Requirements Engineering for Web Applications

PhD Student: CĂLIN EUGEN NICOLAE GAL-CHIȘ
Supervisor: Prof. Univ. BAZIL PÂRV
Faculty of Mathematics and Computer Science
Babeș-Bolyai University
Cluj Napoca - România

A thesis summary submitted for the degree of PhD
Doctor of Philosophy (PhD)
August 2017

Table of Contents

1. Introduction

This work relates *Requirements Engineering* and *Web Engineering* paradigms using a separation of concerns approach.

A **Requirement** is "a description of a service the stakeholders expect of the system, the system behavior, and constraints and standards that it should meet" [114]. The requirements can be derived in sub-requirements and in the same way, concerns can have sub-concerns. According to Friedental et all [109]: "a requirement specifies a capability or condition that must be satisfied. A requirement can define a function that a system must perform, or a performance condition a system must achieve". Usually, requirements can be expressed textually or graphically, using diagrams, and are considered model elements. They are related to other model elements (including other requirements).

A **Concern** is known to be "a high-level unit for system partitioning, a container for localizing semantically related requirements; it can be simple (containing only requirements), or composite (containing other concerns as well as requirements), thus allowing hierarchical structuring of related requirements" [22].

Ossher [91] defines **Separation of Concerns** (SoC) saying that: "in its most general form, the separation of concerns refers to the ability to identify, encapsulate, and manipulate those parts of software that are relevant to a particular concept, goal or purpose.". According to this, a 'concern' is an entity that encapsulates a particular area of interest. The concerns are the main factor in the decision of refining software into more specific, atomic elements. Related concerns are grouped in so-called *concern spaces*. Multi-dimensional SoC refers to SoC using several concern spaces.

The concerns and requirements are entities identified by elements spread along multiple sentences. Regardless, the smallest entity of interaction is a simple sentence, and for the purpose of relating to this *smallest unity of interaction* the elements composing it are needed to be defined. According to Chitchyan et all. [22], the main such elements in the Requirements Description Language are *subject*, *object*, and *relationship*.

Requirements Engineering (RE) is defined as "the area of software engineering that focuses on the RE process which involves understanding customer needs and expectations (requirements elicitation), requirements analysis and specification, requirements prioritization, requirements derivation, partitioning and allocation, requirements tracing, requirements management, requirements verification, and requirements validation" [131].

RE is a field in the Software Engineering discipline and is related to the software requirements. RE is the process in which the requirements of the software systems are methodologically managed regarding how they are formulated, the documents they are specified in or related to them, and the way they are maintained. RE is defined [60] as "the task of capturing, structuring, and accurately representing the user's requirements so that they can be correctly embodied in systems which meet those requirements". More generally, RE is [94]

the process of identifying and articulating needs for a new technology and applications.

Web Engineering (WE) is "the establishment and use of sound scientific, engineering and management principles and disciplined and systematic approaches to the successful development, deployment and maintenance of high quality Web-based applications" [84].

WE is a discipline that promotes the process of producing quality web-based applications using appropriate methodologies, approaches, techniques and tools. WE directions are guiding the process of web application design, development, evolution and evaluation. WE and Software Engineering are related in the sense that WE can be considered as Software Engineering meant to meet the specific profile of web-based applications.

All content delivered through Internet technology is part of Web sites, Web applications or Web-based systems. All software systems generating this content are referred in this thesis as *web applications*. On the other hand, for the applications not based on web the term *traditional software* is used.

In this thesis is presented a RE approach that improves Web application's requirements management. The approach is based on the separation of concerns. Managing the requirements using separation of concerns provides a basis to requirements engineering process along the development cycle.

The requirements engineering in the web application development is somehow similar to the engineering process in the development of traditional software. The main differences are found in the process of capturing web-specific aspects like navigation and technology and in the way that the requirements are implemented into code.

In order to address the need of managing the artefacts of web applications and to facilitate the reuse of artefacts such as code and documentation, a methodology of managing the web requirements is introduced. This methodology may support also the traditional software development and even other new fields, but, in this thesis, we will focus on the web application field. The methodology is based on multidimensional SoC and uses model-to-model transformations and links between models to capture the specific of different web applications using a similarity metric. This approach will allow us to access and reuse the artefacts of a web application in order support the development of another one.

This chapter is organized as follows: in Section 1, the purpose of the thesis is explained. In Section 2 is presented the problem to be solved and main research objectives. Section 3 focuses on the original contributions.

1.1 Motivation

It is very important for requirements engineering activities to be efficient in order to fulfill the users and customers' needs, and that the application should be delivered on time with respects to release time frame and targeted budget [3]. Once the Web Engineering developed in the last decade of the twentieth century, one of the main gaps observed was the lack of proper requirements specification techniques for Web applications [84] [32].

In the domain of the Requirements Engineering there were presented several approaches in the last decades. Some approaches, like Constantine and Lockwood [28], Jaaksi [65], Leite *et al.* [77], Durán *et al.* [35] or Rosenberg and Scott [105], were excellent regarding the requirements specification linked to the structure and behavior of software systems. Other approaches, like of Chung *et al.* [23], Cysneiros *et al.* [30], or Botella *et al.* [10], focused with success on methods to describe non-functional requirements, like usability, reliability, reusability or performance of software systems, collectively known as URPS.

According to Cachero and Koch [13], Schwabe *et al.* [112], Ceri *et al.* [17], all these approaches were missing support for expressing the Web application specific requirements, that have to consider critical Web applications features, such as *navigation*. The navigation is important in Web applications which are based on heavy user interaction [54], as the WWW role is to be a support for information distribution - many web applications being developed as information repositories. Web application development involves complex data capturing and organizing in order to make it available to users. According to hypertext paradigm, Conklin states that users can *navigate* information in non-linear directions in a web of links, nodes and anchors [26].

Approaches like OOHDM [112], WebML [17], UWE [71], WSDM [31], W2000 [9], SOHDM [76], RNA [130] or OOH [14] are supporting the development of Web applications. All of those are offering solutions on representing the navigational aspects.

As the projects are larger and more complex, relevant principles like modularization, abstraction and encapsulation [106] are followed, as they provide an important support in reaching a clear separation of concerns. Structured or Object-Oriented Software Engineering methods are providing methodologies based on such principles, but they are not able to completely separate complex relations between concerns as they focus the modularization process on a dominant decomposition criteria. This *tyranny of the dominant decomposition* [119] leads to an ineffective modularization process focused on a single concern.

Aspect-Oriented Software Development (AOSD) provides a better support in managing the separation of concerns process and reach better modularity by decomposing the system in parts that have as less as possible overlapping functionalities [1]. Also, AOSD purpose is to manage the crosscutting concerns by stressing a methodology based on "systematic identification, separation, representation and composition of concerns" [98]. Basically, AOSD primary targets the crosscutting concerns and their modularization and encapsulation processes. *Crosscutting concerns*, meaning *related concerns*, are solution space attributes that cannot cleanly separate. Their presence affects system architecture and produces code duplication and undesired dependencies between system components. Recent work was dedicated to crosscutting concerns management, not just in the construction stage, but also in earlier phases of software development, like detailed design [24], architecture design [120] and requirement analysis [99, 81].

In order to achieve such a goal, we need (a) a *methodology* of grouping and addressing these artefacts, based on their semantics expressed in terms of concerns, and (b) a *process* of creating relationships from requirements to not-yet-developed artefacts or other existing candidate solutions. Such a methodology will

support artefact reuse (including code reuse), based on their semantic similarity (defined by concern spaces), offering support and guidelines to reuse existing artefacts from identical or similar application domains.

1.2 Problem statement

As we have seen, the requirements engineering of web applications is still an open research topic, as there are areas that can be improved.

Organizing requirements and artefacts in categories and creating links between them can lead to reuse of artefacts and code.

Research question 1: How should web applications requirements be specified and organized in order to capture their attributes and semantics and group them in categories?

Research question 2: What kind of bindings between models can be created (and how) in order to access and reuse existing solutions or resources?

Problem Statement. Provide a methodology for the management of relationships between resources and concerns to be applied in WE. Two resources involved in the WE process (like requirements, artefacts, code, stakeholders, or any other logical or physical *entities*) can be semantically compared in terms of their similarity with respect to the same concern space. Subsequent goals can be derived from this problem, as follows: **G1**: Management of the requirements, artefacts and concerns specific to WE; **G2**: Definition of the matching function measuring similarity between resources with respect to a concern space; **G3**: Application of the similarity relationship to web engineering process.

1.3 Main Contributions

In this thesis the research conducted lead to the development of a methodology supported by a software tool meant to provide a solution to the above problem.

First, an approach is used to organize requirements and artefacts. The approach is based on multidimensional separation of concerns [81] and was adapted to be applied on different artefacts and resources, not limited to one single type, such as requirements [49].

After the requirements and other artefacts are organized according to the approach, we introduced a method to map the requirements and the resources in different problem spaces to various representative concern spaces, based on their semantic.

Once the matching vector is computed, we can compute a semantic similarity coefficient of different pairs of resources, based on the Gower similarity index [53]. According to the weight given by the similarity coefficients, we can trace semantically similar artefacts in different problem spaces or in the same problem space and decide to reuse them. Model to model transformations are performed during the process to allow the method to produce the expected results.

2. State of Art

During the research process, the most important results and studies related to the paradigms being part of the current thesis domain were investigated, outlining for each, their specific and their core concepts and also pointing out the relevant connection to the theme studied.

In order to capture the best from the *big picture* the following fields have been under focus: Web Applications, Web Methodologies, and Requirements Engineering, Requirements for Web Applications, RE Tools for Web applications, Separation of Concerns and their methodologies, Requirements Semantic Analysis.

These concepts will be presented in this chapter.

2.1 Requirements Engineering

In order to have a better understanding of the requirements engineering in this section will be reviewed aspects related to requirements engineering methodologies, requirements management and methods to represent requirements.

The requirements are specifying what a software product should do or what should perform. Given this, requirements are valuable statements that should be specific, as clear as possible, unambiguous.

The recent work performed by Schmidt outlines the software requirements characteristics [111] and proposes a set of best practices to analyze them [110].

2.1.1 Requirements Engineering Approaches

The most important and notable RE approaches are

- Agent oriented requirements engineering
- Goal Oriented requirements analysis/engineering (GORA/KAOS)
- Model Driven requirements engineering
- Aspect-Oriented requirements engineering (AORE)
- ViewPoint Oriented Approach (VOA)
- Business process-driven requirements engineering

Some of the most advanced frameworks supporting these requirements engineering approaches are i*, fUML, KAOS. Other agile methodologies for requirements engineering are proposing self-adapting selection and prioritization techniques for requirements [96].

2.1.2 Requirements Management

Requirements engineering includes the following steps: elicitation, analysis (elaboration and negotiation), specification (producing a document), and verification (validation).

The requirements can be functional or non-functional. Functional requirements for web applications include: data requirements, user interaction/interface requirements, navigational requirements, personalization requirements, and transactional requirements. Non-functional requirements can be grouped under the FURPS+ acronym (F stands for functionality, U for usability, R for reliability, P for performance, S for supportability, and “+” means technical or

development-related requirements (planning, design, implementation, interface, and physical).

Requirements management is a methodology to elicit, organize, and document the requirements of the system in an engineering fashion. It can also be defined as a process that establishes and maintains changing requirements. Requirements management is very important and helpful for real projects. Some of the most common problems are the tracking of requirements change and the difficulty to write them.

2.1.3 Visualizing Requirements in UML

The purpose of a system is to produce certain functionality for a client, in order to assist him or to service him, make his life easier, more efficient. Usually, the person developing the system is not the person the system is intended for, so, there is the risk of miscommunication in between the intended requirements and the product delivered.

For small systems the risk is reduced, as the prototyping and initial versions can be easily adjusted, and the communication between users and developers is not complete, reducing the misunderstanding gap.

On the other hand, for large systems this approach is not suitable, usually leading to projects that will not meet deadlines or budget requirements. For these large projects, most of the developers are not in the same location, some being subcontracted from different companies, and certainly, not in contact with the end-user, due to different levels of access to the project and the professional relationship between developer and contractor. This is giving no chance to a direct communication channel between the end parts of this process: developer and user.

2.2 Web applications development methodologies

2.2.1 Web applications

Web has evolved from simple content pages, linked by hyperlinks to complex platforms for business applications. There are now Rich Internet Applications with complex events and business logic, using AJAX, JSON or other technologies. Given this, the initial paradigm for interacting in web applications (static web pages navigated by hyperlinks) is no longer valid. The use of back and forward buttons is not adequate anymore. The applications are behaving differently even depending on a specific browser and its settings.

