

A Concern-Oriented Requirements Engineering Model

Ana Moreira[†], João Araújo[†], and Awais Rashid[‡]

[†] CITI/Dept. Informática, FCT, Universidade Nova de Lisboa,
2829-516 Caparica, Portugal

[‡] Computing Department, Lancaster University,
Lancaster LA1 4YR, UK
{amm, ja}@di.fct.unl.pt, awais@comp.lancs.ac.uk

Abstract. Traditional requirements engineering approaches suffer from the tyranny of the dominant decomposition, with functional requirements serving as the base decomposition and non-functional requirements cutting across them. In this paper, we propose a model that decomposes requirements in a uniform fashion regardless of their functional or non-functional nature. This makes it possible to project any particular set of requirements on a range of other requirements, hence supporting a multi-dimensional separation. The projections are achieved through composition rules employing informal, often concern-specific, actions and operators. The approach supports establishment of early trade-offs among crosscutting and overlapping requirements. This, in turn, facilitates negotiation and decision-making among stakeholders.

1 Introduction

The tyranny of the dominant decomposition [21] refers to the limited mechanisms used by traditional methods to decompose complex systems into separate concerns. Modern approaches propose mechanisms for decomposition and composition. However, they mostly use a dominant dimension as the base decomposition, with other possible dimensions cutting across them. For example, approaches, such as the NFR framework [2], use non-functional requirements as the dominant dimension with the functional dimension added a posteriori. Other existing requirements engineering (RE) approaches, such as viewpoints [7, 19] and use cases [9], use functional requirements as the dominant decomposition with analysis conducted against a set of non-functional requirements cutting across the base.

It has been argued that crosscutting is a phenomenon that is not limited to non-functional requirements and that functional requirements can also often cut across parts of a system [16]. Existing separation of concerns mechanisms at the RE level do not explicitly account for such crosscutting nature of functional requirements. Consequently, they cannot be handled effectively leading to a lack of identification and characterisation of their influence on other concerns in the system. Furthermore, an initially non-crosscutting set of requirements (functional or non-functional) might become crosscutting in future. The two-dimensional nature of existing decomposition approaches does not provide support to deal with such unanticipated evolution.

In this paper, we propose a model that decomposes requirements in a uniform fashion regardless of their functional or non-functional nature. This makes it possible to *project* any particular set of requirements on a range of other requirements, hence supporting a multi-dimensional separation. A projection specifies the influence of a given concern on other concerns and is achieved through composition rules employing informal, often concern-specific, actions and operators. The rules specify the projection of a particular concern onto other concerns it relates to. The various projections make it possible for us to compose a range of *reflected projections* contributing to an individual concern. The approach supports establishment of early trade-offs among crosscutting and overlapping requirements. This, in turn, facilitates negotiation and decision-making among stakeholders. The uniform nature of the decomposition also makes it possible to deal with situations where an initially non-crosscutting set of requirements evolves to have a wider influence in the system.

Section 2 discusses existing approaches to separate crosscutting concerns at the RE level and highlights how these suffer from the tyranny of dominant decomposition. Section 3 presents our model for multi-dimensional separation of requirements level concerns. Section 4 provides an overview of the realisation of the model using XML and applies it to a case study of a location and context sensitive tourist guide. Section 5 discusses some related work, while Section 6 concludes the paper and identifies directions for future work.

2 Background

Separation of concerns has been contemplated by well-known RE approaches such as goal-oriented techniques and viewpoints. In goal-oriented approaches [12], such as KAOS [3] and i^* [23], a goal is an objective that the system under consideration should achieve. It can be formulated at different levels of abstraction and covers concerns in two dimensions, i.e., functional and non-functional. KAOS uses a formal language (first-order temporal logic with real-time constraints) to specify critical parts of the system, besides allowing informal modelling. Goals are used to detect and manage conflicts among requirements. The i^* framework identifies and models organisational requirements and adopts the goal and softgoal modelling concepts as its dimensions. A softgoal represents a non-functional requirement we expect to satisfy within acceptable limits.

Separation of crosscutting properties has also been considered in PREView [19], a viewpoint-oriented requirements engineering method. A PREView viewpoint encapsulates partial information about the system. Requirements are organised in terms of several viewpoints, and analysis is conducted against a set of concerns intended to correspond broadly to the overall system goals. In applications of the method, the concerns that are identified are typically high-level non-functional requirements. Here again the separation of concerns is two-dimensional: one being the viewpoints that handle functional requirements and the PREView-specific notion of concerns which encapsulate non-functional properties.

The Aspect-Oriented Requirements Engineering (AORE) model presented in [17] is based on treating PREView concerns as adaptations of the aspect-oriented programming [6] notion of aspects and, consequently, carries out the analysis of

broadly scoped properties against a base set of viewpoints. A refinement of this model, presented in [16], supports separation of the specification of aspectual requirements, non-aspectual requirements and composition rules in modules representing coherent abstractions and following well-defined templates. This modularisation makes it possible to establish early trade-offs between aspectual requirements hence providing support for negotiation and subsequent decision-making among stakeholders. However, the composition rules have to be written with reference to a dominant decomposition that aspects cut across.

