioural expression. To solve this problem we need to (i) work on the decomposition of concerns in terms of responsibilities in order to be able to define composition rules at a finer level of granularity and (ii) study the operationalizations of non-functional concerns so that only non-functional concerns that are mapped onto functions or aspects are used in the composition rules. There will be others that will map onto a design decision, for example, and that will not be considered in a LOTOS-like behavioural expression [8]. The decomposition of concerns will help us study the level of granularity at which a conflicting situation can be handled. Another problem we need to further study is the ErrorHandling concern. We believe it is a necessary, but we know that it will be very difficult to specify.

We are also planning to define a visual integrated notation. Finally, we would like to investigate the possibility of extending a LOTOS tool, Light, for example, to support the validation of the composition rules.

5. References