2.2.2 Web technologies

A lot of programming languages for web applications have tried to make it on the market offering solutions for programmers. Over time, some failed, other succeeded. There is good and not-so-good on both sides. A lot of things have changed. Many of these languages and platforms development discontinued. The only reason I have come to find about some them is this research. Other, new ones came out. At this moment in time, I can say that we have some solid ones, but, no one can predict what will come, or how long, those on top will they will be able to stay there.

In programming a web application, one had to use several different programming languages at once.

2.2.3 Web Application Methodologies With RE Tools Support

The information and content available through Internet technology is part of Web sites, Web applications or Web-based systems. All software systems generating this content are considered *web applications*.

There are several notable types of web applications:

- business integrated systems;
- educational systems;
- commercial systems – e-shops;
- regular static web pages;
- social applications;
- multimedia applications;
- developer applications.

The tools support is critical for any development methodology, including web development ones. For certain methodologies, existing tools can be configured and used, while for other, given their restricted or special specs, this is not possible. With the increase technological progress, methodologies tend to be developed integrated solutions to cover not only methodology support but support during the entire application development cycle. Such web methodologies are OOHDM, OO-H, UWE, and OOWS.

2.3 Concerns and Aspects in Software development

Concerns are particular goals, concepts, or areas of interest. A cross-cutting concern is a concern affecting several software entities like classes or modules.

Aspects are pieces of code implementing behaviors common to multiple classes, being considered reusable entities. From our viewpoint, an aspect is a piece of code implementing a concern. Aspect-oriented programming (AOP) is a programming technique allowing developers to manage crosscutting concerns, grouping them into separate and reusable modules.

AORE focuses on the importance of both functional and non-functional aspects as a need to satisfy three important needs [113], related to: composition, tracking and development of new technologies. On requirements level, an aspect is a property of a requirement or of a set of requirements that is affecting a system as it can restrict or modify a particular behaviour of requirements. Requirements influenced by aspects can be decomposed using different abstractions used in capturing requirements (viewpoints or use cases or scenarios).

AORE is based on four principles [113]:

- The identification of transversal properties in requirements specification, and the semantic analysis of requirements documents;
- The ability to group transversal properties related to a particular aspect in modules at requirements level;

- The ability to visualize the influence of aspects on requirement level (tool supported, visually or structured);
- The ability to compose the requirements influenced by the same aspect into high level requirements.

2.3.1. Separation of Concerns

The main reason of using SoC is to manage a problem in an easier way, by splitting it in modules, which can be later combined to provide a complete answer to the initial problem [33]. This principle is important in software development, as it provides certain benefits as: reducing the complexity, supporting traceability, offering better understanding of the problem. But, one of the most important benefits is that it provides means to small impact changes as the concerns are stored in different modules. The *crosscutting concerns* are of major importance and are going to be addressed later. Some concerns cannot be completely incorporated into one module, leading to different approaches in tackling the problems arising.

2.3.2 Methodologies based on Aspects and Separation of Concerns

This section contains a review of the most important methodologies in Aspect-Oriented Requirement Analysis AORA. Each of them uses concerns in the process, involved in the requirements definition.

2.3.3 AORE approaches and methodologies based on Separation of Concerns

When functional or non-functional crosscutting requirements, are difficult to be separated in modules their effect in the system cannot be precisely quantified. In contrast to other traditional approaches, unsuccessful in dealing with crosscutting concerns, the AORE approaches are trying to compose requirements using a high level of abstraction mechanism to manage crosscutting requirements.

The approaches presented have their own use of concerns, for which they define their own models, targeted to support their specific processes. As the concern concept is not uniformly approached, there is a necessity for a generic model for concerns to be introduced, and accepted. A framework that processes concerns should have several features like modeling concern spaces capabilities, designing the interactions with other entities, supporting patterned processes for a variety of situations.

2.3.4 Conclusions

Concerns are responsible providing meaning to the artefacts, but they add workload to the application development phases. As the concerns can be easily related to the requirements, there is a need to address the concerns at the same time we are addressing the requirements.

In contrast to traditional approaches, AORE considers specifically the crosscutting concerns, proposing a complete management process for requirements, from elicitation and analysis to validation and implementation, with efficient composition techniques for the crosscutting concerns.

3. Requirements Engineering Using MultiCoS

The generality of concern in terms of modeling and relating it to other artefacts was one thing that none of the methodologies presented in Chapter 2 possessed. Each one of these approaches defined and used the concern to fit their own processual needs in the development process.

This chapter, tries to give a solution to the three goals set in the Problem Statement - G1: Management of the requirements, artefacts and concerns specific to WE; G2: Definition of the mapping function measuring similarity between resources with respect to a concern space; G3: Application of the similarity relationship to web engineering process. Sections 3.1 and 3.2 give results belonging to G1, while Sections 3.3 thru 3.5 deal with the definition of the mapping function (G2). Finally, Sections 3.6 thru 3.8 show how the similarity measure is used in a WE process (G3).

3.1 Modeling Concern Spaces Using *MultiCoS*, a Multi-Dimensional Separation of Concerns approach

All artefacts produced during the development of software products, including specifications, requirements, models, source code, and documentation, are labeled with key-words, or represented using specific attributes. The main goal of these labels is related to the semantics of the artefact being described, and is used for categorizing and searching purposes.

Such key-words are, in our approach, concerns. A generic definition states that "a *concern* can be considered to be any matter of interest in a software system" [45]. The approach presented in this thesis is named *MultiCoS* (Multiple Concern Spaces), a Multi-Dimensional Separation of Concerns approach that was first presented in [49]. MultiCoS relies on multi-dimensional separation of concerns [81] briefly introduced in 2.3.3.8.

3.2 Primitives of the *MultiCoS* approach

The following primitives of *MultiCoS* will be defined: Concern, Concern Space, MultiSpace, Entity, Entity Space, and Matching Value. They were adopted from [46] [57] and used in the context of [81].

Definition 1 (Concern) A concern is anything considered relevant to a software system. It has the purpose of describing an attribute of an entity.

In this thesis we use the concern definition given above. Alternative definitions, found in the literature, are similar to some extent:

- A concern refers to "a property which addresses a certain problem that is of interest to one or more stakeholders and which can be defined by a set of coherent requirements" [12]. Given this, concerns are properties a

system has to possess in order to meet required functionalities. Related concerns are grouped in concern spaces.

- A general definition coming from AOSD states that "a concern is any matter of interest in a software system" [45].
- A more specific definition is provided by IEEE which states: "a concern is those interests which pertain to the system's development, its operation or any other characteristics that are critical or otherwise important to one or more stakeholders" [62].

Definition 2 (Concern Space) A Concern Space is "a group of concerns referring to/describing similar capacities (issues/behaviors) of at least one type of entities. A Concern Space has a name, a description, a set of concerns, and a threshold (maxValue)" [48]. Other attributes can be specified, to properly configure the Concern Space.

In the software field, concerns are defined many ways, most of them being related to code, but concerns can be used in all processes related to software development.

Definition 3 (MultiSpace) A MultiSpace is a Concern Space of Concern Spaces. A MultiSpace "has a name, a description, and a set of Concern Spaces. The MultiSpaces can be grouped in a Meta MultiSpace. This process can be applied recursively, on higher levels, if necessary" [48]. The same way concerns can be decomposed into sub-concerns, a set of related concerns (grouped in a concern space) can be considered as a high-dimensional concern, part of a multispace.

Definition 4 (Entity) In our study, an Entity is an object related to the software development process. An entity can be associated to one or more Concerns from a Concern Space. Examples of entities are: requirements, artefacts, users, code modules, development technologies, problem statements, and even concern spaces.

Definition 5 (Entity Space) An Entity Space is a collection of cohesive entities. Different categories of entities can be grouped in separate Entity Spaces - such as Technologies entity space, Requirements entity space, Artefacts entity space, Developers entity space.

One of the additions introduced by the *MultiCoS* is that in our model a concern space can be used to describe *multiple* entity spaces (Figure 22) and *multiple* concern spaces can be defined and related to a specific entity space (Figure 24). The relations formed between the entity spaces and concern spaces aim to add a semantic value to the software engineering field, giving solutions to represent relationships based on dependencies and semantic similarities of qualities and attributes for various software concepts or entities.

For example, two different Entity spaces, each related to a web applications are linked by using the same Concern Space to describe their components.

For every *entity* and for each *concern space* there is a vector expressing the semantic relation of the *entity* to the *concern space*. A mathematical function can be defined to define the relation between an *entity* and a *concern space* as they are

provided as arguments. The corresponding vector of concern semantic values will be the result of the function.

Definition 6 (Matching value) To quantify the relationship between an *entity e* and a specific concern *c*, we will assign it a value, measuring the strength of relationship between *e* and *c*, and reflecting the weight (importance, matching) of the entity *e* with respect to the concern *c*:

$$mv(e,c) = v, \text{ where } e \in E,\ c \in C,\ v \in \mathbf{R} \quad (1)$$

Definition 7 (Semantic mapping, [48]**)** Let *e* be an entity in the entity space *E*, and *C* a *n*-dimensional concern space. The *semantic matching function* f (2) describes the *entity-concern space* relation existing for an entity *e* and a concern space *C, measuring the degree of matching of entity e to each concern* $c_i \in C$.

$$f(e,C) = \boldsymbol{v} = (v_1, v_2, \ldots, v_n), \text{ where } e \in E,\ \boldsymbol{v} \in \mathbf{R}^n \quad (2)$$

The vector $\boldsymbol{v}$ is called *matching vector* of the entity *e* in the concern space *C*. It is obvious that $v_i = mv(e, c_i)$, for all $c_i \in C$.

The concern space *C* is described using the following format:

$$C = (id, name, description, set\ of\ concerns, max\_value)$$

where: *id* is the label (providing unique identification), and *name* is the identifier of the concern space. For each entity *e* and each concern $c \in C$, $mv(e,c) \in [0, max\_value]$. The *max_value* is a threshold signifying the maximum weight an entity can get for any concern in the concern space. In other words, for a given concern space *C*, the threshold *max_value* is greater or equal to the maximum value of any component of any matching vector $\boldsymbol{v}$ in (2).

3.3 Modeling Entity Spaces using *MultiCoS*

3.3.1 Addressing the same concern space

Having these matching vectors computed, we can make informed decisions regarding the implementation of modules and available developers.

This way, we introduce a metric (numerical value) characterizing their efficiency and being, this way, a comparison tool. So, we observe that developer *Dev3* does not have the highest skills in each domain, but has one of the highest means, meaning that he/she possesses a strong background in each programming language considered, with the lowest score, a 6 (`Php`), showing good skills even in that domain.

3.3.2 Addressing multiple concern spaces

In this example, the same entity is associated with many Concern Spaces. Consider the following concern spaces:

```
C1={1, "MVC", "the weight in the Model-View-Controller", (Model, View,
Controller), 1},
```

C_2={2, "CRUDS", "Create Read Update Delete Static functionalities of an entity", (read, create, update, delete, static), 1},

C_3={3, "nonFA", "non-functional aspects of entities", (sleekdesign, loadspeed, volatility), 10},

C_4={4, "Priority", "importance of the entity", (priority), 10}.

3.3.3 Traceability issues

During the refinement process of requirements (considered entities in our general model), the child requirements derived inherit all the relations and matchings of the parent requirement from higher levels (concern spaces). Unfortunately, details of the decomposition process are not contained in the existing concern spaces. To deal with this, a new concern space is created, which contains the original information of the parent requirement, showing the relationship between the newly created sub-requirements and the original one.

3.4 Management using MultiCoS

Concerns are widely used to describe requirements. Given a problem statement, requirements will be expressed using custom concern spaces, specific to the problem at hand. These concern spaces can be used to characterize other categories of entities, not only requirements.

From the moment the problem is accepted and assumed, the related concern spaces can be defined. Here the stakeholders can be of a real help in specifying different attributes/features of the application.

3.4.1 Requirements Management using MultiCoS

Concern spaces are used to characterize the semantics of the requirements. In our example, we consider two concern spaces.

As associations are made in the model, a network of edges describe the relation of dependence existing between requirements and concerns in the concern spaces.

3.4.2 Applying the MultiCoS process to manage concern spaces exemplified in existing approaches

The versatility of MultiCoS is that it can represent matchings of concern spaces and entities existing in other approaches. Below, we give examples showing how different situations can be modeled by using concern spaces and semantic mapping function. Obtained matching vectors can be used to compute similarity coefficients, as it is shown in 3.5.