The discussion above demonstrates that, while existing RE approaches support analyses of system requirements from the perspective of non-functional properties, support for identifying the influence of crosscutting functional properties (or a combination of functional and non-functional properties) is not available. Nor is there any support for incorporating such an influence during trade-off analysis and subsequent negotiation among stakeholders. The multi-dimensional approach presented in this paper addresses the above issues by eliminating the dominant decomposition through uniform treatment of the various types of requirements in the system. In deriving our multi-dimensional model we have built on the strengths of the model in [16], mainly the informal composition rules with concern-specific actions and operators and the effective support for establishing trade-offs and negotiations among stakeholders.

3 A Concern-Oriented Model for RE

Modern systems have to run in highly volatile environments where the business rules change rapidly. Therefore, systems must be easy to adapt and evolve. In order to facilitate adaptability and evolution, it is essential that the influence of any set of requirements on the system can be determined. In existing RE approaches, such analyses focus on the influence of non-functional requirements. Functional requirements that might have a wide impact on other functional or non-functional requirements are not effectively dealt with. In Section 2, we argued that this is a direct consequence of having a largely two-dimensional decomposition.

Our proposed model addresses this problem by treating all concerns in a uniform fashion. Concerns in our model imply any coherent collection of requirements. We do not classify concerns into viewpoints, use cases or aspects though our concerns still encapsulate coherent sets of functional and non-functional requirements. As shown in Figure 1, we perceive the concern space at the requirements level as a hypercube. Each face of the hypercube represents a particular concern of interest. By treating all concerns as equal we can choose any set of concerns as a base to project the influence of another concern or set of concerns onto this base. This flexible, multi-dimensional view makes it possible to handle both crosscutting functional and non-functional requirements in an effective fashion.

Our RE model is shown in Figure 2. We start by identifying and specifying concerns. Concern identification is carried out using a synthesis of existing requirements elicitation mechanisms such as viewpoints [7], use cases [9] and goals [12]. The identified concerns are specified using well-defined templates (cf. Section 4.1.1).

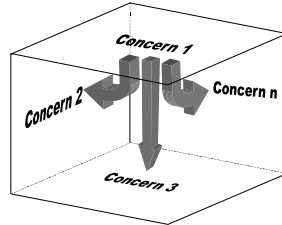


Fig. 1. Concern space represented as a hypercube (the block arrows represent projections)

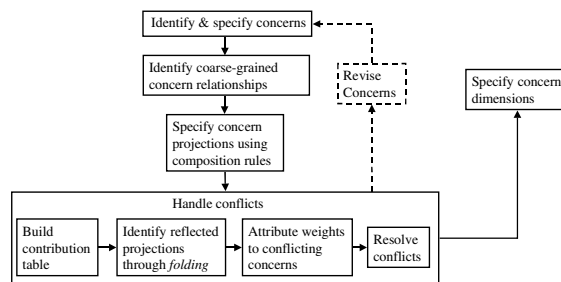


Fig. 2. RE model based on uniform treatment of concerns

The next step is to identify coarse-grained relationships among concerns by relating concerns to each other through a matrix. These relationships are identified using techniques such as domain analysis [10], ethnography [22] and natural language processing [18]. Looking at the matrix (cf. Table 1) we can see which concerns influence other concerns and whether any reciprocal influence exists.

Notice that in this paper, we do not focus on the exact kind of relationships between two concerns. In [3] interested readers can find a model for requirements interdependencies and inter-relationships.

Once the coarse-grained relationships between concerns have been established, the next step is to specify the possible projections of each concern on other concerns. This is achieved through composition rules. These rules operate at the granularity of individual requirements and not just the concerns encapsulating them. Consequently, it is possible to specify how a requirement in the concern in question influences or constrains the behaviour of a set of requirements in various other concerns. At the same time, if desired, trade-offs among concerns can be observed at a finer granularity. This alleviates the need for unnecessary negotiations among stakeholders for cases where there might be an apparent trade-off between two (or more) concerns but, in fact, different, isolated requirements are being influenced by them. It also facilitates identification of individual, conflicting requirements with respect to which negotiations must be carried out and trade-offs established.

