ISSN 2277-3061

Modeling Concern Spaces Using Multi Dimensional Separation of Concerns

Calin Eugen Nicolae Gal-Chis

Universitatea Babeş-Bolyai Cluj-Napoca, Facultatea de Matematică şi Informatică,
Str. Mihail Kogălniceanu, nr. 1, RO-400084 Cluj-Napoca, Romania
calin.gal-chis@ubbcluj.ro

ABSTRACT

For software products, the specifications, the requirements even the variables, the code or the software modules are subject to be labelled with key-terms, or described using attributes or specific values. The purpose of these notations is linked to the semantic of the object labelled, and is used as an indexing form for that specific category. A separation of concerns meta model is proposed here to provide the support of using a unitary type of notation in labelling various kind of resources used in the process of developing software, from requirements and specifications all the way to variables, code or software modules. The use of a standard, unitary notation can have multiple benefits, covering areas like code reusability, reverse engineering, assigning technologies for development, aspect-oriented software development (AOSD), requirements engineering (engineering web applications, grouping requirements by categories, such as: technology, importance, actor, volatility, functionality).

Indexing terms/Keywords

Separation of concerns; requirements; software engineering; concern space modelling.

Academic Discipline And Sub-Disciplines

Computer Science – Software Engineering

SUBJECT CLASSIFICATION

D.2.1 [Software Engineering]: Requirements/Specifications – methodologies

TYPE (METHOD/APPROACH)

Approach, Standardization, Aspects

1. INTRODUCTION

The separation of concerns in software engineering can provide multiple benefits such as reduced complexity, improved reusability, and simpler evolution. Using concerns help programmers to better visualize their work, stakeholders to relate the specifications. Reverse engineering, change impact analysis and reuse of code are also supported by concerns.Basically, software systems should benefit in many ways when using separation of concerns during product cycle.

Looking to a software product as an outsider, one might consider that describing the concerns produces a substantially large amount of metadata that adds bureaucracy during the development process of the software product. Although is needed to monitor closely the additional complexity that multi-dimensional separation of concerns can bring to the table; the software development and visualization tools are supposed to handle the added complexity; the benefits of using concerns outweigh clearly the miss concept of not using them.

With respects to generality, a concern can be considered to be any matter of interest in a software system, and also consider a concern space as an organized representation of concerns and their relationships. Defining concern spaces and relating concerns to the source code accordingly during development stage has the power to add semantic to the components of an application. The semantic value can be added to different entity spaces such as: the application model, the data, the relations, the views, and the design; also to the requirements, and not only.

We show that, while existing modelling approaches address concerns in specific contexts for specific purposes, a general-purpose concern-modelling capability is still needed. The purpose of this paper is to express the need of a unitary framework for modelling Concern Spaces and the relations with different types of entities, to define primitives needed to represent such relations, and to propose patterns that can be followed in various scenarios during the use of the approach.

The paper will present in section 2 methodologies using different approaches regarding separation of concerns or models for the separation of concerns. In section 3 an approach to model concern spaces will be introduced. In section 4 a case study will be discussed. The paper will end with further work and conclusions sections.

2. RELATED WORK

In [4] authors are stressing the need of formalizing the concern spaces and to represent their relations to units as graphs, so a visual representation can be generated using specific tools. Is debated on using an implicit “null” concern in each space, to automatically map units in the system not connected to any other concerns in the space’s domain. Also, an open discussion is presented over how the graphs should be expressed in terms of mathematical relations, with no restrictions or with restrictions, limiting the mapping to be injective (each unit maps to at most one concern), or surjective (each concern is “covered” by at least one unit). In [1], authors are using separation of concerns based by organizational concepts, not programming concepts, in order to close the semantic gap between the software system and its operational environment. They are proposing a software development methodology named Tropos which is founded on concepts used to model early requirements. Tropos posits five main classes of concerns: actors, resources, (hard) goals, soft goals, and tasks.