Example 3 - MultiCoS with Volere

This example shows how concern spaces can be used in requirement prioritization. Considering the prioritization analysis made in the Volere project [104], the main factors that commonly affect prioritization decisions are: Minimize Implementation Cost (development costs), Value to customer (the customer need), Time to deliver (how long it takes to implement), Ease of technical implementation

(ease of use and access required technologies), the level of difficulty in business implementation (ease to organize), Value to the business (to what extent the business benefits), external authority constrains (to abide the law). Each requirement is given a score (Priority Ranking), with respect to each of those factors, on a scale from 1 to 10.

Using our notations, we define the following concern space:

C_2={2, "prioritization", "prioritization factors for requirements", (Value to Customer, Value to Business, Minimize Implementation Cost, Ease of Implementation, Time to Implement), 10}.

Keeping the same notations for requirements (entities), and applying the semantic mapping function *f* we obtain the following matching vectors for Concern Space C_2:

$f(R1, C_2)=(2,7,3,8,3)$
$f(R2, C_2)=(8,8,5,7,6)$
$f(R3, C_2)=(7,3,7,4,5)$
$f(R4, C_2)=(6,8,3,5,9)$
$f(R5, C_2)=(5,5,1,3,7)$
$f(R6, C_2)=(9,6,6,5,4)$
$f(R7, C_2)=(4,3,6,7,6)$

3.5 Measuring semantic similarities

Two metrics will be used to measure the similarity of matching vectors in the *MultiCoS* approach [48], given a concern space *C*. The *similarity s* and the *distance d* are opposite values, the similarity index of two points *x* and *y* in an *n*-dimensional space increases when the distance between them decreases, and vice-versa. The relationship between the distance *d* and the similarity coefficient *s* are mainly given by two equations, depending of the distance being normalized or not. When the distance is normalized as a value in the [0, 1] range, the normalized distance index is also called *dissimilarity coefficient*, and the equation relating the dissimilarity (distance) and similarity is (3):

$$s(x,y) = 1 - d(x,y) \quad (3)$$

Otherwise, the reciprocal (4) is used:

$$s(x,y) = 1/d(x,y) \quad (4)$$

In order to compute semantic similarities, we follow the pattern in [18].

Let there be two entities *X, Y* in the entity space *E: X, Y* $\in$ *E;* a concern space *C*, with *n* dimensions: $C = \{c_1, \ldots, c_n\}$; and $k \in R_+$ being the maximum matching threshold for *C (max_value)*. Applying the semantic mapping function (2) to *X* and *Y*, we obtain the mapping vectors $A = f(X, C) = (a_1, \ldots, a_n)$ and $B = f(Y, C) = (b_1, \ldots, b_n)$. The distance between *X* and *Y* w.r.t. concern space *C* is given by (5):

$$d^C(X,Y) = \frac{1}{n\,k}\sum_{i=1}^{n}|a_i - b_i| \quad (5)$$

This was obtained using the Gower distance formula applied to normalized vectors A and B (with respect to k). With this, the *similarity coefficient* (3) of entities *X* and *Y* with respect to the concern *C* becomes:

$$ss^C(X,Y) = 1 - d^C(X,Y) = 1 - \frac{1}{n\,k}\sum_{i=1}^{n}|a_i - b_i| \quad (6)$$

For example, let there be two entities *X, Y* in the entity space *E, X, Y* $\in$ *E*, and a concern space *C*, with *n*=5 concerns, *C* = {c_1, c_2, c_3, c_4, c_5}, and *k=10* being the maximum matching threshold for *C*. Applying the matching function (2) to X and Y, we obtain the mapping vectors *A = f(X, C) = (8, 4, 3, 2, 1)* and *B = f(Y, C) = (6, 4, 10, 0, 4)*. Figure 28 shows the mappings (left), and the normalized distance and similarity (right), for the two entities, X and Y over *C*.

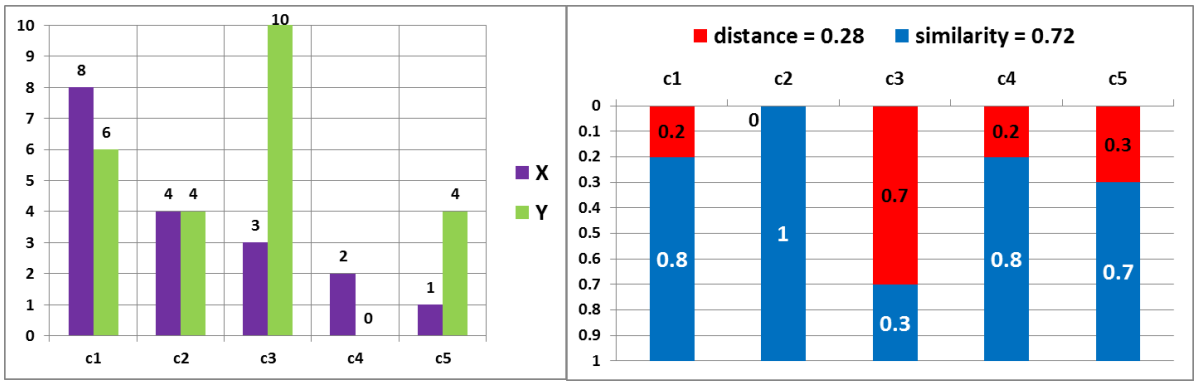


Figure 1 Mapping *and distance/similarity coefficient for entities X and Y over C*

Let us consider $d_i = |a_i - b_i|/k$, called *concern projected distance* corresponding to the *i*-th concern, and $s_i = 1 - d_i$, the corresponding *concern similarity coefficient*. For two equal values, $a_i = b_i$, we get $d_i = 0$, leading to $s_i = 1$, meaning *maximum similarity*. This is true also for two null values, $x_i = y_i = 0$, which leads to $s_i = 1$. For this reason, we will not take into account to compute these values as *concern similarity coefficients*, as we cannot allow increase the similarity of two entities based on an attribute missing in both entities. The same explanation was used in the way the Tanimoto similarity and the Jaccard index [18] were computed. The concept agreed upon is that *the absence of a feature in two objects does not provide information valuable to similarity measures* [53].

The similarity coefficient (6) can be easily generalized if we consider as basis for comparison a set of concern spaces instead of a single concern. The *multispace similarity coefficient, ms,* of two entities *X*, Y over a set of *w* concern spaces C_1, ..., C_w is the computed average of the space similarity coefficients for all *w* spaces in the concern spaces set.

$$ms(X,Y) = \frac{\sum_{j=1}^{w} ss^{C_j}(X,Y)}{w} \quad (7)$$

This gives us the degree of similarity of two entities with respect to a set of concern spaces.

Similarity coefficients (6) and (7) have values greater than or equal to 0 and less than or equal to 1, where 0 stands for *no semantic* (weakest semantic similarity) for two entities, and 1 represents the *identical semantic* (strongest semantic similarity).

3.6 Concern Management and Specification

The concern spaces used in the previous section are proving the variety of concerns that can be related to different entities. Some other will be considered here.

3.6.1 The AORA process

MultiCoS process, our proposal for requirement management using concern spaces, described in the next section, is using AORA process [12]. Next sub-sections describe AORA Plan0 processes and their use in MultiCoS.

3.6.2 Identifying Concerns in MultiCoS

Concerns can be identified using complex and intensive processes, and techniques already exist [87], but this task to identify all the concerns in a system is beyond the purpose of the current work.

3.6.3 Concern specification in MultiCoS

In the process of concern specification, the relationship of each concern to another concern has to be investigated as they are being important to be accounted of. Usually, the concerns in a concern space are not independent of one another.

Adapting the AORA concern template document to the MultiCoS approach format we obtain a meta-concern space as a document.

3.6.4 Composing concerns

Once all concerns are specified, the last step is to compose them. For each matching point identified, activity diagram.After each match point identified, the following activities can be performed in parallel: the identification of cross-cutting concerns, conflict handling, and the definition of compositional rules.

3.7 The MultiCoS process

This section, presents the MultiCoS process, our solution for requirements management using multiple concern spaces. The goal is to create a methodology allowing creation of relationships between semantically *similar* requirements from the same application or different applications, based on their semantic mapping using concern spaces. This process is used for requirements engineering.

Once such semantic connections are defined and established, they will be used to measure similarity between similar types of entities (like requirements and code modules), which, in the next step, will help in code or artifact reuse. We identified two types of questions that can be defined for which our process will be able to provide an answer:

Question type 1 As software has to be developed to implement a solution for a new problem, can we identify similar requirements, and plan to use the same resources to implement part of them?

Question type 2 As new software has to be developed to implement a solution a for new problem, can we identify similar artefacts already created for existing software, and plan to reuse them to implement part of the solution for the new problem?

MultiCoS process has three main steps:

1. Concern management
2. Entity (requirement) management
3. Similarity check

The novelty of our proposal, sketched in Figure 35, is to use these concerns and concern spaces for identification of the reusable artifacts and resources in web engineering processes. It includes newly proposed activities, such as: requirements space transformation into entities space, establishing relationships between entities and concerns (semantic mappings), generating relationships between entities in the same space or different spaces (by computing matching vectors).

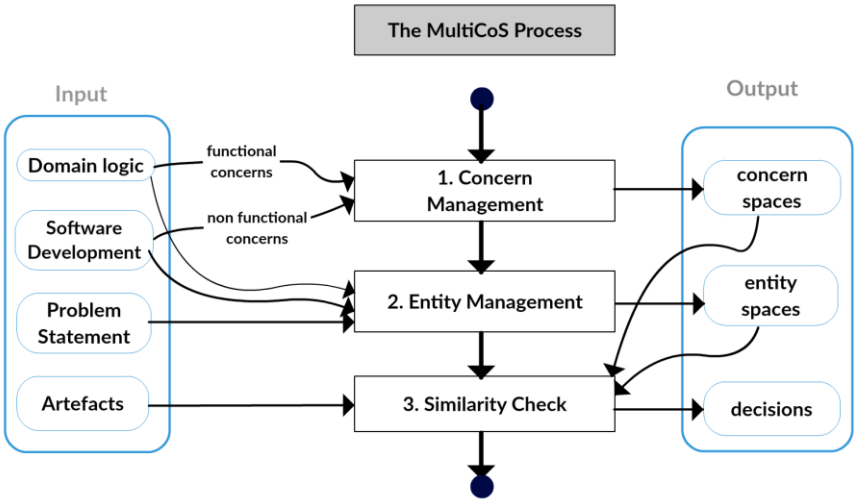


Figure 2 The MultiCoS process

3.7.1 Concern management

For the first step, concern management, the MultiCoS process uses AORA process [13], described in the previous section. Concerns are domain-specific and application-specific, as well as software development process-specific. The deliverables of this process are concern spaces and concern catalogues.

3.7.2 Entity management

The entities to be addressed in our proposal are specific to the problem domain. For the problem at hand there will be a lot of requirements, functional and non-functional, grouped into Requirements spaces. MultiCoS considers a generalization of these problem-specific requirements, using entities instead.

Entities are related to concerns. More specifically, we use concerns to compare entities in a semantic way. Each different entity type (like Requirements, code modules, actors, developers, technologies or any other artefacts or resources involved in the web development process) follows a specific transformation to become an entity. The deliverables of the entity management step are entities and entity spaces.

Entity management step has the following activities: (1) Entity gathering (2) Entity grouping into entity spaces.

3.7.3 Similarity check

This last step of the MultiCoS process deals with semantic similarities between entities with respects to concern spaces. In order to get to this desideratum, several sub steps will be performed (Figure 40): (1) select entity and concern spaces to be considered, (2) compute the matching vectors, (3) compute the similarity coefficients for entities with respects to each concern space selected, and (4) compute the multispace similarity coefficients for entities with respects to all the concern spaces selected.