After specifying the various projections with the aid of composition rules, identification and resolution of conflicts among the concerns is carried out. This is accomplished by:

1. Building a contribution matrix (cf. Table 2) where each concern may contribute negatively (-) or positively (+) to the others (empty cells represent “don’t care” contributions). The diagonal is marked with the concern names to support observation of reflected projections in step 2. This matrix is inspired on the NFR framework [2].
2. *Folding* the table along its diagonal (cf. Figure 3) to obtain the cumulative effect for situations where two concerns directly influence each other. An example of this folding is shown for C_1 and C_{n-1} in Figure 3. The folded table provides us the reflected projections: the combined influence of a set of concerns on a particular concern.
3. Attributing weights to those concerns that contribute negatively to each other in relation to a particular concern. Each weight is a real number in the interval $[0 .. 1]$ and represents the priority of a concern in relation to the concern it is projected on.
4. Solving the conflicts with the stakeholders, using the above prioritisation approach to help negotiation and decision-making.

Table 1. Relating concerns to each other

	C_1	C_2	...	C_n
C_1		√		
C_2				√
...				
C_n		√		

Table 2. Contributions between concerns

	C_1	C_2	...	C_n
C_1		+		
C_2				-
...				
C_n		-		

Conflict resolution might lead to a revision of the requirements specification (concerns and/or composition rules). If this happens, then the projections are revised and any further conflicts arising are resolved. The cycle is repeated until all conflicts have been resolved through effective negotiations.

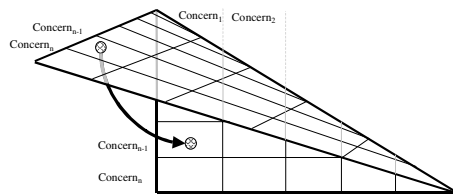


Fig. 3. The concern contribution table folded along its diagonal

The last activity in the model is identification of the dimensions of a concern. As observed in [17], concerns at this early stage can have an impact on artefacts at later development stages that can be described in terms of two dimensions:

- *Mapping*: a concern might map onto a system feature/function (e.g., a simple method, object or component), decision (e.g., a decision for architecture choice) and design (and hence implementation) aspect (e.g., mobility cf. Section 4.1.1).

Note that despite their crosscutting nature at this stage, some concerns might not directly map onto an aspect at later stages.

- *Influence*: a concern might influence different points in a development cycle, e.g., availability influences the system architecture while mobility influences specification, architecture, design and implementation.

4 Realisation of the Model

We have employed the eXtensible Markup Language, XML, as the definition language to specify the concerns and the composition rules to relate them with each other. The concerns and composition rules are specified using pre-defined templates. These templates can, optionally, be enforced using XML schemas. XML has been chosen because, as demonstrated by the following case study, there is a need for concern-specific actions and composition operators when defining the composition rules. The extensible model offered by XML coupled with the rich specification model of the XML schema language makes it an ideal choice as it is virtually impossible to anticipate the various types of composition operators and actions that might be required. Since the XML schema language is extensible – it is based on XML itself – it is possible to enforce constraints on the specification of composition rules when new operators and/or actions are introduced. Furthermore, the ability to define semantically meaningful tags and informal operators ensures that the readability of the requirements specification is not compromised as the specification resides in the stakeholders' domain and must be readable by them.

The use of XML makes it possible to select any projections of interest by using XPath queries and observe their cumulative effect. The selected projections and their effect can also be visualised using the eXtensible Stylesheet Language (XSL). This aids scalability in the presence of a large number of concerns.

4.1 Case Study

The case study we have chosen is a location and context sensitive tourist guide system inspired by a real system implemented at Lancaster [5]:

“The system provides an electronic hand-held guide that offers the following facilities to the visitors: (1) retrieve information about the city, including information about their current location; (2) provide route guidance to help visitors move between locations on the tour; (3) enter a set of preferences and interests to generate suitable tours of the city; (4) access external services, such as hotel and theatre ticket reservations.”

4.1.1 Identify and Specify Concerns

There are some concerns that are probably easier to identify as they directly represent stakeholders views on the basic functionality of the system. For example, we definitely have a concern that reflects the visitor needs and another for tourist information centre. Other concerns reflect more global properties of the system. For example, mobility will be a concern, as the system needs to react while the visitor is moving around. This brings immediately to our minds the context concern as the system needs to recognise the visitor's change in location. This, in turn, suggests portability as another one, since the visitor needs to carry with her/himself the

electronic device to access the system while on the move. Other concerns such as compatibility and availability are two obvious ones, since the system must be compatible with other external services (hotel and ticket reservation systems) and available anytime the visitor is using it. The concerns we identified are as follows:

- *Visitor*: users that can retrieve information from the system, including their current location.
- *Tourist Information Centre*: decides which information goes into the system.
- *Electronic Device*: used by the visitors to access the system.
- *Portability*: the electronic devices to access the system must be carried around by the visitors and therefore must be portable.
- *Mobility*: the system must handle mobility as the visitor will need to access the system on the move during his/her tour.
- *Context*: the system must recognise and handle the visitor's change in location.
- *Compatibility*: the system must be compatible with the external services it has to interact with, in particular, hotel and theatre ticket reservations.
- *Availability*: the system must always be available to react to stimuli (e.g., be accessed by the visitor) and for data updates.

