Multi-Dimensional Separation of Concerns in Requirements Engineering

Ana Moreira[†], Awais Rashid[‡], João Araújo[†]
[†]Dept. Informática, FCT, Universidade Nova de Lisboa, 2829-516 Caparica, Portugal
[‡]Computing Department, Infolab21, Lancaster University, Lancaster LA1 4WA, UK
{amm | ja} @di.fct.unl.pt; awais@comp.lancs.ac.uk

Abstract

Existing requirements engineering approaches manage broadly scoped requirements and constraints in a fashion that is largely two-dimensional, where functional requirements serve as the base decomposition with non-functional requirements cutting across them. Therefore, crosscutting functional requirements are not effectively handled. This in turn leads to architecture trade-offs being mainly guided by the non-functional requirements, so that the system quality attributes can be satisfied.

In this paper, we propose a uniform treatment of concerns at the requirements engineering level, regardless of their functional, non-functional or crosscutting nature. Our approach is based on the observation that concerns in a system are, in fact, a subset, and concrete realisations, of abstract concerns in a meta concern space. One can delineate requirements according to these abstract concerns to derive more system-specific, concrete concerns. We introduce the notion of a compositional intersection, which allows us to choose appropriate sets of concerns in our multi-dimensional separation as a basis to observe trade-offs among other concerns. This provides a rigorous analysis of requirements-level trade-offs as well as important insights into various architectural choices available to satisfy a particular functional or non-functional concern.

1. Introduction

An effective requirements engineering (RE) approach must reconcile the need to achieve separation of concerns with the need to satisfy broadly scoped requirements and constraints. Viewpoints [6], use cases [9] and goals [12] offer means of partitioning requirements as a set of partial specifications that aid traceability and consistency management. However, ensuring the consistency of these partial specifications with global requirements and constraints is largely unsupported.

Some RE approaches have explicitly focused on separation of broadly scoped properties, e.g., [1, 12, 15, 17, 18, 22]. However, such separation is largely two-dimensional. Functional requirements, separated using mechanisms such as viewpoints, use cases or themes, serve as the base decomposition with analysis conducted against a set of non-functional requirements or overall system goals or behaviour that cut 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, this leads to architecture quality analyses being dominantly driven by non-functional requirements and their associated trade-offs. The influence of functional requirements and their associated trade-offs on architectural quality is largely ignored.

In this paper, we propose that requirements should be decomposed 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 key characteristics of our approach include:

- the definition of a meta concern space from where concrete system-specific concerns can be derived based on the specific features of the problem domain;
- the notion of a *compositional intersection* which makes it possible to identify suitable sets of concerns in our multi-dimensional separation as a basis to observe trade-offs among other concerns;

- use our early trade-off analyses to provide insights into the architecture choices suited to each concern hence, facilitating alignment of the derived architecture with the requirements' intentions.

Section 2 discusses existing approaches to separate crosscutting concerns at the RE level and highlights how these suffer from the tyranny of dominant decomposition [20]. Section 3 introduces our approach for multi-dimensional separation of requirements level concerns and a context sensitive tourist guide used to demonstrate our approach. Section 4 discusses the notion of the meta concern space and its use in the delineation of system requirements into concrete concerns. Section 5 focuses on specification of composition rules for each concern as well as compositional intersections to identify potential trade-off points for subsequent analysis. Section 6 provides insights into the initial architecture choices, resulting from the trade-offs, to satisfy each concern. Section 7 discusses some related work, while section 8 concludes the paper and identifies directions for future work.

2. Background and motivation

Separation of concerns has been contemplated by well-known requirements engineering approaches such as goal-oriented techniques and viewpoints. In goal-oriented approaches [12], such as KAOS [3] and i* [21], 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 [18], 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 we presented in [17] is based on treating PREView concerns as adaptations of the aspect-oriented programming [5] notion of aspects and, consequently, carries out the analysis of broadly scoped properties against a base set of viewpoints. A refinement of the model presented in [15] supports separation of the specification of aspectual requirements, non-aspectual requirements and composition rules in modules representing coherent abstractions and following well-defined templates. The 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.

In Theme/Doc [1] 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. While Theme/Doc treats all concerns as themes, there is still the notion of base themes and crosscutting themes hence diluting the uniformity of the model with a strong base-aspect dichotomy.

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, the subsequent negotiation among stakeholders and the derivation of an architecture design based on the combination of the possible choices for each concern.

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 concerns in the system. In deriving our multi-dimensional approach we have built on the strengths of the model in [15], mainly the informal composition rules with concern-specific actions and operators.