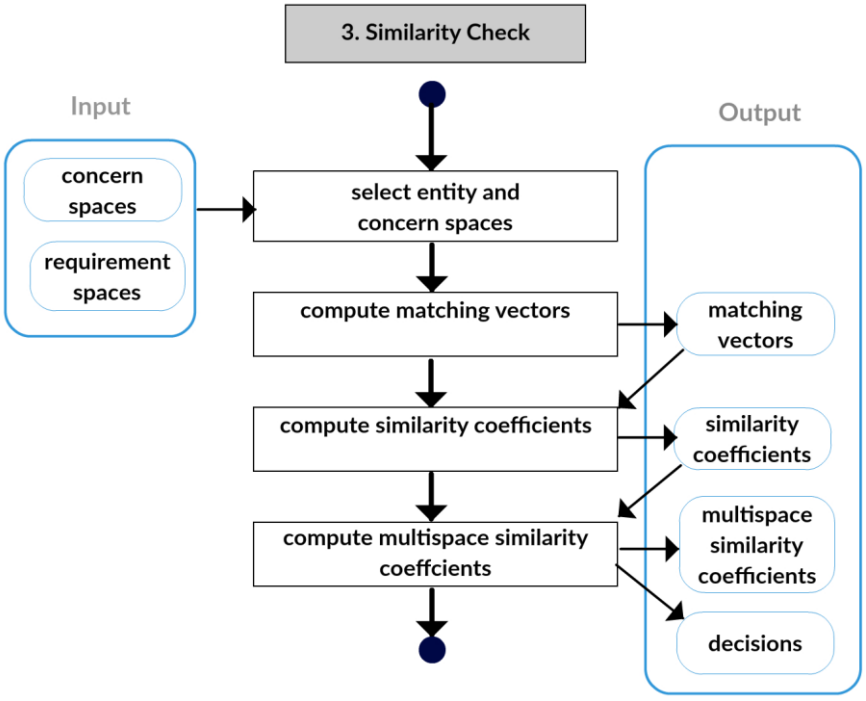


Figure 3 Investigating requirements for semantic similarities in MultiCoS approach

The multispace similarity coefficients will help in the decision of reusing artefacts, answering to the questions stated in the beginning of this section.Of course, the final decision is made by humans, but MultiCoS provides means to assist it. More examples will be presented in the next chapters, where the MultiCoS tool is presented and a Study Case is conducted.

3.8 Using MultiCoS

Semantic similarity is used to answer the two questions stated at the beginning of the previous section.

3.8.1 Similarity of some requirements from a new problem

Question type 1 is looking to identify similar requirements in a problem. We plan to decompose the requirements into sub requirements, identifying those sub requirements with very high similarity coefficients with respect to a concern space.

Having two requirements, A and B, we can decompose them into sub-requirements, up to the level of primitive data operations. In order to compare sub-requirements, we need a concern space. Once the similar sub requirements were

identified, we can decide to use the same code (or the same developer) to implement them.

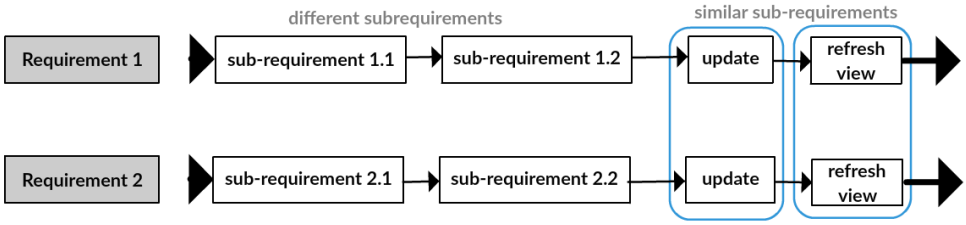


Figure 4 Identical sub requirements refining different requirements

3.8.2 Reusability of artefacts

Question type 2 is looking to identify similar requirements in different problems. Problem P_1 is already solved; during the development process, there are well-established entities (requirements, documentation, code), and the corresponding traceability links between requirements and resulting artifacts. Problem P_2 is a new problem to be solved, for which only the requirements are known. By identifying the requirements with very high similarity coefficients in P_1 and P_2 with respect to some selected concern spaces, we can use the artifacts implementing P_1 requirements to implement P_2 requirements. Figure 45 illustrated the process.

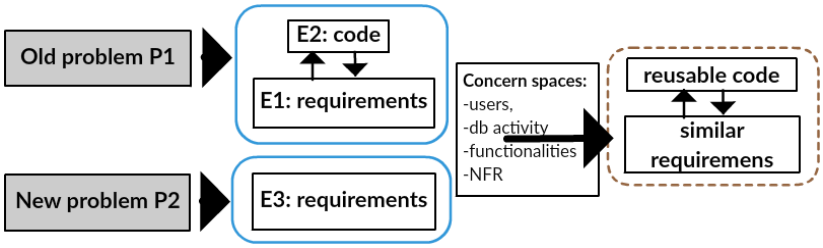


Figure 5 Reusing code to implement semantic similar requirements in MultiCoS

As an outline, we have created two matching types: (1) requirements from one problem space and requirements from another problem space, and (2) requirements and code modules from same problem space. Then, using transitivity, (3) we can create conceptual bindings between requirements from one problem space and code modules from another problem space

3.9 Summary

This chapter presents the MultiCoS process, aimed to help the identification of similar entities with respect to one or more concern spaces. MultiCoS is supporting the reuse of artefacts such as code, based on the semantic similarity of the requirements. Requirements and artefacts are considered entities. MultiCoS process considers separately and entity management as in G1.

Semantic similarity of two entities with respect to a concern space or a set of concern spaces are discussed in 3.2 and 3.5. The last step of MultiCoS process is similarity check. All these issues belong to G2. The results presented in this chapter are included in the papers published in Studia UBB, Informatica [49] and in International Journal of Computers and Technology [46].

4. The MultiCoS Tool

This section presents the MultiCoS tool, intended to help designers and business analysts to manage requirements similarities of web applications from related application domains or problem spaces. MultiCoS process uses AORE, derived from MCSA, for concern management. Original contribution of MultiCoS is entity management and the computation of similarity coefficients for pairs of entities, with respect to a concern space.

MultiCoS tool is supporting all steps of MultiCoS process described in 3.7. Before its description, first section of the chapter refers to AORE and MCSA tools.

4.1 AORE and MCSA tools

Software Intensive Systems (SIS) development faces today a number of challenges: software-based innovations driven by customer needs, increasing complexity, pressure to reduce costs, shorter development times, and higher quality demands. The aim of the MultiCoS approach presented in this thesis is to assist the SIS development. As the approach is based on Aspect-Oriented Requirements Engineering (AORE) and the Multidimensional Concern Spaces Approach (MCSA), in this introductory section of this chapter, we will investigate the AORE tools and the MCSA tool.

4.1.1 AMPLE tool suite

The Aspect-Oriented, Model-Driven, Product Line Engineering (AMPLE) project [4] developed a tool suite for aspect-oriented, model-driven requirements engineering for software product lines. The tool suite include “a set of model-driven tools and languages that allows, at the domain analysis level, to reason about the features, functional and non-functional requirements to help the definition of Software Products Library (SPL) architectures” [2].

4.1.2 AO-MD-PLE tool

Aspect-Oriented Model-Driven Product Line Engineering (AO-MD-PLE) describes product lines using a different range of models. On the model-level, aspect oriented methods are applied to define variants. Also, aspect oriented techniques are used in the transformations that leads to the production of running applications, which are generated based on the variants defined.

4.1.3 Hyper/J tool

Hyper/J is a “tool which supports the multi-dimensional separation and integration of concerns in standard Java software by using the hyperspace approach” [92]

4.1.4 HyperC# tool

HyperC# tool provides support for C# programmers to use multi-dimensional separation of concerns features. The graphical user interface of the system is developed in C# [57].

4.2 The MultiCoS Tool

The purpose of MultiCoS tool is not to replace the other existing RE tools in recording requirements or other RE related activities. The main goal of the MultiCoS tool is to enhance RE processes by supporting MultiCoS process.

4.2.1 Application Architecture

The application logic (main loop) has five steps: (1) the user submits the request, (2) the request is routed to the appropriate controller, (3) the controller interacts with the Data Model, (4) the controller invokes the View to display the result, and (5) the View is rendered in the web browser.

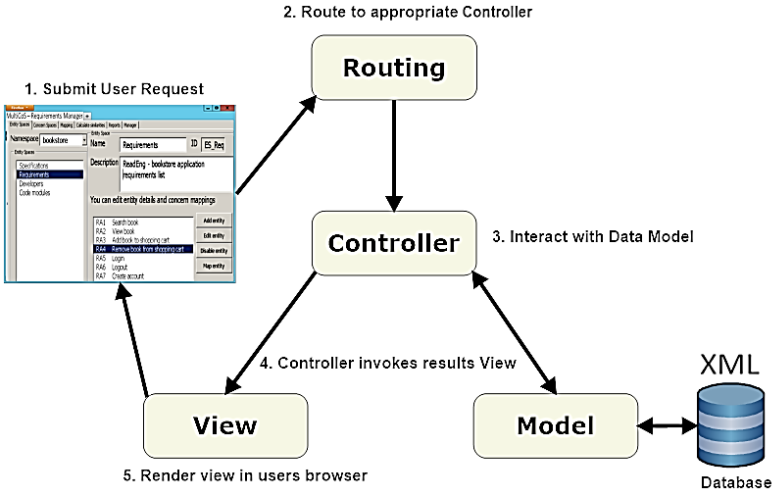


Figure 6 The MultiCoS tool architecture

4.2.2 Data Model and Storage

The MultiCoS tool can import requirements from other tools as long as the data is recorded in XML format and the schema definition is mapped to facilitate the XML file import. If the MultiCoS tool is used in capturing requirements, the Volere Template [127] is used, in a reduced form, and a list of requirements is produced and saved in XML format.

4.2.3 GUI and Reporting

The user interface of the tool helps the user to manage the MultiCoS process, from recording and loading data (e.g. requirements) related to different projects into the application data model, to processing and providing report tables of similarity indices, and exporting data in XML format.

The user interface consists of six views, managing different parts of the data model and the process: a) concern spaces, b) entity spaces, c) matching vectors, d) computing similarities, e) generating reports, and f) transition manager.

4.3 Summary

This chapter describes *MultiCoS* tool, developed to support MultiCoS process. Its design was inspired by other similar tools. It is subject of a research paper accepted by IJSEKE journal [47].

5. Using MultiCoS

The goal of this case study is to evaluate the extent to which the research goals stated in the Introduction chapter are fulfilled. The scenario is as follows:

- There is a web application already developed, denoted by A – *ReadEng*; during its development, MultiCoS process and tool were used, and the tool repository contains at least three entity spaces (Requirements Space, Documents Space, Code Space);
- There is a list of general concern spaces considered, as presented in Table 23;
- The development team needs to develop another web application, denoted by B – *MyTuneCaster*, assisted by MultiCoS tool.

Given these facts, two case studies were performed: investigation of similar requirements in the two applications, and investigation of code reuse (from application A to application B).

The last part of the chapter contains a quantitative and qualitative evaluation of MultiCoS, by comparing it with other similar approaches.

5.1 Case Study

Consider two web applications, A - *ReadEng* and B - *MyTuneCaster*. The problem statements and the initial feature lists are given below.

Problem A – ReadEng.

A bookstore sells books online using ReadEng, a web application. This application allows clients to browse the book catalog and make book orders. There are three types of users: visitor, reader, and administrator.

The visitors are able just to browse the book catalog are they don't need an account.

A visitor who creates an account becomes reader. Authenticated readers create online orders (add/remove books in the shopping cart), read book previews and book excerpts, consult their orders' history, manage their account profile.

The administrators manage the bookstore content, reject or approve readers' orders, and generate reports referring to: books sold, readers profiles, and financial status of the bookstore.

Our intent is to develop ReadEng application which should allow users (visitors, readers, administrators) to collaborate and interact in order to achieve their own goals.

Problem B – MyTuneCaster.

An Audio Library online application offers users the possibility to search and listen audio content available in the library. There are three types of users: visitor, client, and administrator.

The visitors are able to browse the audio library, play regular audio content and play excerpts of premium content. The visitors are displayed ads.

Visitors creating an account become clients. Authenticated clients have unrestricted access to all audio content (regular and premium) and they are not displayed ads. Visitors manage online playlists, and their account profiles. Clients are charged a membership fee.

Administrators manage the audio library content, clients' accounts, and generate different reports: audio files played, clients profiles and financial status.

Usually, requirements are gathered and specified from the feature lists. They express features in a more structured and precise way. The requirements from each application will be grouped in two different Requirements Spaces.

For simplicity, we do not list the complete set of requirements, but we focus on similar ones. At first sight, users in both applications can log in/out, search for content, see/view a product, create orders of products, and buy them. Table 24 contains a subset of those similar requirements from the two problems considered. Their identifiers are of the form **Rxy**, where **R** stands for **R**equirement, **x** denotes the problem, and **y** is a serial number, problem-specific.

Table 1 Similar requirements in problems *ReadEng* and *MyTuneCaster*

Problem P-A –*ReadEng*		Problem P-B - *MyTuneCaster*	
Id	Requirement	Id	Requirement
RA1	Search book	RB1	Search song
RA2	View book	RB2	Listen song
RA3	Add book to shopping cart	RB3	Add song to playlist
RA4	Remove book from shopping cart	RB4	Remove song from playlist
RA5	Login	RB5	Login
RA6	Logout	RB6	Logout
RA7	Create account	RB7	Create account
RA8	Logged user can buy products in shopping cart	RB8	Logged user can listen songs in playlist

Our goal is to show how MultiCoS can be used in two areas: (a) requirements similarity and (b) the reuse of code modules based on similarity of requirements. Next two sections address these problems.