We will be using the concerns *Visitor*, *Mobility* and *Compatibility* to illustrate our approach. Figures 4 through 6 show these concerns specified in XML.

```
<?xml version="1.0" ?>
- <Concern name="Visitor">
  - <Requirement id="1">
    The visitor will be able to retrieve information from the system.
    <Requirement id="1.1">
      The visitor will be able to access information about the attractions.
    </Requirement>
    - <Requirement id="1.2">
      The visitor will be able to access information about his/her location.
      <Requirement id="1.2.1">
        The visitor will be able to validate the information about the location if it does not correspond to what s/he sees.
      </Requirement>
    </Requirement>
    - <Requirement id="1.3">
      The visitor will be able to obtain a list of available preset tours.
    </Requirement>
  - <Requirement id="2">
    The visitor will be able to create a custom tour.
    <Requirement id="2.1">
      The visitor will be able to specify preferences.
    </Requirement>
  </Requirement>
- <Requirement id="3">
  The visitor will be able to follow a tour.
  <Requirement id="3.1">
    The visitor will be able to reconfigure the tour.
  </Requirement>
</Requirement>
```

```
- <Requirement id="4">
  The visitor will be able to access external services.
  <Requirement id="4.1">
    The visitor will be able to access hotel reservation.
  </Requirement>
  <Requirement id="4.2">
    The visitor will be able to access theatre ticket reservation.
  </Requirement>
</Requirement>
</Concern>
```

Fig. 4. The *Visitor* concern in XML

```
<?xml version="1.0" ?>
- <Concern name="Mobility">
  - <Requirement id="1">
    The system will be accessed on the move.
    <Requirement id="1.1">
      The system will be accessed from within a limited area.
    </Requirement>
  </Requirement>
</Concern>
```

Fig. 5. The *Mobility* concern in XML

```
<?xml version="1.0" ?>
- <Concern name="Compatibility">
  <Requirement id="1">
    The system must be able to interact with external services.
  </Requirement>
</Concern>
```

Fig. 6. The *Compatibility* concern in XML

The structure is self-explanatory: a Concern tag denotes the start of a concern while a Requirement tag denotes the start of a requirement. Refinements such as sub-requirements are represented via the nesting of the tags. Each requirement has an id which is unique within its defining scope i.e. the concern. Concern names are unique within the case study. However, XML namespaces can be used for the purpose as well.

4.1.2 Identify Coarse-Grained Concern Relationships

As we identify and describe concerns we can relate them, by building the matrix in Table 3. The tick indicates a unidirectional relationship, from left to right, between two concerns. For example, *Tourist Information Centre* has an impact on *Visitor*, as it is responsible to make available the information visitors can access. Between *Visitor* and *Mobility*, on the other hand, we can identify two unidirectional relationships (as they are semantically different cf. composition rules in Section 4.1.3): one from *Visitor* to *Mobility* indicating that visitors require mobility; and another from *Mobility* to *Visitor* indicating that mobility has to support information access for visitor.

4.1.3 Specify Concern Projections Using Composition Rules

Having studied the impact of each concern on all the others we can now start by analysing in more detail each relationship. The fundamental idea is that we can *project* each concern on all the others with which the first has a relationship. The projection specifies the influence of a given concern (represented in a row in Table 3) on other concerns (represented in columns in Table 3). Whenever a concern affects several other concerns it has a broadly scoped impact on the system and, therefore, can be classified as a crosscutting concern. As we can see from Table 3, not only non-functional concerns such as *Mobility* are crosscutting but also functional concerns such as *Visitor* have a similar nature.

Table 3. Matrix relating concerns (**Vis:** Visitor; **TIC:** Tourist information centre; **Port:** Portability; **Mob:** Mobility; **ED:** Electronic device; **Cont:** Context; **Comp:** Compatibility; **Avail:** Availability)

Concerns Concerns	Vis	TIC	Port	Mob	ED	Cont	Comp	Avail
Vis				✓	✓			
TIC	✓							
Port				✓	✓			
Mob	✓		✓			✓		✓
ED								
Cont								
Comp	✓	✓		✓				
Avail	✓	✓		✓	✓			

The materialisation of these projections is accomplished here by defining a set of composition rules, one for each projection. Composition rules define the relationships between concerns requirements at a fine granularity (unlike the relationship matrix in Section 4.1.2 which is aimed at identifying coarse-grained relationships). Composition rule definitions can be governed by an XML schema. However, for simplification we describe the structure of composition rules with reference to some examples and not the XML schema definition. As shown in Figures 7 through 9, a

coherent set of composition rules is encapsulated in a Composition tag. Figure 7 encapsulates all compositions (i.e. projections) for the *Visitor* requirements while Figures 8 and 9 do so for *Mobility* requirements and *Compatibility* requirements respectively. The semantics of the Requirement tag here differ from the tags in the concern definition. If a concern requirement has any sub-requirements these must be explicitly excluded or included in the Constraint imposed by a concern requirement. This is done by providing an “include” or “exclude” value to the optional children attribute. A value of “all” for a concern or id value implies that all the requirements within the specified concern are to be constrained.