In [14] we proposed a model for multi-dimentional separation of concerns at the RE level. This paper adds to the model by defining a meta concern space, by adding the notion of a compositional intersection which helps to identify suitable sets of concerns in our multi-dimensional separation as a basis to observe trade-offs among other concerns, and by using our early trade-off analyses to provide insights into the architecture choices suited to each concern.

3. Multi-dimensional separation of concerns in RE

Concerns in our multi-dimensional approach imply any coherent collection of requirements. We treat all concerns in a uniform fashion and hence, do not classify concerns into viewpoints, use cases or aspects though our concerns still encapsulate coherent sets of functional and non-functional requirements. We perceive the concern space at the requirements level as a hypercube. Figure 1 shows a simplified representation of this as a cube. 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. The block arrows represent projections. This flexible, multi-dimensional view makes it possible to handle both crosscutting functional and non-functional requirements in an effective fashion. We observe later, in Section 6, that the chosen architecture lies somewhere in the area bounded by the hypercube walls.

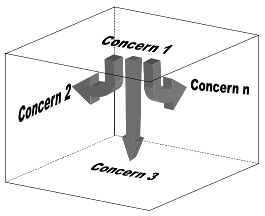

Figure 1. Hypercube representing concerns in a system

3.1. Running example

In order to illustrate the various concepts and techniques, we use a case study based on a location and context sensitive tourist guide system developed at Lancaster [4]. The system has the following key characteristics:

"It 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."

The various requirements-level concerns in this particular system and their categorisation based on our meta concern space is described next. This is followed by a discussion of our concern composition and trade-off analysis mechanisms.

4. Meta concern space and system space

When developing our multi-dimensional approach to RE, we have taken into account the fact that most requirements engineers, and developers in general for that matter, are used to thinking in two-dimensional terms. A shift from such a two-dimensional perspective on requirements engineering to a fully multi-dimensional view poses a significant cognitive challenge. Furthermore, requirements engineers and developers often still find it a non-trivial task to identify and categorise requirements using well-established mechanisms such as viewpoints and use cases. So it is even more crucial to facilitate identification of concerns that would form suitable candidates for providing a multi-dimensional view and subsequent analysis of such a view.

Concern identification in our approach is based on the observation that certain concerns, both functional and non-functional, appear time and again during system development. A catalogue of non-functional concerns has been provided by [2] but we extend the notion of such a catalogue to typical functional concerns. Examples of such repeatedly appearing concerns include registration, ordering, billing, booking, mobility, availability and security (note that the examples include both functional and non-functional concerns). Based on this observation, we divide the requirements space into two separate spaces (cf. Figure 2):

- System space, which comprises of the various types of systems developers want to realise.
- Meta concern space, which comprises of the above mentioned abstract set of typical concerns (functional and non-functional), which repeatedly manifest themselves in various systems.

Each system in the system space has a number of desirable features. These lead to establishment of the requirements (gray dots in Figure 2), through interviews, ethnographic studies, analysis of business practices, etc., of the system to be developed (depicted by the solid arrows in Figure 2). Once the requirements have been derived from desirable system features, we categorise them into concerns from the meta concern space (shown by dashed arrows in Figure 2). This leads to concrete, system- and domain-specific definitions (ovals in Figure 2) of the abstract concerns and can be achieved iteratively and incrementally, by handling a small set of concerns at a time. Note that not all concerns in the meta concern space are necessarily used during this categorisation; mostly only a subset of concerns relevant to the problem domain is needed. The categorisation of requirements into concrete concerns leads to creation of a relationship between the meta concern space and the system space (shown by the gray arrow in Figure 2). This creates a conceptual binding between the abstract representations of concerns in the meta concern space with their concrete representations in the system space.

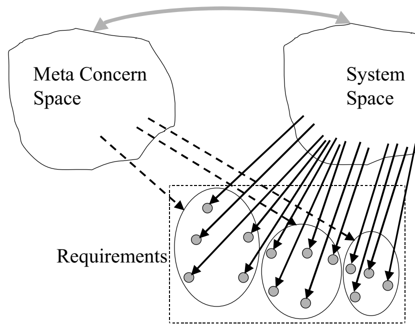

Figure 2. The system space and meta concern space

From the outline specification of our tourist guide case study, we can identify several concerns from the meta concern space manifesting themselves. For example, the visitors access information while moving between locations so *Information Retrieval* and *Mobility* are two concerns of interest. The system also interacts with external reservation services so compatibility is also a concern. Other concerns are more clearly visible in the detailed system descriptions, notes of discussions and interviews with stakeholders as well as domain knowledge of experts in designing mobile interactive systems. Consequently, the concrete concerns (from the catalogue of abstract concerns in the meta concern space) we have identified for our mobile tourist guide case study are:

- *Authentication:* to check personal information against a valid ID before loaning the mobile device to a visitor.
- *Availability:* to ensure that the system is always reactive to stimuli.
- *Compatibility:* to ensure interaction with external services such as hotel and ticket reservation.
- *Connectivity:* to provide network connections to access external services.
- *Context:* to recognise the change in location as a visitor moves around the city.
- *Customisability:* to allow the visitor to configure the tours to their personal preferences and interests.
- *Information Retrieval:* to obtain information (e.g., by visitors or the tourist information centre staff) from the system.
- *Information Update:* to add, remove or update information about the various sites of interest to tourists.
- *Mobility:* to ensure that visitors can access the system while on the move.
- *Navigation:* to provide directions for a visitor to follow a tour.
- *Portability:* to provide lightweight devices for accessing the system.
- *Registration:* to obtain personal information from visitors before loaning out the electronic devices for accessing the system.
- *Cost:* to account for costs (e.g., development, equipment, deployment, operation maintenance).

For the concrete realisation of our approach, we have chosen to represent both the abstract concerns in the meta concern space as well as their concrete realisations using well-defined templates based on the eXtensible Markup Language (XML). Such a semi-structured representation of the concerns makes it possible for us to define composition rules specifying how a concern constrains or influences the requirements in other concerns (cf. section 5). It also makes it possible to analyse the concerns and their compositions for establishment of potential trade-off points. A similar XML-based approach effectively facilitated partial automation of the requirements analysis in our earlier work on the Aspectual Requirements Composition and Decision Support tool, ARCADE [15]. In fact, our composition rules build upon the composition operators and actions developed as part of this earlier work (cf. section 5).

Throughout the paper we will be using the concerns Information Retrieval and Mobility from our tourist guide case study to illustrate our approach. Figures 3 and 4 show the abstract definitions of these concerns in the meta concern space. The abstract definitions are enclosed in <MetaConcern> </MetaConcern> tags, which also specify the name of the concern in the meta concern space. The definition also includes a brief description of the concern, some typical examples of use derived from past experience and domain knowledge and the (meta) concerns it might relate to or interact with (these concern names are enclosed in the <Relationships> </Relationships> tags). Note that the relationships information is provided as a helpful guide and may not hold for all systems and, in several cases, there might be additional relationships with other concerns in a system.

```
 <?xml version="1.0" ?>
- <MetaConcern name="InformationRetrieval">
     <Description>The operation of accessing information from a
          computer system </Description>
     <Examples>Database retrieval, Multimedia retrieval</Examples>
     <Relationships> Availability, Mobility, InformationUpdate
          </Relationships>
  </MetaConcern>
```

Figure 3. *InformationRetrieval* meta concern in XML

```
 <?xml version="1.0" ?>
- <MetaConcern name="Mobility">
     <Description>The quality of moving freely </Description>
     <Examples>Wireless networks, Mobile phones, Context aware
          systems </Examples>
     <Relationships> Availability, Portability, Context </Relationships>
  </MetaConcern>
```

Figure 4. *Mobility* meta concern in XML

Figures 5 and 6 show the XML specifications of the concrete concerns InformationRetrieval and Mobility in our mobile tourist guide. 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 also be used to realise naming scopes.

```
 <?xml version="1.0" ?>
- <Concern name="InformationRetrieval">
   - <Requirement id="1">
        It should be possible to retrieve information from the system.
        <Requirement id="1.1">It should be possible to access
             information about the attractions. </Requirement>
        <Requirement id="1.2">It should be possible to access
             information about the current location. </Requirement>
     </Requirement>
     <Requirement id="1.3">It should be possible to obtain a list of
          available preset tours. </Requirement>
  </Concern>
```