5.2 Using MultiCoS to investigate requirement similarity

In what follows the steps of MultiCoS are applied to our example applications to investigate the similarity of requirements.

5.2.1 Managing concerns

While there are several concern spaces to cover regular attributes specific to entities in generic software systems, other concern spaces are specific to almost all web applications. In this study case we will make use of concern spaces suitable for requirements. They are given in Table 25.

It is important to note that concerns (and concern spaces) are independent of a particular application. They are used to compare entities (in our case requirements).

Table 2 Samples of Concern Spaces used in MultiCoS

Id	Space Name	Concerns
C1	Type of user:1	Guest, user, superuser, admin
C2	Information retrieval:1	Application, hostmachine, user, database
C3	Prioritization:10	Time to Implement, Ease of Implementation, Minimize Implementation Cost, Value to Business, Value to Customer
C4	DB Action type:1	Create, Read, Update, Delete, Static
C5	Navigation type:1	Hyperlink, Header, Post, Get, State
C6	App type:10	Social, Ubiquitos, Eshop, Integrated system, Multimedia
C7	Server Side Language:1	Python, Php, ASP, AJAX, MySQL, MS-SQL, Perl, .NET, C#, Ruby
C8	Data format:1	Database, text, CSV, HTML, UML, XML, richformat, DOM, spreadsheet,
C9	Activity details:1	add items to list, buys, view details, logges in, logges out, remove item from list
C10	MVC:10	Model, View, Controller
C11	NonFunctional requirements:10	sleekdesign, loadspeed, volatility, security

5.2.2 Managing requirements

Each problem has its own namespace: **bookstore** for *ReadEng* and **tunecast** for *MyTuneCast*.

5.2.3 Similarity check

After the requirements spaces are stored, the similarity check is performed as described in 3.7.

5.2.3.1 Computing matching vectors

First, we have to select concern spaces of interest for our study: C1, C2, C3, C4, C9, C10 and C11. Next, for each requirement ***r*** and each concern ***c*** in a concern space, matching vectors are computed by assigning a matching value to each pair (***r, c***), as in Figure 57 (Mapping View). The value assigned is expressing the beliefs of the human decider regarding the weight (importance, matching) of the requirement ***r*** with respect to concern ***c***.

As an example, for requirement RA4 (remove book) and the concern space C4 with maxval = 1 (Table 25), the concerns (*read*, *update*, *delete*) are assigned a maximum value of 1, because these operations are all needed to complete the functionality of the requirement, while the concern (*create*) is not needed at all, so it will get a matching value of 0.

This sub step is repeated for all requirement spaces and all concerns selected. The results are stored as matching vectors in MultiCoS data model.

5.2.3.2 Computing Similarity Coefficients and Multispace Similarity Coefficients

First, we select the requirements (as entities) from the two namespaces considered and the concern spaces of interest in our study (shown in Table 26). Then, by pressing **generate coefficients** button, the multispace similarity

coefficients are computed and displayed, as shown in Figure 62. As an intermediate step, the grid below displays space similarity coefficients for the selected row in the grid above (requirement RA4 in our example).

As an illustrative example, the process of computing space similarity coefficients and multispace similarity coefficient for the pair (RA8, RB8) of requirements is detailed below. Table 26 shows matching vectors and space similarity coefficients for all concern spaces selected, and the multispace similarity coefficient computed by equation (7).

Table 3 The multispace similarity coefficient for requirements RA8 and RB8

Concern Space	Requirement RA8	Requirement RB8	Space similarity coefficient
C1	f(RA8,C1) = (0,1,1,0)	f(RB8,C1) = (0,1,1,0)	ss1 = 1
C2	f(RA8,C2) = (0,1,0,1)	f(RB8,C2) = (0,1,0,1)	ss2 = 1
C3	f(RA8,C3) = (5,10,9,9,8)	f(RB8,C3) = (10,9,9,8,9)	ss3 = 0.84
C4	f(RA8,C4) = (1,0,1,0,0)	f(RB8,C4) = (1,1,0,0,0)	ss4 = 0.33
C9	f(RA8,C9) = (0,0,0,1,0,1)	f(RB8,C9) = (0,0,1,0,1,1)	ss9 = 0.25
C10	f(RA8,C10) = (8,0,10)	f(RB8,C10) = (5,8,10)	ss10 = 0.37
C11	f(RA8,C11) = (7,5,2,10)	f(RB8,C11) = (9,10,6,5)	ss11 = 0.60
		multispace **similarity**	**ms=0.627**

5.2.3.3 Investigating semantic similarity

The similarity matrix shown in Figure 62 reveals important (and informed) information about all pairs (RA, RB) of requirements considered. Considering first its main diagonal (which corresponds to the pairs listed in Table 24), one observe that the similarity coefficients are closer to the maximum value: 90, 90, 95, 95, 100, 100, 100, 63. This means that our initial guess was correct in almost all cases. The pair (RA8, RB8) has the weakest similarity.

Second, there are other pairs, not included in Table 24, which have strong similarities. Table 27 shows similarity coefficients in decreasing order of their values, including those already mentioned, and can serve as a guide in selecting candidate artefacts (code modules implementing requirements from problem A) to be reused in developing a web application for problem B.

Table 4 Strong semantic similarities in requirements spaces A and B

Similarity coefficient	Entity Space A		Entity Space B	
	Id	**Requirement**	**Id**	**Requirement**
100	**RA5**	Login	**RB5**	Login
100	**RA6**	Logout	**RB6**	Logout
100	**RA7**	Create account	**RB7**	Create account
95	**RA3**	Add book to shopping cart	**RB3**	Add song to playlist
95	**RA4**	Remove book from shopping cart	**RB4**	Remove song from playlist

5.3 Using MultiCoS to investigate reuse of code modules

Considering code modules as concerns, traceability of requirements to code can be stored as a matching value from an entity to a concern. Using MultiCoS, we

can create a matching between a requirement space and a *code modules* concern space, both in the same problem space. This is possible, usually, post-factum, i.e. after the development process is completed.

5.3.1. Concern management

The code modules are recorded as concerns, in the problem space of the application they belong.

5.3.2. Entity management

No other entities are needed. Of course, if we need a more detailed view, requirements can be decomposed into sub-requirements and the corresponding links can be recorded.

5.3.3. Similarity check

In this case, similarity check step can be used to investigate the similarity of requirements with respect to a concern space representing code modules.

5.3.3.1 Computing matching vectors

Consider an entity (requirement) ***r*** and a concern (code module) ***cm***. The matching value ***mv(r,cm)*** has here the following semantics: the degree to which code module ***cm*** implements the requirement **r**. Of course, a requirement is implemented in one or many code modules, and in the same time a code module can represent the implementation of one or more requirements. This way, the matching value is a measure of traceability of requirements to code and vice versa.

5.3.3.2 Computing Similarity Coefficients and Multispace Similarity Coefficients

This sub step is not needed here. We use similarity coefficients and the multispace similarity coefficients already computed in 5.2.3.2 (Figure 62). Note that $ms(RA_i, RB_j) = ms(RB_j, RA_i)$.

5.3.3.3 Investigating semantic similarity

This sub step is performed using Mapping View (Figure 67). The process uses the following input data: (a) multispace similarity coefficients computed in 5.2.3.2 $ms(RB_j, RA_i)$ and (b) matching values $mv(RA_i, cm_k)$ assigned in 5.3.3.1. Its meaning is that a greater similarity and a greater matching value will lead to a greater probability of reusing cm_k to implement RB_j.

The grid in Figure 67 contains (from left to right): multispace similarity coefficients $ms(RB_j, RA_i)$ and matching values $mv(RA_i, cm_k)$. In other words, it defines a "logical", "transitive" mapping from a code module cm_k to a requirement RB_j. The decision to reuse cm_k in the implementation of RB_j must take into account (a) the value of $ms(RB_j, RA_i)$ and (b) the value of $mv(RA_i, cm_k)$.

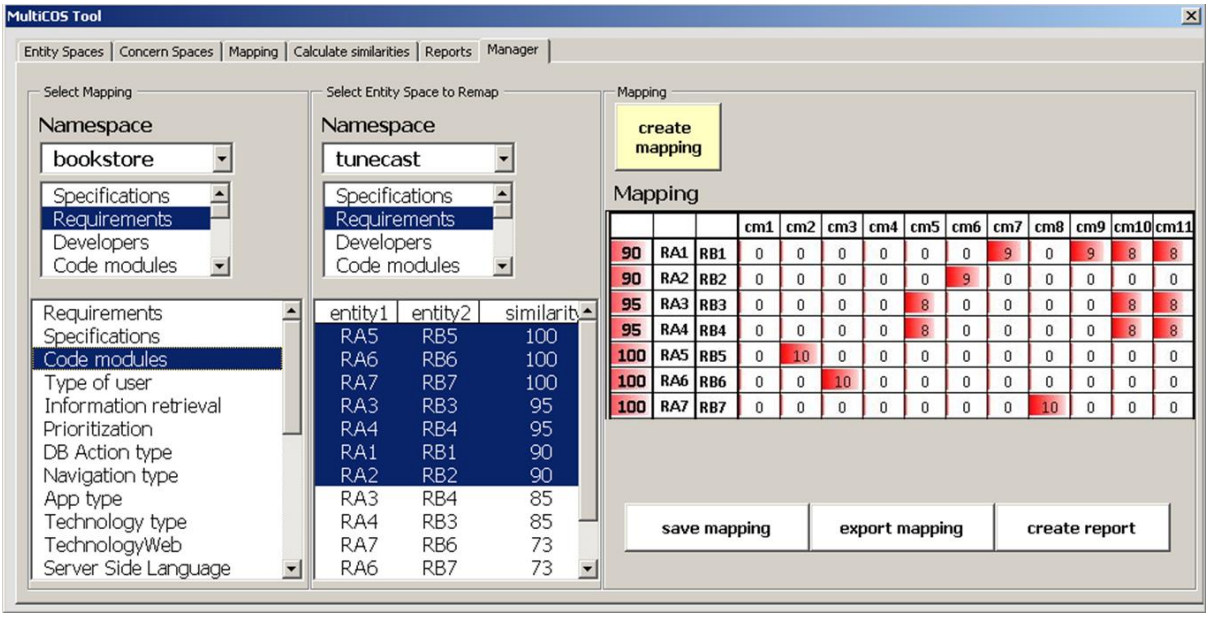
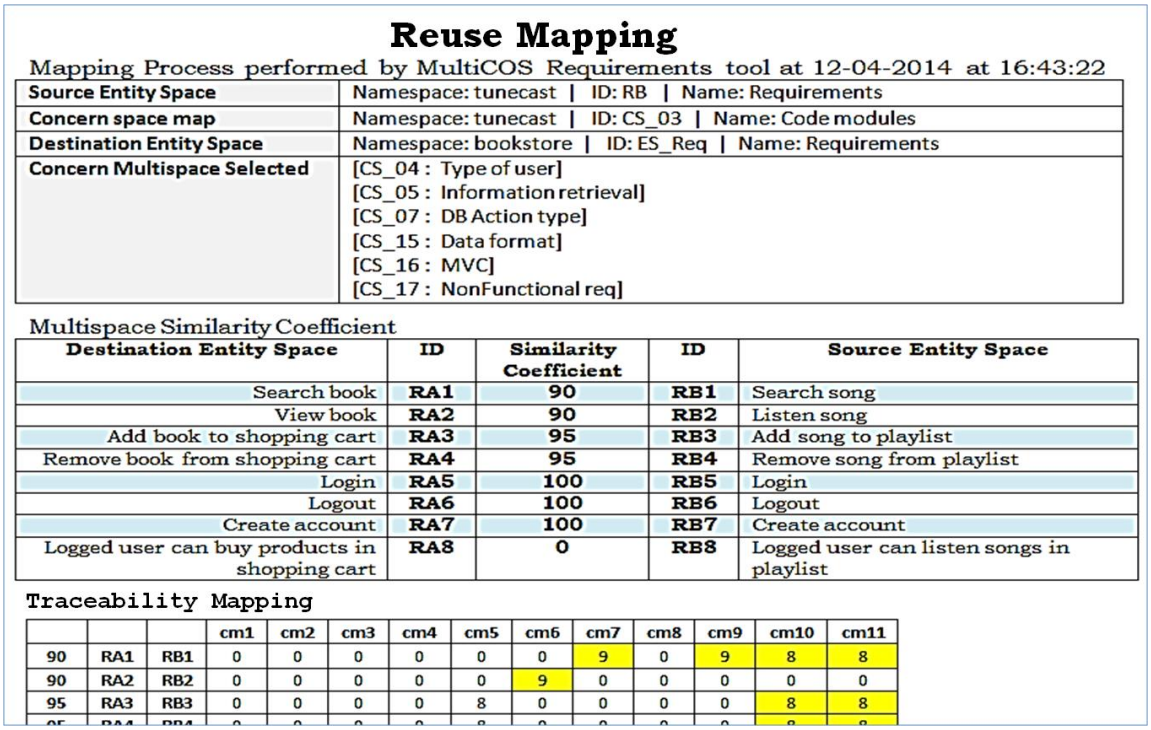