The Constraint tag defines an, often concern-specific, action and operator defining how the concern requirements are to be constrained by another concern requirement. Although the actions and operators are informal, they have clearly defined meaning and semantics to ensure valid composition of concerns. This provides the architects and designers a systematic means to interpret the requirements specification. The Outcome tag defines the result of constraining the concern requirements with another concern requirement. The action value describes whether another concern requirement or a set of concern requirements must be satisfied or merely the constraint specified has to be fulfilled (see Table 6).

```
<?xml version="1.0" ?>
<Composition>
  <Requirement concern="Visitor" id="all">
    <Constraint action="ensure"
      operator="during">
      <Requirement concern="Mobility" id="1"
        children="include" />
    </Constraint>
    <Outcome action="fulfilled" />
  </Requirement>
  <Requirement concern="Visitor" id="all">
    <Constraint action="provide" operator="by">
      <Requirement concern="ElectronicDevice"
        id="all" />
    </Constraint>
    <Outcome action="fulfilled" />
  </Requirement>
</Composition>
```

Fig. 7. Composition rule for *Visitor*

```
<?xml version="1.0" ?>
<Composition>
  <Requirement concern="Mobility" id="1"
    children="include">
    <Constraint action="provide" operator="for">
      <Requirement concern="Visitor" id="all" />
    </Constraint>
    <Outcome action="fulfilled" />
  </Requirement>
  <Requirement concern="Mobility" id="1"
    children="exclude">
    <Constraint action="enforce" operator="for">
      <Requirement concern="Portability" id="1" />
    </Constraint>
    <Outcome action="fulfilled" />
  </Requirement>
  <Requirement concern="Mobility" id="1"
    children="include">
```

```
  <Constraint action="affect" operator="on">
    <Requirement concern="Context" id="1"
      children="include" />
  </Constraint>
  <Outcome action="fulfilled" />
</Requirement>
<Requirement concern="Mobility" id="1.1">
  <Constraint action="affect" operator="on">
    <Requirement concern="Availability" id="all"/>
  </Constraint>
  <Outcome action="fulfilled" />
</Requirement>
</Composition>
```

Fig. 8. Composition rule for *Mobility*

```
<?xml version="1.0" ?>
<Composition>
  <Requirement concern="Compatibility" id="1">
    <Constraint action="ensure" operator="with">
      <Requirement concern="Visitor" id="4"
        children="include" />
    </Constraint>
    <Outcome action="satisfied">
      <Requirement concern="Mobility" id="1"
        children="include" />
    </Outcome>
  </Requirement>
  <Requirement concern="Compatibility" id="1">
    <Constraint action="ensure" operator="with">
      <Requirement
        concern="TouristInformationCent
          re" id="2" />
    </Constraint>
    <Outcome action="fulfilled" />
  </Requirement>
</Composition>
```

Fig. 9. Composition rule for *Compatibility*

The informality of the actions and operators ensures that the composition specification is still readable by the stakeholders, an important consideration during

requirements engineering. For example, if we look at the first composition rule in Figure 7 and focus on the values in bold we get the following: “**All Visitor** requirements must be **ensured during** requirement **1** of **Mobility**, including its children, with the outcome that the Visitor’s requirements are **fulfilled**”.

Tables 4 to 6 describe the semantics of the actions and operators, which we have defined so far, for Constraint and Outcome. The initial set of these actions and operators was first defined in [16]. Here, we have validated that proposal and identified one new action: *affect*.

Table 4. Description of *Constraint* actions

Constraint Action	
Type	Description
enforce	Used to impose an additional condition over a set of concern requirements.
ensure	Used to assert that a condition that should exist for a set of concern requirements actually exists.
provide	Used to specify additional features to be incorporated for a set of concern requirements.
applied	Used to describe rules that apply to a set of concern requirements and might alter their outcome.
exclude	Used to exclude some concerns or requirements if the value <i>all</i> is specified.
affect	Used to specify that a set of concern requirements will alter the state of another concern.

Table 5. Description of *Constraint* operators

Constraint Operator	
Type	Description
during	Describes the temporal interval during which a set of requirements is being satisfied.
between	Describes the temporal interval falling between the satisfaction of two requirements. The interval starts when the first requirement is satisfied and ends when the second one is to start being satisfied.
on	Describes the temporal point after a set of requirements has been satisfied.
for	Describes that additional features will complement the concern requirements.
with	Describes that a condition will hold for two sets of requirements with respect to each other.
in	Describes that a condition will hold for a set of requirements that has been satisfied.
AND,OR, XOR	Conjunction, disjunction and exclusive-OR (when either requirement is satisfied but not both)

The interesting point to note here is that not all operators are concern-specific, e.g. XOR is a generic operator. Also, the actions for the Outcome are generic and not specific to a particular concern. It is, however, not possible to say whether Outcome actions are always generic, as more case studies need to be carried out before arriving at such a conclusion. It is also worth noting that although the same operator might apply to different concern requirements, not all operator-action combinations are valid in the Constraint specification for a particular concern. More case studies need to be carried out to validate the set of operator-action combinations.