Figure 5. *InformationRetrival* 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>
```

Figure 6. *Mobility* concern in XML

5. Composition and trade-off analysis

Having categorised the various requirements of the system into concrete concerns from our meta concern space, we move onto defining composition rules for each concern. This is followed by choosing specific sets of concerns, in a methodical fashion, as base to observe trade-offs among other concerns.

5.1. Composition specification

The potential relationships that a concern might bear with other concerns are documented in our abstract concern definitions in the meta concern space. These relationships might not hold in case of all systems and in some cases additional relationships with other concerns might arise. However, they still provide a good starting point for a requirements engineer to start specifying how a concern constrains or influences requirements in other concerns it relates to. In other words, our composition specification, provided in the form of composition rules, describes how a concern cuts across other concerns in our multi-dimensional separation.

Composition rules define the relationships between concerns requirements at a fine granularity. Like our concern definitions, we have an XML-based composition specification language to specify the composition rules. Note that 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 and 8, a coherent set of composition rules is encapsulated in a Composition tag. Figure 7 encapsulates all compositions for InformationRetrieval while Figure 8 does so for Mobility. The semantics of the Requirement tag here differ from the tags in the concern definition. The concern to which a requirement belongs is explicitly listed as an attribute; this is essential for scoping purposes. If a concern requirement has any sub-requirements these must be explicitly excluded or included. This is done by providing an "include" or "exclude" value for the optional children attribute. A value of "all" for the *id* attribute in a Requirement tag 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 requirements from one concern constrain those in a set of other concerns. 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. 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.

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: "**Information Retrieval** must be **provide**d during **all the Customisability** requirements, requirement **1** of **Navigation** and requirement **1** of **Mobility**, **includ**ing its children, with the outcome that the specified constraint is **fulfilled**". The second part of the composition rule should be interpreted in a similar fashion.

```
 <?xml version="1.0" ?>
- <Composition>
  - <Requirement concern="InformationRetrieval" id="all">
    - <Constraint action="provide" operator="during">
        <Requirement concern="Customisability" id="all" />
        <Requirement concern="Navigation" id="1" />
        <Requirement concern="Mobility" id="1" children="include" />
        <Requirement concern="InformationUpdate" id="all" />
      </Constraint>
      <Outcome action="fulfilled" />
    </Requirement>
 </Composition>
```

Figure 7. Composition rule for *InformationRetrieval*