Figure 7 Mapping manager in MultiCoS tool, with `create report` option



Reuse Mapping

Mapping Process performed by MultiCOS Requirements tool at 12-04-2014 at 16:43:22

Source Entity Space	Namespace: tunecast | ID: RB | Name: Requirements
Concern space map	Namespace: tunecast | ID: CS_03 | Name: Code modules
Destination Entity Space	Namespace: bookstore | ID: ES_Req | Name: Requirements
Concern Multispace Selected	[CS_04 : Type of user] [CS_05 : Information retrieval] [CS_07 : DB Action type] [CS_15 : Data format] [CS_16 : MVC] [CS_17 : NonFunctional req]

Multispace Similarity Coefficient

Destination Entity Space	ID	Similarity Coefficient	ID	Source Entity Space
Search book	RA1	90	RB1	Search song
View book	RA2	90	RB2	Listen song
Add book to shopping cart	RA3	95	RB3	Add song to playlist
Remove book from shopping cart	RA4	95	RB4	Remove song from playlist
Login	RA5	100	RB5	Login
Logout	RA6	100	RB6	Logout
Create account	RA7	100	RB7	Create account
Logged user can buy products in shopping cart	RA8	0	RB8	Logged user can listen songs in playlist

Traceability Mapping

			cm1	cm2	cm3	cm4	cm5	cm6	cm7	cm8	cm9	cm10	cm11
90	RA1	RB1	0	0	0	0	0	0	9	0	9	8	8
90	RA2	RB2	0	0	0	0	0	9	0	0	0	0	0
95	RA3	RB3	0	0	0	0	8	0	0	0	0	8	8

Figure 8 Reuse Report generated by MultiCoS tool.

A report can be generated here, as shown in Figure 68 and Table 28, containing information on the code modules mapping. The mapping details provided can assist the user in deciding over reusing code modules from the existing application into the new one.

We can read from this report that, basically, the "`Login`" requirements in the two applications are identical, which is a trivial situation. Still, we can reuse the code related to the login process However, the code will have to be adjusted to fit in the new application profile, but as long as we are using the existing

application as the only source for similar code, and the same framework, the other libraries should be common, and the code adjustments will be minor.

Table 5 Detailed results of the transitivity mapping from MultiCoS report

Requirement	Code module	Details
RB5: Login		
Matching value: **10**	**cm2**: doLogin()	user login **file**: Login.php, index.php, User.pho
RB6: Logout		
Matching value: **10**	**cm3**: doLogut()	user logout **file**: Login.php, index.php, User.pho
RB7: Create account		
Matching value: **10**	**cm8**: createAccount()	creates an user account **file**: register.blade.php, main.blade.php, User.pho
RB2: Listen song		
Matching value: **9**	**cm6**: bookStore()	displays book content **file**: bookreader.php
RB3: Add song to playlist		
Matching value: **8**	**cm10**: guestView	creates a view for guest **file**: bookreader.php
Matching value: **8**	**cm11**: userView	creates a view for an autenticathed user **file**: bookreader.php
RB4: Remove song from playlist		
Matching value: **8**	**cm10**: guestView	creates a view for guest **file**: bookreader.php
Matching value: **8**	**cm11**: userView	creates a view for logged user **file**: bookreader.php
RA1: Search song		
Matching value: **9**	**cm7**: search()	searches database for song **file**: dbclass.php
Matching value: **9**	**cm9**: displayResults	creates a view creates a view with the search() function response **file**: results.blade.php , index.php
Matching value: **8**	**cm10**: guestView	creates a view for guest **file**: guest.blade.php, index.php
Matching value: **8**	**cm11**: userView	creates a view for logged user **file**: user.blade.php, index.php

So far, we had no surprises as we could expect to get a high or a very high semantic similarity coefficient for requirements implementing the same identical operations in both applications, operations such as login, logout, create account. This only validates the hypothesis regarding the reliability of using the MultiCoS method in order to find out that identical requisitions in different problem spaces are similar semantic entities. Also, an expected outcome of using the MultiCoS tool is that similarity coefficients computed for identical requirements over a particular multispace are of maximum value.

Such examples are given by following pairs of requirements, as indicated in Figure 68:

1. (RA3 – "`Add book to shopping cart`", RB3 – "`Add song to playlist`") with a semantic similarity coefficient of 95.
2. (RA4 – "`Remove book from shopping cart`", RB4 – "`Remove song from playlist`") with a semantic similarity coefficient of 95.

Investigating the Transitivity Map from Table 28, for this two pairs, we notice that the code modules available to be reused (`bookstore.php`) contain on one hand, code snippets that had to be changed, as they are being specific to online book readers, but, on the other hand, they contain code that can be reused with small adjustments, like, the function calls, user type related specific limitations, the design, the structure of the file and other dependencies (*Appendix 3F*). The code changes suggested are proposed below.

3. (RA1 – Search Book, RB1 – Search Song) with a semantic similarity coefficient of 90.
4. (RA2 – View Book, RB2 – Listen Song) with a semantic similarity coefficient of 90.

Once this investigation is finished and the decision to reuse code is performed, the code from *ReadEng* application can be reused and adapted to fit the requirements of *MyTuneCast* application.

5.3 MultiCoS Evaluation

This section refers to the evaluation of MultiCoS process from quantitative and qualitative point of view. The quantitative evaluation is done by applying several metrics taken from different AORE approaches. We will than compare the results computed for the selected approaches. As here we will just test the amount of resources and artefacts quantitatively; we will use in testing requirements specifications relevant to the "Testbed for Aspect-Oriented Software Development project (TAO)" [118] environment. This project's main purpose was to define a framework for testing AOSD techniques.

5.3.1 A Testbed for AOSD

The following testbed was also used as part in the other evaluations. The TAO[1] project was a large project, involving several research teams and it was conducted by a team in Lancaster University with the purpose of assessing the AOSD.

5.3.2 MultiCoS quantitative evaluation

The quantitative analysis of AORE approaches in the TAO project is presented here. As investigated and stated in [12], "the metrics used were originally proposed to RE for Object-Oriented approaches and then were adapted to AOSD.".

In order to measure the product, the process and the individual skills, certain metrics are used in Software Engineering and artefacts, characteristics; resources are measured [36]. Table 31 displays the values of metrics.

MultiCoS and AORA have the lowest value of Concern Diffusion over Artefacts when compared with the other two approaches, as their primary artefacts are Concerns. The MultiCoS and AORA approaches, have zero value for the cohesion metric. This is the case when concerns are the primary artefacts of the approaches.

As a conclusion, MultiCoS obtained a good result at the quantitative evaluation compared to other approaches, with a value of 3.2 operations per artefact. This is due to the high level of abstraction of the process.

[1] http://www.comp.lancs.ac.uk/research/projects/project.php?pid=215

Table 6 Experimental values of metrics from Table 29 for AORE approaches, adapted from [12]

Attributes	Metrics	MultiCoS	EA-Miner	Multi-Dimensional CORE	AOVGraph	AORA
SoC	Concern Diffusion over Artefacts (CDA)	2	3	1	2	1
	Concern Diffusion over Operations (CDO)	5	23	11	12	7
	Concern Diffusions over LOC (CDLOC)	8	11	14	10	9
Coupling	Coupling Between Artefacts (CBA)	4	3	10	3	6
Cohesion	Lack of Cohesion in Operations (LCOO)	0	23	0	24	0
Size	Artefacts (Artefacts)	20	15	18	22	22
	Vocabulary Size (VS)	9	3	2	6	9
	Weighted OperatiperArtefact (WOA)	Average: 3.2	Average: 6	Average: 4.4	Average: 8.3	Average: 3

5.3.3 MultiCoS quality evaluation

The evaluation of MultiCoS quality is guided on the rules used in measuring requirements engineering methodologies proposed in [21]. The qualities measured are: uniformity in treating requirements, assistance in decision and trade-off resolution, validation and verification, crosscutting requirements management, mapping requirements to other artefacts. The quality evaluation of MultiCoS is outlined in Table below.

Table 7 MultiCoS quality control

Quality	Acquired	Reason
Traceability	Yes	Entity data model supports traceability of requirements from their originator. Using code modules as concern spaces, Relation class stores links from entities to code modules.
Modularization of crosscutting requirements	Yes	Requirements are considered entities which can be grouped into entity spaces. Each problem has its own requirement space.
Identification of crosscutting requirements	Yes	Similar requirements in different applications are detected by computing multispace similarity coefficients
Composition of crosscutting requirements	Partly	There is a reversible decomposition mechanism described in Figure 44.
Conflict management	No	Not covered yet
Tool support	Yes	MultiCoS - Web based Tool
Validation	Yes	Based on case studies
Mapping crosscutting requirements to later development stages	Partly	Crosscutting requirements candidate to be mapped in later stages can be grouped in separate entity spaces, to be processed later.
Maturity	No	Not tested in large projects

5.4 Summary

This chapter is dedicated to validate MultiCoS process and tool. The two case studies performed, as well as the evaluation made in Section 5.3 prove that the proposed method and tool are useful in RE for web applications.

The results are disseminated in a papers published in Studia UBB, Informatica [48].

6. Conclusions and further work

6.1 General conclusions

In this thesis, a new AORA type methodology was proposed for web requirement engineering. The proposed process, called MultiCoS (Multiple Concern Spaces) has three steps: concern management, entity management, and similarity check.

Our proposal is applied in Requirement Engineering for web applications. In MultiCoS, requirements are considered entities. MultiCoS is used to measure the similarity between requirements from the same application or from requirements from different applications. Also, MultiCoS allows us to define traceability of requirements to code, by considering code modules as a concern space. MultiCoS process is supported by a MultiCoS tool, a web-based application designed with the goal of (a) maintaining a repository of data regarding concern spaces, entity spaces, and similarity measures; (b) assisting the user in performing similarity studies, by providing reporting and graphical means.

6.2 Limitations

The primary limitation of our work is due to the novelty of the AORA methods in conjunction with separation of concerns principle. Also, some concrete issues remain to be solved. All in all, this is, in our opinion, a good starting point in using separation of concerns in requirement engineering of web applications.

6.3 Further work

Further research is aiming to: (a) improve MultiCoS process; (b) improve the tool and (c) use MultiCoS in other fields.

Also, research should be done to investigate and select methodically the right concern spaces fit for certain types of entities or scenarios. An important contribution in this direction was done by Poshyvanyk et all in [97], and any related work should take into account their results.

Another direction worth investigating is to attempt reverse engineering by including concern related details into the source code of the web application. Such an approach is described by Marcus and Rajlich in [78], where we can find outlined the challenges they faced and the directions worth to follow.

The effect of using MultiCoS in web RE and during the software development process will be part of future studies. Referring to tool improvement, we mention import/export functions from the proprietary format to different existing XML formats used to record requirements in UML and Eclipse SysML.

Referring to other application areas, further work will include the use of MultiCoS for the development of medical applications. As an example, it can be used to investigate similar medical conditions in patients; also, historical and political circumstances can be modeled as entities and compared using this method. Another use can be in image analysis of visual motion detectors, weather phenomena, recruiting in human resources or in biomedical imaging.

References

[1] I. Sommerville, Software Engineering, 7ed: Addison-Wesley, 2004.

[2] S. Friedental, A. Moore and R. Steiner, A Practical Guide to SysML: The Systems Modeling Language, 3rd edition, Waltham: OMG, 2015.

[3] R. Chitchyan, A. Rashid, P. Rayson and R. Waters, "Semantics-based Composition for Aspect-Oriented Requirements Engineering," in *AOSD IEEE Conference*, 2007.

[4] H. Ossher and P. Tarr, "Hyper/J: Multidimensional Separation of Concerns for Java (Demonstration Proposal).," IBM Research, 2000.

[5] R. Young, The Requirements Engineering Handbook, Norwood, MA, USA: Artech House, Incorporated, 2003.

[6] D. Howe, "Free On-line Dictionary of Computing," 1995. [Online]. Available: http://dictionary.reference.com/browse/requirements engineering. [Accessed 10 10 2013].