Table 6. Description of *Outcome* actions

Outcome Action	
Type	Description
satisfied	Used to assert that a set of viewpoint requirements will be satisfied after the constraints of a concern requirement have been applied.
fulfilled	Used to assert that the constraints of a concern requirement have been successfully imposed.

4.1.4 Handle Conflicts

The composition rules leads to the identification of conflicts among concerns whose requirements constrain the same or overlapping sets of other concern requirements. In case of our approach this process is optimised as any potential interaction or conflict can be deduced from the composition rules. Conflict resolution is carried out in the four steps described below.

Build the Contribution Table. The contribution table (cf. Table 7) shows in which way (negatively or positively) a concern contributes to the others. Each cell shows a unidirectional contribution between a concern located in a line and a concern located in a column. In this case, *Availability* contributes positively to *Visitor* and *Tourist Information Centre* and negatively to *Mobility* and *Electronic Device*.

Table 7. Contribution table

Concerns \ Concerns	Vis	TIC	Port	Mob	ED	Cont	Comp	Avail
Vis				+	+			
TIC	+							
Port				+	+			
Mob	+		+			+		-
ED								
Cont								
Comp	+	+		-	-			
Avail	+	+		-	-			

Identify Reflected Projections Through Folding. Having studied the contributions between concerns we can now fold the table in order to reduce the range of projections we have to deal with. During folding, concerns that have a symmetric projection on each other have their effects accumulated. This is shown in Table 8. The columns in the table show the reflected projections of various concerns on an individual concern.

Table 8 shows that *Compatibility* and *Availability* contribute negatively to *Mobility*. We can help resolve such conflict by attributing weights to the concerns involved in the conflicting situation.

Table 8. Folded table w/ reflected projections

Vis	Vis							
TIC	+	TIC						
Port			Port					
Mob	+		+	Mob				
ED	+		+		ED			
Cont				+		Cont		
Comp	+	+		-			Comp	
Avail	+	+		-	-			Avail

Attribute Weights to Conflicting Concerns. Weighting allows us to describe the extent to which a concern may constrain another. The values are given according to the importance each concern has with respect to another one. The scales we are using are based on ideas from fuzzy logic and have the following meaning:

- *Very important* takes values in the interval] 0,8 .. 1,0]
- *Important* takes values in the interval] 0,5 .. 0,8]
- *Average* takes values in the interval] 0,3 .. 0,5]
- *Not so important* takes values in the interval] 0,1 .. 0,3]
- *Do not care much* takes values in the interval [0 .. 0,1]

Using fuzzy values (very important, important, not so important, etc.) facilitates the stakeholders' task of attributing priorities to conflicting concerns.

Weights will be given to concerns with respect to the concern for which we have specified a composition rule. For example, with respect to *Mobility*, *Compatibility* can have a weight of 0.5, since accessing external services is not a fundamental issue in our system. In turn, *Availability* is very important for *Mobility* (a weight of 1.0), as without the system being available, we cannot offer mobility.

Table 9. Weighted (folded) contribution table

<i>Vis</i>	<i>Vis</i>								
<i>TIC</i>	+	<i>TIC</i>							
<i>Port</i>			<i>Port</i>						
<i>Mob</i>	+		+	<i>Mob</i>					
<i>ED</i>	+		+		<i>ED</i>				
<i>Cont</i>					+	<i>Cont</i>			
<i>Comp</i>	+	+			0,5			<i>Comp</i>	
<i>Avail</i>	+	+			1,0	1,0			<i>Avail</i>

Resolve Conflicts. The conflicts mentioned above for *Mobility* should not be too difficult to resolve, as the weights express priorities. If this was not the case negotiation would be needed among the stakeholders. Once all the conflicts have been resolved the specification is revised and recomposition carried out to identify any further conflicts.

4.1.5 Specify Concern Dimensions

Specification of a concern's dimensions makes it possible to determine its influence on later development stages and identify its mapping onto a function, a decision or an aspect. The various concerns in our case study and their mappings and influences are shown in Table 10.