Sutton and Rouvellou are proposing to model the software concerns in Cosmos [11]. The Cosmos schema includes three types of elements: concerns, relationships, and predicates. In Cosmos, concerns are divided in two main types: logical and physical. Logical concerns are describing the concepts of interest regarding a system or artefact, such as: example, issues, aspects, features, and properties. Physical concerns indicate the elements of a system or the artefacts of software where the logical concerns can be applied. Cosmos presents a set of five logical concerns: classifications, classes, instances, properties, and topics. We can include in this category: functionality, behaviour, performance, robustness, state, coupling, configurability, usability, size, cost, and others. Physical Concerns are considered to be the “real world” entities of a system, like software, hardware, systems, and services. There are three types of physical concerns in Cosmos: instances, collections, and attributes. Physical instances are particular software units, physical collections are groups of instances or of other (sub) collections, while physical attributes are the characteristics of the previous two physical concerns: instances or collections.Another approach, ModelSoC, that uses concern separation for all artefacts (documents, models, code) as the primary (de)composition method for the complete process, is presented in [3]. They extend the hyperspace model for multi-dimensional separation of concerns to deal with information that is replicated in different models. A implementation based on their framework Reuseware organises all data provided during development in a concern space and composes integrated views as well as the final system from that. Their multidimensional approach refers here to artefacts and not to concern spaces.

In [9] are provided hyperspaces to model the multi–dimensional separation of concerns. The process is used to simultaneous separation of units according to multiple, arbitrary kinds (dimensions) of concern. This allows on-demand re-modularization. One gain is that concerns can overlap and interact. On the other hand, there are parts difficult to map to the hyperspace model as it is, because they are either not considered by the model or further refined parts of the model cannot be easily added to the system.In [12], authors are describing a paradigm for modelling and implementing software artefacts that allows separation of overlapping concerns along with multiple dimensions of composition and decomposition. Their paradigm addresses different stages of software lifecycle to offer artefacts traceability, flexibility and quality.

The separation of concerns method is applied by Chen, Liu and Mencl [13] on Requirements Modelling. Even, though they split the model into several parts their approach is supporting separation of concerns and consistent and incremental modelling of requirements.The separation of concerns is taken to a higher level by Moreira, Rashid and Araujo [7], in a multi-dimensional separation of concerns. Despite of the fact that they give up on using viewpoints, use cases or themes in representing the requirements, the solution provided is conceptualized in such a way that different types of requirements (functional and NFR) are no longer described using separate representations, but are using a unique representation. All requirements are decomposed in a uniform pattern regardless of their functional or non-functional nature. The requirements space is divided into the system space and meta concern space, as in Figure 1. The system space gathers different types of systems that are possible to be realized (i.e. requirements associated to application that are just part of the requirements space); while the meta concern space comprises an abstract set of typical concerns (functional and non-functional requirements), that are found in various systems. Still, the relation in between the meta concern space and the system space entity is an restricting injective mapping.

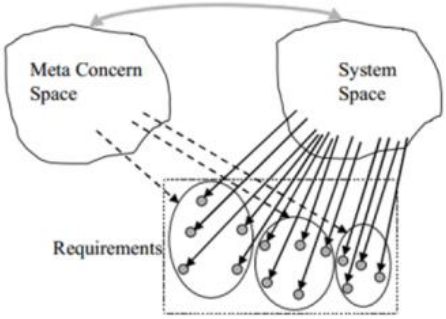


Figure 1. The requirements space in the vision of Moreira, Rashid and Arajo

3. MODELING CONCERN SPACES

One common characteristic of all approaches presented before is the focus on just one particular type of concern spaces. Mostly, the representations of the concern spaces used in each approach has appeared from the need to solve a certain problem, and for that reason a Concern Space was defined. The needs met by the use of Concern Spaces in the presented approaches are: organizational needs of the software system and system's operational environment in [1], artefacts classification needs in [3], remodularization needs by separating *units* in [9], the needs of separating concerns in logical ones and physical ones in software development only [11], artefacts traceability, flexibility and quality needs in [12], requirements modelling needs in [7] and [13]. So, the same concept of Concern Spaces was introduced in different ways by each of the approaches discussed before. A proper, generic type was not introduced, and the lack of standardization allowed multiple definitions of the same concept.