[7] P. Parker, McGraw-Hill Dictionary of Scientific & Technical Terms, 6E, The McGraw-Hill Companies, Inc., 2003.

[8] S. Murugesan, Y. Deshpande, S. Hansen and A. Ginige, "Web Engineering: A New Discipline for Development of Web-based Systems," in *Proceedings of theFirst ICSE Workshop on Web Engineering, International Conference on Software Engineering*, 1999.

[9] A. Al-Rawas and S. Easterbrook, "Communication problems in requirements engineering: a field study," in *In Proceedings of the First Westminster Conference on Professional Awareness in Software Engineering*, London, 1996.

[10] Y. Deshpande and S. Hansen, "Web engineering: creating a discipline among disciplines.," *IEEE Multimedia,* no. April - June, pp. 82-87, 2001.

[11] L. Constantine and L. Lockwood, Software for Use: A practical Guide to the Models and Methods of Usage-Centered Design, Reading, MA: Addison-Wesley, 1999.

[12] A. Jaaksi, "Our cases with use cases," *Journal of Object-Oriented Programming,* vol. 10, no. 9, pp. 58-65, 1998.

[13] J. Leite, G. Rossi, F. Balaguer and V. Maiorana, "Enhancing a requirements baseline with scenarios," *Requirements Engineering,* vol. 1, no. 4, pp. 184-198, 1998.

[14] A. Durán, B. Bernárdez, A. Ruiz and M. Toro, "A Requirements Elicitation Approach Based on Templates and Patterns," in *International Workshop on Requirements Engineering (WER'99)*, Buenos Aires (Argentina), 1999.

[15] D. Rosenberg and K. Scott, "Use CASE Driven Object Modelling with UML," Addison Wesley, 1999.

[16] L. Chung, B. A. Nixon, E. Yu and J. Mylopuolos, "Non-Functional Requirements in Software Engineering, Volume 5," in *International Series in Software Engineering*, Springer, 1999.

[17] L. Cysneiros, J. Leite and J. Neto, "A Framework for integrating non-functional requirements into conceptual models," in *Requirements Engineering, Vol. 6, 2*, London, Springer, 2001, pp. 97-155.

[18] P. Botella, X. Burgués, X. Franch, M. Huerta and G. Salazar, "Modeling Non-Functional Requirements," in *Applying Requirements Engineering - JIRA Catedral publicaciones*, 2001.

[19] C. Cachero and N. Koch, "Conceptual Navigation Analysis: a Device and Platform Independent Navigation Specification," *2 nd International Workshop on Web-oriented Software Technology (IWWOST'02),* June 2202.

[20] D. Schwabe, G. Rossi and S. Barbosa, "Systematic Hypermedia Application Design with OOHDM," *In Proceedings of the Seventh ACM Conference on Hypertext. Bethesda, Maryland, United States, HYPERTEXT '96. ACM, New York, NY,* pp. 116-128, 16-20 March 1996.

[21] S. Ceri, P. Fraternali and A. Bongio, "Web Modeling Language (WebML): a Modeling Language for Designing Web Sites," in *WWW9 Conference*, Amsterdam, 2000.

[22] P. Greenspun, hilip and Alex guide to Web publishing, Morgan Kaufmann; 1st edition, 1999.

[23] J. Conklin, "Hypertext: An Introduction and Survey," *IEEE Computer,* vol. 20, no. 9, pp. 17-41, 1987.

[24] N. Koch, Software Engineering for Adaptive Hypermedia Applications. PhD thesis, Muynich, Germany: Ludwig-Maximilians-University, 2000.

[25] O. De Troyer and C. Leune, "WSDM: A User-Centered Design Method for Web Sites. Computer Networks and ISDN systems," in *Proceedings of the 7th International World Wide Web Conference, pp. 85 - 94, Publ. Elsevier*, Brisbane, Australia, 1998.

[26] L. Baresi, F. Garzotto and P. Paolini, "Extending UML for Modeling Web Applications," in *Proceedings of the 34th Hawaii International Conference on System Sciences*, 2001.

[27] H. Lee, C. Lee and C. Yoo, "A Scenario-Based Object Oriented Methodology for Developing Hypermedia Information Systems," in *In Proceedings of the Thirty-First Annual Hawaii international Conference on System Sciences-Volume 2. January 06 - 09*, 1998.

[28] J. Yoo and M. Bieber, "A Systematic Relationship Analysis for Modeling Information Domains," 1998. [Online]. Available: http://citeseerx.ist.psu.edu/viewdoc/download?doi=10.1.1.35.2058&rep=rep1&type=pdf. [Accessed 5 5 2013].

[29] C. Cachero, Una extensión a los métodos OO para el modelado y generación automática de interfaces hipermediales. PhD Thesis, Alicante, Spain: Universidad de Alicante, 2003.

[30] D. Ross, J. Goodenough and C. Irvine, "Software Engineering: Processes, Principles, and Goals," in *IEEE Computer*, IEEE Computer Society, 1975, pp. 17-27.

[31] P. Tarr, H. Ossher, H. Harrison and S. Sutton Jr, "N Degrees of Separation: Multi-Dimensional Separation of Concerns," *In 21th International Conference on Software Engineering (ICSE'99),* pp. 107-119, 1999.

[32] ACM, "Special Issue on Aspect-Oriented Programming," ACM Press, 2001.

[33] A. Rashid, A. Moreira and J. Araújo, "Modularization and Composition of Aspectual Requirements.," in *In 2nd Aspect-Oriented Software Development Conference (AOSD'03)*, Boston, USA, 2003.

[34] S. Clarke and R. Walker, "Composition Patterns: An Approach to Designing Reusable Aspects," in *23rd International Conference on Software Engineering, (ICSE'01)*, Ontario, 2001.

[35] B. Tekinerdogan, "ASAAM: Aspectual Software Architecture Analysis Method," in *4th Working IEEE/IFIP Conference on Software Architecture (WICSA'04)*, Oslo, 2004.

[36] A. Rashid, A. Moreira, J. Araújo, B. Tekinerdogan, E. Baniassad and P. Clements, "Early Aspects," 2006. [Online]. Available: http://www.early-aspects.net/.

[37] A. Moreira, A. Rashid and J. Araújo, "Multi-dimensional Separation of Concerns in Requirements Engineering," in *In 13th Requirements Engineering Conference (RE'05),*, Paris, 2005.

[38] C. Gal-Chis, "A Multi-Dimensional Separation of Concerns of the Web Application Requirements," *Studia Universitatis Babes-Bolyai, Series Informatica,* vol. LVIII, pp. 29-40, 2013.

[39] J. C. Gower, "Measures of similarity, dissimilarity, and distance," in *Encyclopedia of statistical sciences*, New York, S. Kotz. , Wiley, 1985, pp. 397-405.

[40] R. Schmidt, "Chapter 7 – Understanding Software Requirements," in *Software Engineering: Architecture-driven Software Development*, Morgan Kaufmann, 2013, pp. 121-137.

[41] R. Schmidt, "Chapter 8 – Software Requirements Analysis Practice," in *Software Engineering: Architecture-driven Software Development*, Morgan Kaufmann, 2013, p. 139–158.

[42] A. M. Pitangueira, R. S. Pitangueira Maciela and M. Barros, "Software requirements selection and prioritization using SBSE approaches: A systematic review and mapping of the literature," *Journal of Systems and Software,* vol. 103, p. 267–280, 2015.

[43] L. Silva, E. Huzita and T. Tait, "Comparing approaches in AORE through ISO/IEC 9126," in *New Trends in Software Methodologies, Tools and Techniques*, Québec, Canada, 2006.

[44] E. W. Dijkstra, A discipline of programming, Prentice Hall, 1976.

[45] R. Filman, T. Elrad, S. Clarke and M. Aksit, Aspect-Oriented Software Development, Addison-Wesley, 2005.

[46] C. Gal Chis, "Modeling Concern Spaces Using Multi Dimensional Separation of Concern," *International Journal of Computers and Techology,* vol. 11, no. 2, pp. 2302-2313, 2013.

[47] Z. C. Hantelmann Angela, "Adding Aspect-Oriented Programming Features to C#.NET by Using Multidimensional Separation of Concerns (MDSOC) Approach," *Journal of Object Technology,* vol. 5, no. 4, pp. 59-83, May-June 2006.

[48] I. S. S. Brito, Aspect-Oriented Requirements Analysis - PHd Thesis, Lisboa: Universidade Nova de Lisboa, Faculdade de Ciência e Tecnologia, 2008.

[49] IEEE, IEEE Recommended Practice for Architectural Description of Software-Intensive Systems, IEEE, 2000.

[50] C. Gal-Chis, "Using Concern Spaces to Measure Requirements Similarities," *Studia Universitatis Babes-Bolyai - Series Informatica,* vol. LX, no. 1, pp. 35-46, 2015.

[51] J. Robertson and S. Robertson, "Volere Requirements Specification Template," Atlantic Systems Guild, London, UK, 2012.

[52] S.-H. Cha, "Comprehensive Survey on Distance/Similarity Measures between Probability Density Functions," *International Journal Of Mathematical Models And Methods In Applied Sciences,* vol. 1, no. 4, pp. 300-307, 2007.

[53] B. Nuseibeh and S. Easterbrook, "Requirements Engineering: A Roadmap," in *22nd International Conference on Software Engineering (ICSE'00)*, Limerick, Ireland, 2000.

[54] M. Alferez, V. Amaral, J. Araújo, P. Greenwood, U. Kulesza, R. Mateus and A. Moreira, "D1.5 - Tool suite for aspect-oriented, model-driven requirements engineering - AMPLE Project," Lisboa, Portugal, Universidade Nova de Lisboa, 2009 M. Alférez, et al.

[55] AMPLE, "Deliverable survey of state of the art AMPLE Aspect-Oriented, Model-Driven, Product Line Engineering".

[56] H. Ossher and P. Tarr, Hyper/J User and Installation Manual, IBM Research, 2000.

[57] Volere, "Volere Requirements Specification Template," [Online]. Available: http://www.volere.co.uk/template.htm. [Accessed 5 5 2013].

[58] C. Gal-Chis, "Web Application Methodologies with RE Tools Support," *International Journal of Computers and Technology,* vol. 11, no. 2, pp. 2314-2320, 2013.

[59] C. Gal-Chis and B. Pârv, "MultiCoS - A Requirements Engineering Tool," *International Journal of Software Engineering and Knowledge Engineering,* vol. accepted for publication, 7 2018.

[60] "TAO: A Testbed for Aspect Oriented Software Developmen," TAO Project, 2007. [Online]. Available: http://www.comp.lancs.ac.uk/research/projects/project.php?pid=215. [Accessed 2012].

[61] C. Ebert, R. Dumke, M. Bundschuh and A. Schmietendorf, "Best Practices in Software Measurement. How to Use Metrics to Improve Project and Process Performance," Springer, New York, 2005.

[62] A. Garcia, C. Sant'Anna, C. Chavez and V. Silva, "Separation of Concerns in Multi-Agent Systems: An Empirical Study," in *Software Engineering for Multi-Agent Systems II. Lecture Notes in Computer Science*, New York, Springer, 2940, 2004, pp. 49-72.

[63] A. Garcia, C. Sant'Anna, E. Figueiredo, U. Kulesza and et al, "Modularizing Design Patterns with Aspects: a Quantitative Study," in *4th Aspect-Oriented Software Development Conference (AOSD'04)*, Chicago,, 2005.

[64] G. Mussbacher, D. Amyot, J. Araújo, A. Moreira and M. Weiss, "Visualizing Aspect-Oriented Goal Models with AoGRL," in *2nd International Workshop on Requirements Engineering Visualization at 15th Requirements Engineering Conference (RE'07),*, New Delhi, 2007.

[65] R. Chitchyan, A. Rashid, P. Sawer, A. Garcia, M. Alarcom, J. Bakker, B. Tekinerdogan, S. Clarke and A. Jackson, "Survey of Analysis and Design Approaches," http://www.comp.lancs.ac.uk/computing/aose/papers/d11.pdf, 2005.

[66] D. Poshyvanyk, M. Gethers and A. Marcus, "Transactions on Software Engineering and Concept Location using Formal Concept Analysis and Information Retrieval," *ACM Transactions on Software Engineering and Methodology (TOSEM),* vol. 21, no. 4, 2012.

[67] A. Marcus and V. Rajlich, "Panel: Identifications of Concepts, Features, and Concerns in Source Code," in

21st IEEE International Conference on Software Maintenance (ICSM2005), Budapest, 2005.

[68] Cutter Consortium, "Poor Project Management Number-one Problem of Outsourced E-projects. CutterResearch Briefs," Cutter Consortium, 11 2000. [Online]. Available: http://www.cutter.com/research/2000/crb001107.html. [Accessed 5 5 2013].