Consider our Compatibility concern. The requirements derived from this concern will influence parts of the system specification, architecture and design pertaining to requirements derived from other concerns constrained by it. They will also influence system evolution as change of the external services must be anticipated. The Compatibility concern will, however, map on to a function allowing visitors to connect to both hotel and ticket reservations. The Mobility concern, on the other

Table 10. Concern dimension specification

Concern	Influence	Mapping
Visitor	Specification, design, evolution	Function
Tourist inf. centre	Specification, design, evolution	Function
Portability	Specification, architecture, design, implementation	Decision
Mobility	Specification, architecture, design, implementation	Aspect
Electronic device	Specification, architecture, design	Function
Context	Architecture, design, implementation	Aspect
Compatibility	Specification, architecture, design, evolution	Function
Availability	Architecture	Decision

hand, will influence the specification, the type of architecture chosen and the design of the classes realising the requirements constrained by Mobility. It will map to an aspect at the design and implementation level because mobility properties cannot be encapsulated in a single class and will be otherwise spread across a number of classes.

5 Related Work

Multi-dimensional separation of concerns is supported by Hyperspaces [21] and Cosmos [20]. The Hyperspaces approach employs hyperslices as a decomposition mechanism where concerns are organised according to multiple dimensions, where each dimension is partitioned by concerns of the same type (e.g. classes, functions). A hypermodule is a set of hyperslices together with a composition rule that specifies how the hyperslices are composed to form a more complex hyperslice. Our model can be seen as a specific instantiation of the hyperspaces model at the requirements level. Concerns in our model can be perceived as hyperslices while composition rules defining the projections can be seen as a specific instance of hypermodules. Cosmos is a concern-space modelling schema. Here a concern is any matter of interest in a system. A concern-space is an organised representation of concerns and their relationships. Similar to our work, Cosmos generalises the idea of a concern hyperspace (or hyperslice). It models concern-spaces through concerns, relationships and predicates. Concerns are classified as logical (representing concepts) and physical (representing elements of software systems). Some of the concerns and relationships e.g. physical ones are not relevant at the requirements level. Moreover, the projections of concerns on other concerns are not truly achieved.

Grundy proposes an aspect-oriented requirements engineering method targeted at component based software development [8]. The approach provides a categorisation of diverse aspects of a system that each component provides to end users or other components. A UML compliant approach to handle quality attributes (i.e. non-functional requirements) at the early stages of the development process is proposed in [14]. In both of these approaches, the separation of concerns is two-dimensional (i.e., functional and non-functional concerns (or aspects)). Moreover, the projections are limited from aspects to functional requirements.

In the Architecture Trade-off Analysis Method (ATAM) [11] various competing quality attributes and their interactions are characterised. This is achieved by building and maintaining both quantitative and qualitative models of these attributes. The

models are used as a basis to evaluate and evolve the architecture. The main focus of ATAM is on identifying the trade-off points at the architecture level. The work described in this paper focuses on identifying conflicting concerns in a uniform fashion and establishing critical trade-offs before the architecture is derived. Consequently, it is closer to the Twin Peaks model [15] which focuses on developing requirements and architectures in parallel in order to develop an early understanding of the system's technical feasibility, discover further requirements and constraints and evaluate alternative design solutions.

Theme/Doc [1] provides support for aspect-oriented analysis. Analysis is carried out by first identifying a set of actions in the requirements list which are, in turn, used to identify crosscutting behaviours. A theme is a collection of structures and behaviours that represent one feature. It is related to a concern in the work described here. While Theme/Doc is useful to identify themes in a requirements document, our approach complements this work by considering not only the identification of concerns, but also their specification and composition.

6 Conclusions and Future Work

In this paper, we have proposed a model to support multi-dimensional separation of concerns at the requirements level. This multi-dimensionality is achieved through a uniform treatment of both functional and non-functional properties. This is in direct contrast with existing RE approaches which typically focus on identifying the effects of non-functional requirements with reference to the functional requirements. Consequently, any broadly scoped influence of functional properties is not effectively dealt with.

The uniform treatment of concerns in our model makes it possible for us to define the projections of each concern on any set of concerns it relates to. By folding the resulting contribution matrix, we obtain a set of reflected projections which are then used for analysing the contribution of multiple concerns towards one particular concern. It is a difficult and complex task to derive such reflected projections otherwise.

The trade-off analysis and stakeholder negotiation supported by our RE model is based on a simple yet natural separation of concerns. This offers a powerful mechanism to identify influences of the various concerns in the system in a multi-dimensional fashion. This, in turn, supports better understanding of both crosscutting functional and non-functional requirements. Also, if a previously non-crosscutting set of requirements evolves to have a wider impact, the approach can easily deal with such a change through revision and recomposition of its projections. Any changes to relationships among concerns can be identified through requirements level impact analysis techniques [13].

Our future work will focus on developing case studies to further validate the proposed model and our set of concern specific actions and operators. In the near future we aim to incorporate the approach in our concern composition and decision support tool ARCADE which already provides support for the composition rules used for our case study. We are also interested in exploring the use of fuzzy logic for trade-

off analysis based on the weights we may give to concerns. This could help us identify a process to rank concerns by degree of importance in a system and use the result as a basis for incremental development.