The model presented in this paper was introduced in [2], and extends the multi-dimensional separation of concerns defined by [7] already presented in the previous section. This higher level approach of the concerns spaces, provides a flexible structure and has the power to adapt the domain composition to changes in the range, always offering default mapping solutions. The range can express various types of entities, from common ones like code, software specifications or requirements to more specific ones like software artefacts, user types, developer profiles, technologies or even various informational systems.Concern Space groups one or more concerns and they are represented as multiple dimensions of the space (Figure 2).

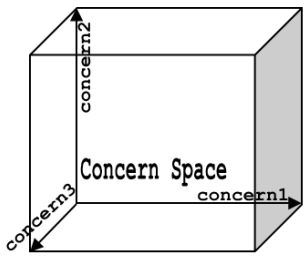


Figure 2. A three-dimensional Concern Space

A novelty introduced by the approach is that the model can refer to multiple system spaces (Figure 3) and can handle multiple concern spaces (Figure 4). The relations formed among these system spaces and concern spaces can add value to the software engineering field, providing a structure to represent relations, dependencies, qualities/attributes of different software concepts or entities.For example, different System Spaces (of two different web applications) can be related through the use of a Concern Space. This can lead to artefacts and code reusability.

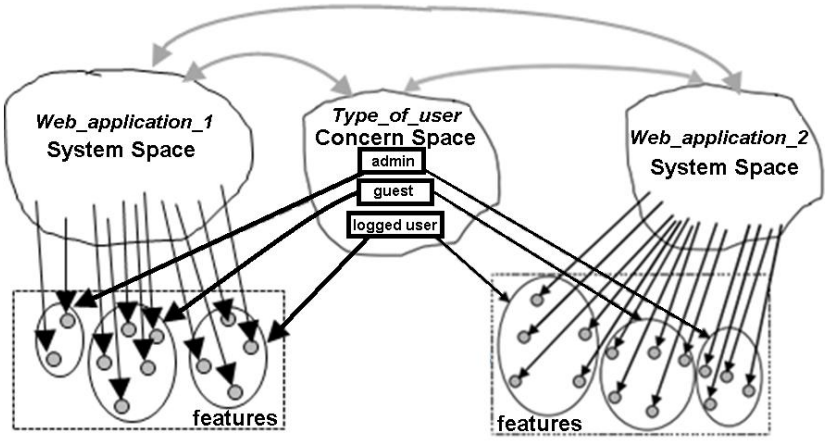


Figure 3. Different System Spaces related through the use of a Concern Space

The following primitives will be defined: Concern Space, MultiSpace, Concern (Concern Dimension), Entity, Concern Value.

- Concern Space = a group of concerns referring to/describing similar capacities (issues /behaviour) of at least one type of entities. A Concern Space has a name, a description, and a set of dimensions (stored as a vector - one dimension for each concern in the space). Optional, parameters can be declared, to configure the Concern Space (a range of values for each dimension – [minValue, maxValue], by default [0,1]). The multiple dimensions of a concern space are represented visually in Figure 3.
- MultiSpace = Concern Space of Concern Spaces – Concern Spaces grouped by the same category of meta data. A MultiSpace space has a name, a description, and a set of Concern Spaces. On same level, the MultiSpaces, can be grouped interest-wise in a Meta MultiSpace on a higher level. So, MultiSpaces can be applied over MultiSpaces. This process can be applied recursively, on higher levels, if necessary.
- Concern = Concern Dimension – one of the elements in the set of dimensions of a Concern Space. A Concern has the purpose of describing an attribute of an entity, by assigning a corresponding value to the relation of an entity to that specific concern.
- Entity – an object that can be associated to one or more Concern Dimensions from a Concern Space. Entities can be requirements, artefacts, users, software modules, technologies, specifications, even concern spaces.
- System Spaces – a collection of entities with cohesion. As entities can be of different categories, they can be grouped in separate System Spaces - such as Requirements system space, Viewpoints system space, Developers system space.
- Concern Value – a value that describes numerically (scalar) the relation between the Entity and the Concern Dimension. The value of the concern reflects the attribute weight of the entity regarding the concern space, the concern range values and the comparing/contrasting relation of the concern with the other concerns in the concern space. The Concern Space had to include at least one dimension (concern).