[69] E. Insfrán, A Requirements Engineering Approach for Object-Oriented Conceptual Modeling. PhD Thesis, Valencia, Spain: Department of Information Systems and Computation. Technical University of Valencia, October 2003.

[70] M. Saeki, "Semantic Requirements Engineering," in *Intentional Perspectives on Information Systems Engineering*, Berlin Heidelberg, Springer-Verlag, 2010, p. 67.

[71] K. Verma, A. Kass and R. Vasquez, "Using syntactic and semantic analyses to improve the quality of requirements documentation," *Semantic Web,* vol. 5, no. 5, pp. 405-419, 2014.

[72] K. Wiegers, "Software Requirements," Microsoft Press, 2003.

[73] International Institute of Business Analysis, "The Guide to the Business Analysis Body of Knowledge® (BABOK® Guide) Version 1.6," 2006. [Online]. Available: https://cs.anu.edu.au/courses/comp3120/public_docs/BOKV1_6.pdf. [Accessed 12 12 2015].

[74] J. Murray, "Model Based Systems Engineering," 2 5 2012. [Online]. Available: http://syse.pdx.edu/program/portfolios/julia/MBSE.pdf. [Accessed 5 5 2013].

[75] M. Brambilla, J. Cabot and M. Grossniklaus, "Modeling Safe Interface Interactions in Web Applications," in *International Conference on Web Engineering*, 2009.

[76] L. Olsina, "Building a Web-based Information System applying the Hypermedia Flexible Process Modeling Strategy," in *The 1st Int. Workshop on Hypermedia Development, Hypertext*, Pittsburg, 1998.

[77] U. Consortium, "Ubiquitous Web Applications," in *eBusiness and eWork Conference 2002*, Prague, 2002.

[78] A. T. Fernando Molina, "Integrating that can be evaluated in design time into Model Driven Engineering of Web Information Systems," *Advances in Engineering Software,* vol. 40, no. 12, p. 1306–1317, 2009.

[79] R. Laddad, "I want my AOP!, JavaWorld," 2002. [Online]. Available: http://www.javaworld.com/article/2073918/core-java/i-want-my-aop---part-1.html. [Accessed 5 5 2013].

[80] G. Kiczales, "The Fun Has Just Begun.," in *In 2nd Aspect Oriented Software Development Conference (AOSD'03)*, Boston, USA, 2003.

[81] "AspectJ," 2007, [Online]. Available: http://www.eclipse.org/aspectj/.

[82] IBM Research, "MDSOC: Software Engineering Using Hyperspaces," IBM Research, 2007. [Online]. Available: http://www.research.ibm.com/hyperspace/..

[83] L. Fernandes, "An Aspect-Oriented Approach to Model Requirements," in *In 13th Requirements Engineering Conference (RE'05)*, Paris, 2005.

[84] V. Kruskal, "A Blast from the Past: Using P-EDIT for Multidimensional Editing.," in *Workshop on Multi-Dimensional Separation of Concerns in Software Engineering, ICSE 2000*, 2000.

[85] Eclipse, "Project Plan - Eclipse Process Framework," Eclipse Fundation, 2012. [Online]. Available: http://epf.eclipse.org/wikis/openup/.

[86] A. Van Lamsweerde, "Goal-Oriented Requirements Engineering: A Guided Tour," in *In: Proceedings of the 5th IEEE international Symposium on Requirements Engineering*, Washington, 2001.

[87] Haryono, "Adding Aspect-Oriented Programming features to Visual Basic.NET by using Multidimensional Separation of Concerns(MDSOC) approach," in *MS Project, California State University*, Sacramento, 2003.

[88] P. J. Valderas, A Requirements Engineering Approach for the Development of Web Applications, Valencia, Spain: Department of Information Systems and Computation Technical University of Valencia, 2007.

[89] E. Baniassad, P. Clements, A. Rashid, J. Araújo, A. Moreira and B. Tekinerdogan, "Discovering Early Aspects. IEEE Software Special Issue on Aspect-Oriented Programming.," IEEE, 2006.

[90] E. Baniassad and S. Clarke, "Finding Aspects in Requirements with Theme/Doc," in *The 3rd Aspect-*

Oriented Software Development International Conference (AOSD'04),, Lancaster, 2004.

[91] J. Castro, M. Klop and J. Mylopoulos, "Towards requirements-driven information systems engineering: the Tropos project," *Information Systems,* no. 27, pp. 365-389, 2002.

[92] X. Chen, Z. Liu and V. Mencl, "Separation of Concerns and Consistent Integration in Requirements Modelling," Macao, 2007.

[93] R. Chitchyan, A. Rashid, P. Rayson and R. Waters, "Semantics-Based Composition for Aspect- Oriented Requirements Engineering," in *In 6th Aspect-Oriented Software Development Conference (AOSD'07)*, Vancouver, Canada, 2007.

[94] J. Noppen and et al., "Modelling Imperfect Product Line Requirements with Fuzzy Feature Diagrams," in *Variability Modelling of Software-intensive Systems*, Sevilla, 2009.

[95] I. Groher and M. Voelter, "Expressing Feature-Based Variability in Structural Models," in *Workshop on Managing Variability for Software Product Lines*, 2007.

[96] I. Groher and M. Völter, "Using Aspects to Model Product Line Variability," in *Software Product Lines , 12th International Conference*, Limerick, Ireland, 2008.

[97] J. M. Conejero, R.-E. Roberto, F. Sánchez-Figueroa, M. Linaje, J. C. Preciado and C. P. J., "Re-engineering legacy Web applications into RIAs by aligning modernization requirements, patterns and RIA features," *Journal of Systems and Software,* vol. 86, no. 12, pp. 2981-2994, 2013.

[98] J. M. Rivero and G. Rossi, "MockupDD: Facilitating Agile Support for Model-Driven Web Engineering," in *ICWE 2013 International Workshops on Current Trends in Web Engineering*, New York, 2013.

[99] J. M. Carrillo de Gea, J. Nicolás, J. L. Fernández Alemán, A. Toval and A. V. Christof Ebert, "Requirements Engineering Tools," *IEEE Software,* vol. 28, no. 4, pp. 86-91, 2011.

[100] M. Alférez and et al., "A Model-Driven Approach for Requirements Refinement to Architecture, AMPLE Project," Deliverable D1.4, 2009.

[101] M. Voelter and I. Groher, "Handling variability in model transformations and generators," in *7th OOPSLA Workshop on Domain-Specific Modeling*, Oopsla, 2007.

[102] M. J. Escalona, J. Torres and M. Mejías, "NDT-Tool: A case tool to deal with requirements in web information systems," in *ICWE*, 2003.

[103] M. J. Escalona and N. Koch, "Metamodeling the Requirements of Web Systems," in *WEBIST*, Setubal, 2006.

[104] N. Koch, A. Knapp, G. Zhang and H. Baumeister, "UML-BASED WEB ENGINEERING - An Approach Based on Standards," in *Web Engineering: Modelling and Implementing Web Applications*, Springer, 2008, pp. pp 157-191.

[105] P. Kaminski, "Applying Multi-dimensional Separation of Concerns to Software Visualization," in *Workshop on Advanced Separation of Concerns, ICSE*, 2001.

[106] P. Salinia and S. Kanmani, "Security Requirements Engineering Process for Web Applications," *Procedia Engineering,* vol. 38, p. 2799–2807, 2012.

[107] T. Stahl and M. Voelter, Model-Driven Software Development, Wiley & Sons, 2006.

[108] W. Harrison, H. Ossher and P. Tarr, "General Composition of Software Artifacts," in *Software Composition Workshop 2006*, 2006.

[109] T. Riechert, K. Lauenroth, J. Lehmann and S. Auer, "Towards semantic based requirements engineering," in *Proceedings of the 7th International Conference on Knowledge Management, I-KNOW*, 2007.

[110] A. Rashid, P. Sawyer, A. Moreira and J. Araújo, "Early Aspects, A Model for Aspect Oriented Requirements Engineering," in *In 10th Requirements Engineering Conference (RE'02)*, Essen, Germany, 2002.

[111] A. Rashid, A. Moreira and B. Tekinerdogan, "Editoral Early Aspects: Aspect-Oriented Requirements Engineering and Architecture Design.," *IEE - Proceedings Software,* vol. 151, no. 4, pp. 153-156, 2004.

[112] V. Perrone and D. Bolchini, "iDesigning Communication Intensive Web Applications: A Case Study," in *VII Workshop on Requirements Engineering (WER 2004),*, Tandil, 2004.

[113] H. Ossher and P. Tarr, "Multi-Dimensional Separation of Concern and the Hyperspace Approach.," in

Software Architectures and Component Technology, Kluwer Academic Publishers, 2001, p. Capitol 3.

[114] H. Ossher and P. Tarr, "Hyper/J™: Multi-Dimensional Separation of Concerns for Java™," in *Proceedings of the 23rd International Conference on Software Engineering (ICSE'01)*, IBM T. J. Watson Research Center.

[115] M. Escalona, M. Urbietab, G. Rossib, G. J.A. and E. Robles Luna, "Detecting Web requirements conflicts and inconsistencies under a model-based perspective," *Journal of Systems and Software,* vol. 86, no. 12, pp. 3024-3038, 2013.

[116] A. Vallecillo, N. Koch, C. Cachero and et al, "MDWEnet: A Practical Approach to Achieving Interoperability of Model-Driven Web Engineering Methods. A position paper," in *MDWEnet workshop*, Como, Italy, 2007.

[117] T. Zernadji, C. Tibermacinev, F. Cherifc and A. Zouiouech, "Integrating quality requirements in engineering web service orchestrations," *Journal of Systems and Software,* no. http://information.popular-study.com/stories/52026/Integrating_quality_requirements_in_engineering_web_service_orchestrations.html.

[118] A. Moreira, A. Rashid and J. Araújo, "Concern-Oriented Requirements Engineering Model," in *In 17th Advanced Information Systems Engineering Conference (CAISE'05)*, Porto, Portuga, 2005.

[119] G. Kiczales, J. Lamping, A. Mendhekar and C. Maeda, "Aspect-Oriented Programming," in *In 11th European Conference Object-Oriented Programming (ECOOP'97)*, Jyvaskyla, Finland, 1997.

[120] D. Distante, P. Pedone, G. Rossi and G. Canfora, "Model-Driven Development of Web Applications with UWA, MVC and JavaServer Faces," *ICWE,* pp. 457-472, 2007.

[121] M. Escalona and N. Koch, "Requirements Engineering for Web Applications: A comparative study," *Journal on Web Engineering,* vol. 2, no. 3, pp. 193-212, 2004.

[122] R. Filman, T. Elrad, S. Clarke and M. Aksit, Aspect-Oriented Software Development, Amsterdam: Addison-Wesley Longman,, 2006.

[123] I. N. P. Jacobson, I. Jacobson and P. Ng, "Aspect-Oriented Software Development with Use Cases," Addison-Wesley, 2004.

[124] J. Jendrik and A. Uwe, "Concern-Based (de)composition of Model-Driven Software Development Processes," in *Model Driven Engineering Languages and Systems*, 2010, pp. 47-62.

[125] A. Lai and G. Murphy, "The Structure of Features in Java Code: An Exploratory Investigation," in *Workshop on Multi-Dimensional Separation of Concerns in Object-Oriented System*, Oopsla, 1999.

[126] A. McDonald and R. Welland, "Web Engineering in Practice," in *Proceedings of the 4th Workshop on Web Engineering (in conjunction with 10th Int. Conf. on WWW)*, Hong Kong, 2001.

[127] S. Mellor, A. Clark and T. Futagami, "Model-driven development - Guest editor's introduction," *IEEE Software,* vol. 20, no. 5, pp. 14-18, 2003.

[128] O. Pastor, J. Fons and V. Pelechano, "OOWS: A Method to Develop Web Applications from Web-Oriented Conceptual Models," in *Web Oriented Software Technology IWWOST 2003*, 2003.

[129] I. Sommerville and P. Sawyer, Requirements Engineering: A good practice guide., West Sussex, England: John Wiley & Sons, 1997.

[130] S. Sutton Jr and I. Rouvellou, "Concern Modeling for Aspect-Oriented Software Development," in *Aspect-Oriented Software Development*, Addison-Wesley, 2004.

[131] P. Vilain, D. Schwabe and C. Sieckenius de Souza, "A Diagrammatic Tool for Representing User Interaction," in *UML'2000*, York, 2000.

[132] J. Whittle and J. Araújo, "Scenario Modeling with Aspects," *IEE Proceedings Software,* vol. 151, no. 4, pp. 157-172, 2004.