```
<?xml version="1.0" ?>
- <Composition>
   - <Requirement concern="Mobility" id="all">
      - <Constraint action="affect" operator="on">
           <Requirement concern="Availability" id="all" />
           <Requirement concern="Connectivity" id="all" />
           <Requirement concern="Context" id="all" />
           <Requirement concern="Navigation" id="1" />
           <Requirement concern="InformationRetrieval" id="all" />
        </Constraint>
        <Outcome action="fulfilled" />
     </Requirement>
 </Composition>
```

Figure 8. Composition rule for *Mobility*

Composition rules in the multi-dimensional approach have been inspired by our earlier work on composition of aspectual requirements during viewpoint-oriented requirements engineering [15]. The key difference is that a concern constrains other concerns and not only viewpoints.

Tables 1, 2 and 3 describe the semantics of the actions and operators for Constraint and Outcome, including those used in the composition rules shown in figures 7, 8.

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. We have validated these in other case studies previously, e.g., [15], and aim to continue validation with further case studies.

Table 1. 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 *all* is specified.
affect	Used to specify that a set of concern requirements will alter the state of another concern.

Table 2. 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)

Table 3. 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.

5.2. Compositional intersection

In order to identify trade-off points, one needs to observe the interactions of a concern with other concerns with reference to some base. This is quite straightforward in two-dimensional approaches to separation of concerns in RE, e.g., PREView [18], the NFR Framework [2] and our earlier work on aspect-oriented requirements engineering [15, 17]. In all these approaches, functional requirements provide a preset base for observing interactions among the non-functional concerns in order to identify potential trade-off points. Such a preset base is not readily available in our multi-dimensional separation as all concerns are peers and a particular type of concerns does not dominate those of another.

We can see from the composition rules examples in section 5.1 that functional requirements (e.g., InformationRetrieval) can also constrain or influence requirements within non-functional concerns in our multi-dimensional separation. Therefore, the trade-off analysis also must be multi-dimensional in nature. The brute force method of doing this would be to choose every possible combination of concerns as a basis to study interactions among two concerns and then synthesising the results of such an analysis. However, this poses a significant overhead, even in the presence of tool support, as the number of potential combinations of concerns to be used as a base would be extremely high. We, therefore, introduce the notion of a *compositional intersection* (denoted by "$\cap$") to constrain the potential combinations of concerns to be used as a base to those combinations that have real value to offer in terms of requirements-level trade-off analysis.

Let C_1, C_2, C_3, …, C_n be the concrete concerns in our system requirements and Sc_1, Sc_2, Sc_3, …, Sc_n be the sets of concerns that each of them cuts across respectively. Let us suppose we want to identify the trade-offs (if any) between C_1 and C_2. In order to do this we should take the compositional intersection of Sc_1 and Sc_2. However, note that a compositional intersection is not a simple intersection as in set theory. Let C_a be a member of both Sc_1 and Sc_2. C_a will appear in the compositional intersection iff both C_1 and C_2 influence/constrain the same or overlapping set of requirements in C_a. That is, if C_1 and C_2 influence disjoint sets then C_a will not be in the compositional intersection.

If the result of the compositional intersection is a non-empty set, we need to analyse the trade-offs and specify any priorities. The process is repeated for C_1 and C_3 and so on until C_1 and C_n. It is then repeated for C_2 but, obviously, the compositional intersection for C_2 and C_1 does not need to be repeated since it has already been carried out when identifying trade-off points for C_1. The process continues to be repeated for all concerns up to C_n. This means that the maximum number of compositional intersections we have to take is given by:

$$C_n^2 = \frac{n!}{(n-2)!*2!}$$

where n is the number of concerns.

Let us consider a requirements specification with n concerns where the set of concerns each concern cuts across is given by $S_{C1}=\{C_2, C_5, C_7, C_n\}$, $S_{C2} = \{C_5, C_6, C_n\}$ and $S_{Cn} = \{C_1, C_2, C_9\}$. The remaining concerns do not cut across any other concern. Let us represent the compositional intersection set between concerns C_i and C_j as $S_{Ci} \cap S_{Cj}$. The list of possible compositional intersection sets will be:

$$S_{C1} \cap S_{C2} = \{C_5, C_n\}$$
$$S_{C1} \cap S_{Cn} = \{C_2\}$$
$$S_{C2} \cap S_{Cn} = \phi$$

This illustrates that the number of compositional intersections is likely to be smaller than the combinatorial number, since some concerns may not affect, i.e., be related to, many other concerns.

For our tourist guide example, if $S_{\text{InformationRetrieval}}$ = {Customisability, Mobility, InformationUpdate, Navigation} denotes the set of concerns that InformationRetrival cuts across and S_{Mobility} = {Availability, Connectivity, Context, Navigation, InformationRetrieval} denotes a similar set for Mobility, the compositional intersection is given by $S_{\text{InformationReteriaval}} \cap S_{\text{Mobility}}$ = {Navigation} since both composition rules affect requirement 1 of Navigation.

The compositional intersections provide us with the trade-off points for our trade-off analysis.

