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.

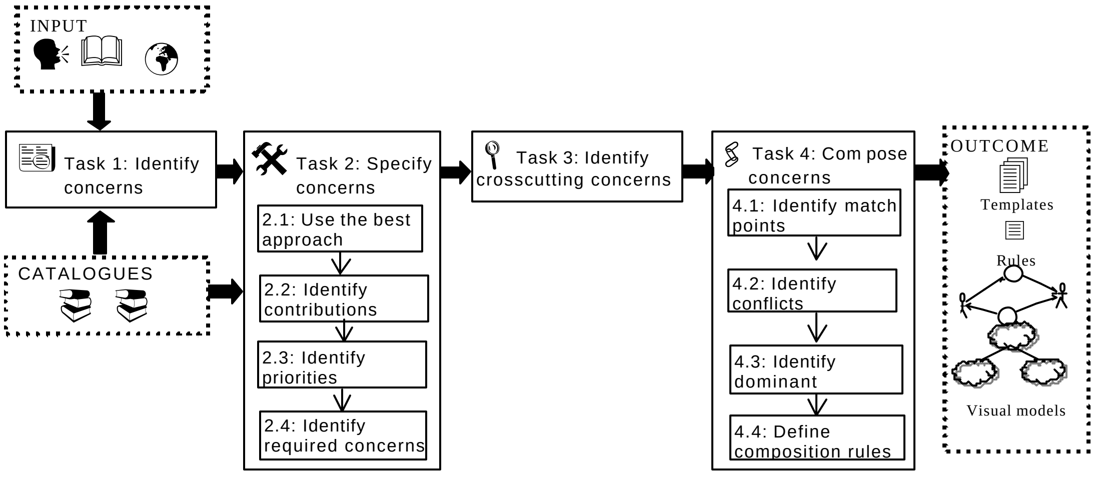
INPUT
Task 1: Identify concerns
CATALOGUES
Task 2: Specify concerns
2.1: Use the best approach
2.2: Identify contributions
2.3: Identify priorities
2.4: Identify required concerns
Task 3: Identify crosscutting concerns
Task 4: Com pose concerns
4.1: Identify match points
4.2: Identify conflicts
4.3: Identify dominant
4.4: Define composition rules
OUTCOME
Templates
Rules
Visual models

Figure 1: A model for separation of concerns at RE

Despite the direction of the thick black arrows, indicating forward development, an iterative and incremental process should guide the model.

Task1: 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' 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.

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.
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 represented by "+", or damage (or negative) and is represented by "-". This information is added to the *Contribution* row in Table 1. It is based on this information that we will be able to detect conflicts between concerns.

Task 2.3: Identify priorities. For each concern we need to identify its degree of importance in the system. This is done with the stakeholders' help. This information is added to row *Priority* in Table 1.

Task 2.4: Identify required concerns. This provides the list of concerns that the concern under study concern needs to accomplish its role. If this concern does not require any other concern the keyword <*none*> is used.

Task 3: **Identify crosscutting concerns**. A concern is crosscutting if it is -required by more than one other concern. This task is accomplished by taking into account the information in rows *Required concerns* and *Descriptions* of the template describing that concern.

Task 4: Compose concerns. The goal of this task is to compose all the concerns so that the developer can get a grasp of the whole system. In this paper we will focus our attention to the composition of those concerns that are crosscutting, as the non-crosscutting ones bring no new problems. To guide the composition, we propose four subtasks: identify match points, identify conflicts, identify the dominant concern, and define the composition rules.

Task 4.1: Identify match points. Based on row *Required concerns* in Table 1 we identify the points where the composition will take place, i.e. the match points. A match point tells us which crosscutting concerns should be composed with a given concern. This can be better illustrated with a bi-dimensional table of Stakeholders X Concerns (see Table 2). Each cell is filled with the list of crosscutting concerns (denoted CC_i) that affect a given concern and represents a match point (denoted MP_i). One composition rule must be defined for each match point.

Table 2: Match points identification

Concerns / Stakeholders	Concern$_1$	…	Concern$_m$
Stakeholder$_1$	CC_1, CC_2 **(MP_A)**		CC_2, CC_5 **(MP_J)**
Stakeholder $_2$	CC_1, CC_3 **(MP_B)**		
…	…		
Stakeholder $_n$			CC_3, CC_5 **(MP_Z)**

Task 4.2: Identify conflicts. This subtask supports the identification of conflicting situations between concerns. For a given match point we need to analyse if any of the involved crosscutting concerns contribute negatively to any other (based on the *Contribution* row in Table 1). Concerns contributing positively to other concerns raise no problems.