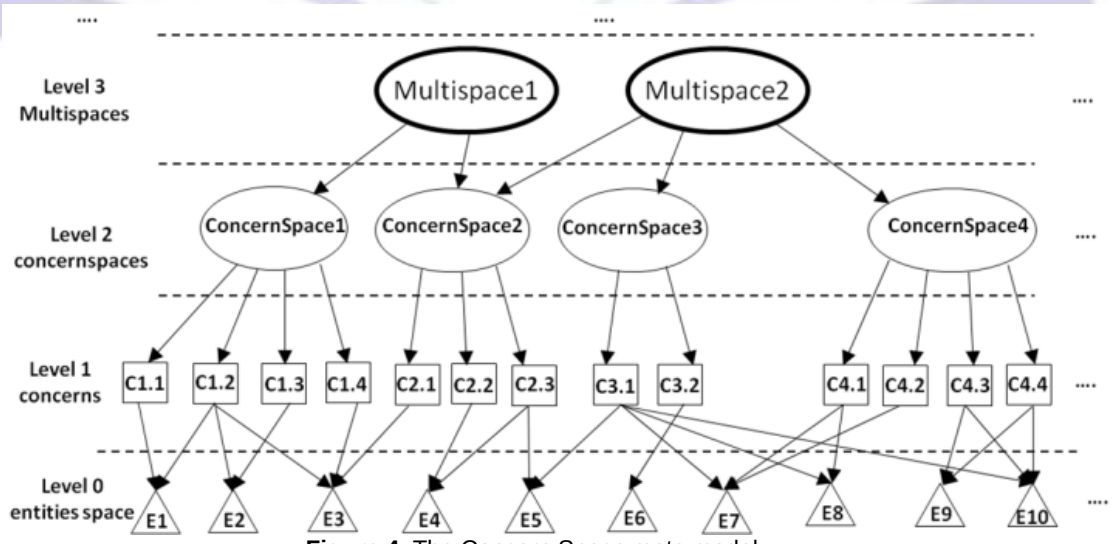


Figure 4. The Concern Space meta model

It can be observed that any two consecutive levels in the meta model can be represented as weighted bipartite graphs. The relation in between the primitives is represented in the form of classes and relations in Figure 5.