5.3. Trade-off analysis

Trade-offs are analysed based on the type of contribution one concern may have on another with respect to the base identified via the compositional intersection. These contributions may be positive, negative or "none".

Firstly, we build a contribution matrix (cf. Table 4) where each concern may contribute negatively (-) or positively (+) to the others (empty cells represent "don't care" contributions). Each cell shows the type of contribution ("-" or "+") and also the compositional intersection set used to find the contribution. If there is no contribution in one of the directions of the relationship, then the cell only shows the compositional intersection. Empty cells denote the inexistence of a relationship. Therefore, the relationships represented in Table 4 are bi-directional, meaning that we should analyse the relationship in both directions, with respect to the same base. For example, C_1 contributes positively to C_2 with respect to C_5 and C_n while C_2 contributes negatively to C_1 with respect to C_5 and C_n. The "composed contribution" is negative. This means that the direction of the contribution relationship is important.

The decision about the type of contribution for each particular case is usually difficult to make, especially if the base is a set with several concerns. We may look for inspiration in existing catalogues, such as [2] as well as experimental and empirical domain knowledge.

Table 4. Contributions between concerns

	C_1	C_2	...	C_n
C_1	▨	+ $\{C_5,C_n\}$		+ $\{C_2\}$
C_2	– $\{C_5,C_n\}$	▨		
...			▨	
C_n	+ $\{C_2\}$			▨

Table 5 depicts the contributions between some of the concerns that compose our tourist guide example.

Table 5. Part of the contribution matrix for the guiding system
(**Cont:** Context; **Cust:** customisability; **IU:** Information Update; **Mob:** Mobility; **Nav:** Navigability)

	Availability	Information Retrieval	Mobility	...
Availability	▨	**+ {Cust, Nav, Mob, IU}**	**+ {Cont, Nav}**	
Information Retrieval	**{Cust, Nav, Mob, IU}**	▨	**– {Nav}**	
Mobility	**– {Cont, Nav}**	**– {Nav}**	▨	
...				

Focusing our attention on the trade-offs between InformationRetrieval and Mobility concerns, we must study the contribution between them with respect to the base composed of Navigation. Mobility contributes negatively to InformationRetrieval with respect to Navigation. This means that the more a visitor moves, the more difficulties s/he will have to retrieve information from the system. The contribution in the opposite direction is also negative, since the more complex the information needed is, the less mobile the system may be, as wireless networks have limited bandwidth.

Having identified the trade-off points, from a multidimensional perspective, we must now study the cumulative effect of a set of concerns on a given concern. Let P_1 be the set of concerns that contribute positively or negatively to C_1. We can project these influences on to C_1 to see how the positive or negative contributions affect C_1. We repeat the process for all the concerns up to C_n.

This process can be better illustrated by folding each successive column on one another, to obtain the cumulative effect for situations where several concerns directly influence a specific one (see Figure 9). Given that the concerns are repeated in the columns and rows, the cumulative effect can be observed in both directions, e.g., from Availability to InformationRetrieval and from InformationRetrieval to Availability. This folding provides us the cumulative projections: the combined influence of a set of concerns on a particular concern.

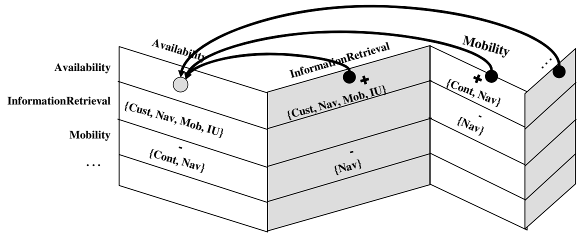

Figure 9. Contribution table folded along its columns

For example, the cumulative effect of InformationRetrieval and Mobility on Availability is positive and, therefore, no trade-off is necessary. On the other hand, the cumulative effect of Availability and InformationRetrieval on Mobility is negative. Therefore, a trade-off must be made in order to handle this conflicting situation. In [15] we have proposed a possible solution. The main ideas are still useful in a multidimensional approach. The solution proposed is to attribute weights to those concerns that contribute negatively to each other in relation to a particular base (the base in [15] was a set of viewpoints, as mentioned previously). 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. These 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. The major problem occurs when the two concerns involved in the conflict have the same priority. In this case a decision must be taken by the stakeholders to lower one of the priorities.

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.

6. Architectural choices

Now that we have undertaken trade-off analysis of our requirements level concerns, we are armed with a better understanding of them when making architecture choices. Before we discuss how these trade-offs identified at the requirements level *pull* our choice of architecture in various directions, it is important to note that each concern in the multi-dimensional separation leads to a number of architecture choices that would serve its needs with varying levels of stakeholder satisfaction (cf. Figure 10). The gray boxes in Figure 10 show the ideal or most suitable architectural choices for each concern. Note that these architectural choices are unlikely to be the same and could even be conflicting (which is often the case). Our trade-off analysis at the requirements level warns us about such potential conflicts and we can observe, and appreciate, them clearly when we start making architecture choices. This highlights the importance of the early trade-off analysis we carry out at the requirements level, as some of these conflicts are resolved early via stakeholder negotiations and prioritisation of concerns.