Acknowledgements

This work is supported by EPSRC Grant MULDRE (EP/C003330/1) and Portuguese FCT Grant SOFTAS (POSI/EIA/60189/2004).

References

- [1] E. Baniassad and S. Clarke, "Theme: An approach for aspect-oriented analysis and design". In 26th International Conference on Software Engineering (ICSE), (Edinburgh, Scotland), 2004.
- [2] L. Chung, B. Nixon, E. Yu, and J. Mylopoulos, *Non-Functional Requirements in Software Engineering*: Kluwer, 2000.
- [3] Å. Dahlstedt and A. Persson, "Requirements Interdependencies - Moulding the State of Research into a Research Agenda". The Ninth International Workshop on Requirements Engineering: Foundation for Software Quality (REFSQ 2003), Klagenfurt/Velden, Austria, pp 71-80, 2003
- [4] A. Dardenne, A. Lamsweerde, and S. Fickas, "Goal-directed Requirements Acquisition", *Science of Computer Programming*, 20, pp. 3-50, 1993.
- [5] N. Davies, K. Cheverst, K. Mitchell, and A. Efrat, "Using and Determining Location in a Context-Sensitive Tour Guide", *IEEE Computer*, 34(8), pp. 35-41, 2001.
- [6] T. Elrad, R. Filman, and A. Bader (eds), "Theme Section on Aspect-Oriented Programming", *CACM*, 44(10), 2001.
- [7] A. Finkelstein and I. Sommerville, "The Viewpoints FAQ." *BCS/IEEE Software Engineering Journal*, 11(1), 1996.
- [8] J. Grundy, "Aspect-Oriented Requirements Engineering for Component-based Software Systems", 4th IEEE International Symposium on Requirements Engineering, 1999, IEEE Computer Society Press, pp. 84-91.
- [9] I. Jacobson, *Object-Oriented Software Engineering - a Use Case Driven Approach*: Addison-Wesley, 1992.
- [10] K. C. Kang, S. G. Cohen, J. A. Hess, W. E. Novak, and A. S. Peterson, "Feature-Oriented Domain Analysis (FODA) Feasibility Study", Software Engineering Institute Technical Report CMU/SEI-90-TR-21 1990.
- [11] R. Kazman, M. Klein, M. Barbacci, T. Longstaff, H. Lipson, and J. Carriere, "The Architecture Tradeoff Analysis Method", Proc. ICECCS, 1998, IEEE Computer Society Press, pp. 68-78.
- [12] A. Lamsweerde, "Goal-Oriented Requirements Engineering: A Guided Tour", 5th International Symposium on Requirements Engineering, 2001, IEEE Computer Society Press, pp. 249-261.
- [13] S. Lock and G. Kotonya, "An Integrated, Probabilistic Framework for Requirement Change Impact Analysis", *Australian Journal of Information Systems*, 6(2), 1999.
- [14] A. Moreira, J. Araújo, and I. Brito, "Crosscutting Quality Attributes for Requirements Engineering", In 14th International conference on Software Engineering and Knowledge Engineering (SEKE), 2002, ACM, pp. 167-174.

- [15] B. Nuseibeh, "Weaving Together Requirements and Architectures", *IEEE Computer*, 34(3), pp. 115-117, 2001.
- [16] A. Rashid, A. Moreira, and J. Araújo, "Modularisation and Composition of Aspectual Requirements", In International Conference on Aspect-Oriented Software Development (AOSD), 2003, ACM, pp. 11-20.
- [17] A. Rashid, P. Sawyer, A. Moreira, and J. Araújo, "Early Aspects: A Model for Aspect-Oriented Requirements Engineering", In International Conference on Requirements Engineering (RE), 2002, IEEE Computer Society Press, pp. 199-202.
- [18] P. Rayson, L. Emmet, R. Garside, and P. Sawyer, "The REVERE Project: Experiments with the application of probabilistic NLP to Systems Engineering", Proc. NLDB 2000, LNCS 1959, pp. 288-300.
- [19] I. Sommerville and P. Sawyer, *Requirements Engineering - A Good Practice Guide*: John Wiley and Sons, 1997.
- [20] S. M. Sutton and I. Rouvellou, "Modeling of Software Concerns in Cosmos", In International Conference on Aspect-Oriented Software Development (AOSD), 2002, ACM, pp. 127-133.
- [21] P. L. Tarr, H. Ossher, W. H. Harrison, and S. M. Sutton, "N Degrees of Separation: Multi-Dimensional Separation of Concerns", In International Conference on Software Engineering (ICSE), 1999, ACM, pp. 107-119.
- [22] S. Viller and I. Sommerville, "Social Analysis in the Requirements Engineering Process: From Ethnography to Method", In International Conference on Requirements Engineering (RE), 1998, IEEE Computer Society, pp. 6-13.
- [23] E. Yu, "Modelling Strategic Relationships for Process Reengineering": PhD Thesis, University of Toronto, 1995.