Task 4.3: Identify dominant concern. This subtask helps solving the conflicts identified in the previous subtask and is based on *Priority* row of the template. If the priority attributed to each concern is different, the problem is not too difficult to solve, and the dominant concern is that with higher priority. However, if at least two crosscutting 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_i denotes i th Concern:

- Enabling (denoted by $C_1 >> C_2$): This is a sequential composition and means that the behaviour of C_2 begins if and only if C_1 terminates successfully.
- Disabling (denoted by $C_1 [> C_2$): This is the disabling composition and means that C2 interrupts the behaviour of C_1 when it starts its own behaviour. This allows the representation of interruptions.
- Pure interleaving (denoted by $C_1 ||| C_2$): This is a parallel operator and means that C_1 evolves separately from C_2. It represents concurrent composition without interaction.
- Full synchronization (denoted by $C_1 || C_2$): This is another parallel operator and means that the behaviour of C_1 must be synchronized with the behaviour of C_2. 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 uses 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).

Task2: 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.

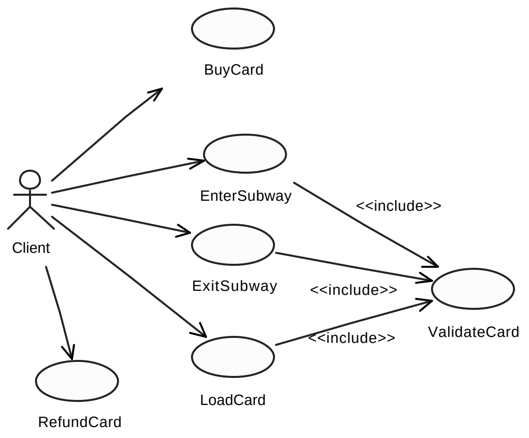

Figure 2. The use case diagram of the Subway System

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.)

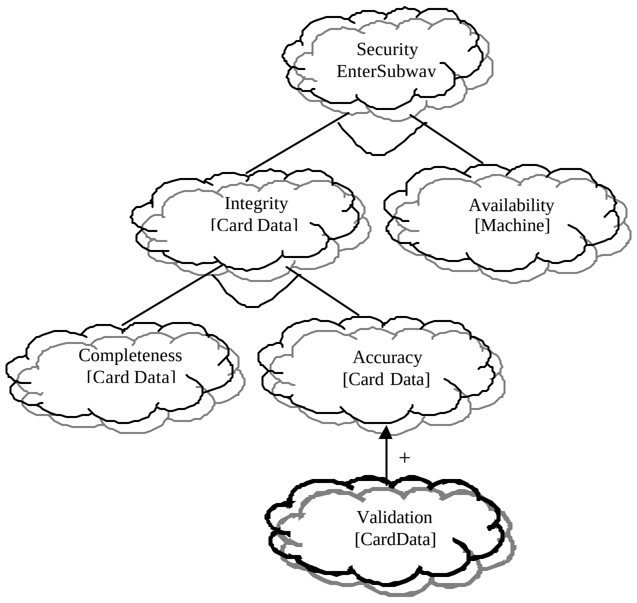

Figure 3: Security specification for EnterSubway

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

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

Table 4: Response Time template

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

Table 5: Availability template.

Name	Availability
Source	Stakeholders, set of initial requirements, NFR catalogue
Stakeholders	Client, owner of the system
Description	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.
Classification	Non-functional
Contribution	(+) Multi-access, (-) Response Time
Priority	Very Important
Required concerns	<none>

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)

Concern / **Stakeholder**	EnterSubway	BuyCard	ValidateCard	…
Client	RT, S, AC, AV, V **(MP$_A$)**	S, AC, AV **(MP$_B$)**	AC	

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_A 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

[1] Brinksma E. (ed): *Information Processing Systems -- Open Systems Interconnection-- LOTOS -- A Formal Description Technique Based on the Temporal Ordering of Observational Behaviour*, ISO 8807, 1988.

[2] Brito, I. and Moreira A. "Advanced Separation of Concerns for Requirements Engineering". VIII Jornadas de Ingeniería del Software e Base de Datos, JISBD 2003. Alicante, España. 2003.

[3] Brito, I. and Moreira A. "Towards a Composition Process for Aspect-Oriented Requirements". Early Aspects Workshop at AOSD Conference. Boston, USA. 2003.

[4] Chung, L., Nixon, B., Yu, E. and Mylopoulos, J. Non-Functional Requirements in Software Engineering, Kluwer Academic Publishers, 2000.

[5] Dijkstra, E.W., *A Discipline of Programming*, Prentice-Hall, 1976

[6] IBM Research, MDSOC: Software Engineering Using Hyperspaces, http://www.research.ibm.com/hyperspace/

[7]Moreira, A., Araújo, J., Brito, I.: Crosscutting Quality Attributes for Requirements Engineering, 14th International Conference on Software Engineering and Knowledge Engineering (SEKE 2002), ACM Press, Italy, July 2002.

[8] Rashid, A., Sawyer, P., Moreira, A. and Araújo, J. "Early Aspects: a Model for Aspect-Oriented Requirements Engineering", IEEE Joint Conference on Requirements Engineering, Essen, Germany, September 2002.

[9] Workshop on Multi-Dimensional Separation of Concerns, International Conference on Software Engineering, ICSE 2000, http://www.research.ibm.com/hyperspace/workshops/icse2000

[10] Schneider, G. and Winters, J., Applying Use Cases – A Practical Guide, Addison-Wesley, 1998.