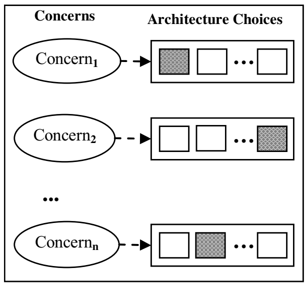

Figure 10. Architectural choices to satisfy each concern

Returning to our tourist guide case study, the Availability concern can be satisfied by a number of architecture choices, e.g., via multiple replica servers, high integrity network links, or a combination of both these factors. At the same time, we can observe that the Mobility concern requires an architecture based on a wireless network so that visitors can walk around while accessing the information. This is at odds with the architectural needs for Availability (our requirements analysis already alerted us to the negative contribution between Mobility and Availability), as wireless networks tend to be less reliable than traditional wired networks. Maintaining Availability in a wireless network would require a small number of powerful wireless base stations in a restricted area. However, this would compromise Mobility, which aims to allow users to roam as freely as possible.

We can observe similar situations for other concerns. Consider, for example, InformationRetrieval. The various architectural choices pertain to different kinds of media made available to the visitors. These could range from a simple text-based retrieval mechanism to multimedia retrieval involving photographs, graphic maps, animations, videos, etc. Of course, our ideal choice for information retrieval would be a fully-fledged multimedia solution. However, we observed in section 5.3 that Mobility and InformationRetrieval contribute negatively to each other. We also noted that this negative contribution was with reference to Navigation. If we look at our architectural choice for Mobility, this analysis makes sense. An extensive multimedia solution would help the visitor retrieve information that is more comprehensible and useful. At the same time, mechanisms such as graphic maps would help the visitor navigate more effectively. However, the bandwidth available over a wireless connection in a busy city centre is fairly limited (especially if there are fewer base stations deployed). This means we would need to compromise to a less elaborate multimedia solution for information retrieval or accept that information retrieval may not always be possible, e.g., when the user drifts in and out of the range of a wireless base station.

Cost is a concern that is likely to conflict with architectural choices for other concerns and often plays a major role in determining the final system architecture. In our tourist guide system, choices such as the number/variety of replica servers, the number/variety of wireless base stations, the amount of multimedia content developed, etc. are strongly influenced and constrained by Cost. From a Cost perspective the ideal architecture is the one that costs least but this is unlikely to be the optimal architecture to support other concerns.

All these, often conflicting architectural choices *pull* our final architecture choice in various directions (cf. Figure 11). Our requirements-level trade-off analysis gives us some early insights into such a pull and helps us resolve some of the conflicts. However, it is when we make our architecture choice that we have to identify and maintain the optimal position for the architecture in the presence of such diverse and conflicting needs of concerns.

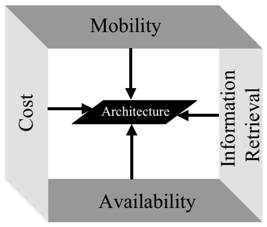

Figure 11. Architectural pull of various concerns

7. Related work

There are other mechanisms that aim to support a uniform separation of concerns during requirements engineering. Problem frames [8] focus on the environment in which a system is located instead of the system itself or its interfaces. Problem frames are concerns and each one can be seen as a single-dimension.

The NFR framework [2] focuses on non-functional requirements. It does not explicitly deal with functional concerns, but establishes a link to them. Also, it does not take into account the crosscutting nature of those requirements. There are approaches that integrate functional and non-functional concerns [3], but, again, the crosscutting nature of those concerns is not addressed.

Multi-dimensional separation of concerns is supported by Hyperspaces [20] and Cosmos [19]. 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 [7]. 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 [13]. 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 [22], an approach is proposed for discovering aspects from relationships between goals. This is accomplished by using a goal model, where functional and non-functional requirements are represented through goals and softgoals, plus tasks that contribute to their satisfaction. The model is analysed in order to identify aspects. Our work, in contrast, takes a multi-dimensional perspective by treating concerns and their trade-offs uniformly across requirements and architecture. This can be further mapped onto any design and implementation approach supporting multi-dimensionality, e.g., [20].

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.

The PROBE framework [10] supports traceability of aspectual requirements and associated trade-offs to detailed design and implementation. Our multi-dimensional approach focuses on tracing trade-offs to architectural decisions. It would be interesting to extend PROBE to trace the multi-dimensional separation and trade-offs to detailed design and implementation.

8. Conclusions

This paper has proposed a multi-dimensional approach to separation of concerns in requirements engineering as well as trade-off analysis of the requirements specification from such a multi-dimensional perspective. We focus on removing the notion of a fixed functional base with respect to which trade-offs among non-functional concerns are traditionally observed, analysed and resolved; this leads to an architecture that is misaligned with the initial system requirements as the architecture choices that would have otherwise been driven by functional requirements and their associated trade-offs are largely ignored. The proposed multi-dimensional approach addresses this misalignment by treating all concerns, whether functional or non-functional, as peers. This provides the requirements engineer with an opportunity to analyse the influence of crosscutting functional properties on other requirements in the system. It also facilitates analysis of trade-offs arising from this and negotiation amongst stakeholders.

One of the key elements of the approach is the notion of a meta concern space, a catalogue of typical concerns, functional and non-functional, that manifest themselves time and again in various software systems. The abstract concern definitions in this meta concern space are used as a basis to delineate requirements into concrete concerns. Another novel aspect of the work is the notion of a compositional intersection which significantly limits the potential number of concerns to be used as a base to observe trade-offs between two concerns. This is crucial as the compositional intersection allows us to limit this number without compromising the rigour of the trade-off analysis.

The trade-off analysis and stakeholder negotiation supported by our approach 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. The early, multi-dimensional trade-off analysis provides the requirements engineers with insights into problematic interactions amongst concerns as well as the cumulative effect of multiple concerns on a single concern in the system. These can then be discussed and resolved with stakeholders before the architecture is derived. This provides an opportunity to remedy some of the intricacies of reaching an optimal architecture choice. Even so, each concern in our multi-dimensional separation still has a number of architectural choices (of varying degrees of suitability) to satisfy its requirements. We can choose the most suitable architectural choice for each concern but further trade-offs must be made as the various choices pull the architecture in their respective directions. The optimal architecture is the one that involves architectural choices satisfying each concern in the multi-dimensional separation within some acceptable limits. These limits are derived from discussion with stakeholders during requirements level trade-off analysis and subsequent negotiations. This provides effective traceability of trade-offs and decisions from the requirements level to the architecture.

The multi-dimensional approach presented in this paper is a key stepping stone towards more rigorous analysis of requirements. To date, requirements engineering approaches have remained largely two-dimensional in their approach to such analysis. Our future work will focus on further case studies to validate the multi-dimensional approach in structuring and analysing requirements in a variety of systems and domains. 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).

9. References

[1] E. Baniassad, S. Clarke, "Theme: An Approach for Aspect-Oriented Analysis and Design", Proc ICSE 2004.

[2] L. Chung, et al., Non-Functional Requirements in Software Engineering: Kluwer, 2000.

[3] A. Dardenne, A. Lamsweerde, and S. Fickas, "Goal-directed Requirements Acquisition", Science of Computer Programming, Vol. 20, pp. 3-50, 1993.

[4] 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.

[5] T. Elrad, R. Filman, A. Bader (eds.), "Theme Section on Aspect-Oriented Programming", CACM, 44(10), 2001.

[6] A. Finkelstein, I. Sommerville, "The Viewpoints FAQ." BCS/IEE Software Engineering Journal, 11(1), 1996.

[7] J. Grundy, "Aspect-Oriented Requirements Engineering for Component-based Software Systems", 4th IEEE Int'l Symp. on RE, 1999, IEEE CS Press, pp. 84-91.

[8] M. Jackson, Problem Frames: Analyzing and Structuring Software Development Problems: Addison Wesley, 2000.

[9] I. Jacobson, Object-Oriented Software Engineering - a Use Case Driven Approach: Addison-Wesley, 1992.

[10] S. Katz, A. Rashid, "From Aspectual Requirements to Proof Obligations for Aspect-Oriented Systems", Proc. RE, 2004, IEEE CS Press, pp. 43-52.

[11] R. Kazman, M. Klein, M. Barbacci, T. Longstaff, H. Lipson, and J. Carriere, "The Architecture Tradeoff Analysis Method", Int'l Conf. Engg. Complex Computer Systems (ICECCS), 1998, IEEE CS Press, pp. 68-78.

[12] A. Lamsweerde, "Goal-Oriented Requirements Engineering: A Guided Tour", 5th Int'l Symp. on RE, 2001, IEEE CS Press, pp. 249-261.

[13] A. Moreira, J. Araújo, and I. Brito, "Crosscutting Quality Attributes for Requirements Engineering", Proc. SEKE, 2002, ACM, pp. 167-174.

[14] A. Moreira, J. Araújo, A. Rashid "A Concern-Oriented Requirements Engineering Model", Proc. CAiSE 2005, LNCS, Vol. 3520, pp 293-308.

[15] A. Rashid, A. Moreira, J. Araújo, "Modularisation and Composition of Aspectual Requirements", Proc. AOSD, 2003, ACM, pp. 11-20.

[16] A. Rashid, P. Sawyer, "Aspect-Orientation and Database Systems: An Effective Customisation Approach", IEE Proceedings - Software, 148(5), pp. 156-164, 2001.

[17] A. Rashid, P. Sawyer, A. Moreira, J. Araújo, "Early Aspects: A Model for Aspect-Oriented Requirements Engineering", Proc. RE, 2002, IEEE CS Press, pp. 199-202.

[18] I. Sommerville, P. Sawyer, Requirements Engineering - A Good Practice Guide: John Wiley and Sons, 1997.

[19] S. M. Sutton, I. Rouvellou, "Modeling of Software Concerns in Cosmos", Proc. AOSD, 2002, ACM, pp. 127-133.

[20] P. L. Tarr, H. Ossher, W. H. Harrison, and S. M. Sutton, "N Degrees of Separation: Multi-Dimensional Separation of Concerns", Proc. ICSE, 1999, ACM, pp. 107-119.

[21] E. Yu, "Modelling Strategic Relationships for Process Reengineering": PhD Thesis, University of Toronto, 1995.

[22] Y. Yu, et al., "From Goals to Aspects: Discovering Aspects from Requirements Goal Models", Proc. RE 2004, IEEE CS Press, pp. 38-47.