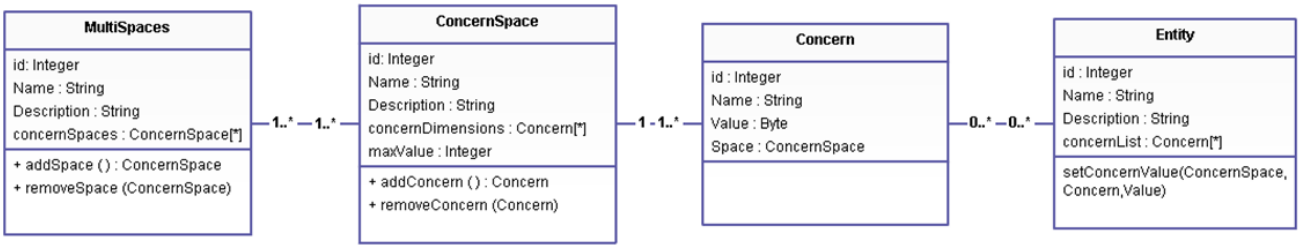


Figure 5. Multi-dimensional separation of concerns Metamodel

In concern spaces one or more dimensions (concerns) can be associated with the entities. Values are assigned for the dimensions from the {0,1} set, or from a set of positive fuzzy values, according to the considered impact of the dimension in the entity can be used, providing in this way a certain weight to the edge. The weight can be specified for each dimension (concern) in the space. By default, all the concerns in a Concern Space are associated to each entity with a value of 0, so no edge will be drawn to indicate the entity representation in that concern. Also, the vector of concerns of a particular concern space assigned to an entity is a vector of concerns with all values set at zero. The weight of a concern-entity relation is not limited just as Boolean or Integer values; real numbers for describing weights (continuous quantities, time) had to be supported as has been demonstrated in [5].

For every entity, there is a vector from each Concern Space that is associated with the entity, expressing the blueprint of the entity in the Concern Space. This association can be expressed as a function with the entity and the concern space as the arguments, and with the associated vector of concerns as a result.

Let *e* be an entity in the system space *E*, *S* a *n*-dimension concern space and *f* a mapping function that describes the relation between the entity *e* and the concern space *S*.

$$f(e,S) = v, \qquad \text{where } e \in E,\ v \in \mathbf{R}^n$$

We will present an example to illustrate the use of the function.

Let's consider the concern space S_1,

S_1={1, "programming language", "the type of programming language", (html, javascript, php, mysql, css), 10}.

The system space E_1 is formed by the available software developers

E_1= {*Dev1*, *Dev2*, *Dev3*, Dev4}.

The system space E_2 is formed by software modules to be implemented

E_2= {*Module1*, *Module2*, *Module3*}, where *Module1* is a *web* form, *Module2* is a database search engine and *Module3* is a product page.

The relations in between the entities in the system spaces and the concern space are described below, using the mapping function.

f(*Dev1*, S_1)={7,4,10,9,2} – developer *Dev1* is specialized in php and mysql

f(*Dev2*, S_1)={10,8,3,2,10} – developer *Dev2* is specialized in html, javascript, css

f(*Dev3*, S_1)={9,7,6,7,8} – developer *Dev3* is proficient in all programming languages

f(*Dev4*, S_1)={10,10,5,4,9} – developer *Dev4* is specialized in html, javascript, css

f(*Module1*, S_1)={8,7,1,1,8} – *Module1*requires developer solid knowledge of html, javascript and css

f(*Module2*, S_1)={4,2,8,9,2} – *Module2* requires developer strong knowledge of php and mysql

f(*Module3*, S_1)={9,5,4,3,10} – *Module3* requires developer strong knowledge of html, css and moderate knowledge of the other languages

So, we can make associations between developers and modules to be implemented. A solution may be: *Dev1* to *Module2*, *Dev2* and *Dev3* to *Module3*, and *Dev4* to *Module1*, and can be influenced by entities relations to other *concerns*, such as hours alocated, priority, etc.

In this way, relating different system spaces E_1, E_2 through the use of a Concern Space S_1 is usefull in linking entities from one System Spaces to entities from another one, such as assigning the right developer to the task of implementing a module in the web application.

As represented in the Figure 5, one entity can be associated with more than one Concern Space. We will consider the following concern spaces:

S_1={1, “MVC”, “the weight in the ModelViewController”, (Model, View, Controller), 1},

S_2={2, “CRUDS”, “Create Read Update Delete Static functionalities of an entity”, (read, create, update, delete, static), 1},

S_3={3, “nonFA”, “non functional aspects of entities”, (sleekdesign, loadspeed, volatility), 10},

S_4={4, “Priority”, “importance of the entity”, (priority), 10}.

The *Entity1* is a requirement in the Requirements Space with the name ”productpage” and the description:“Display a professional looking Product Page with product details”. Below are represented the values of the mapping function *f* with the *Entity1* as the first argument and each of the Concern Spaces introduced as the second argument.

f(*Entity1*, S_1)={0,1,0}

f(*Entity1*, S_2)={1,0,0,0,0}

f(*Entity1*, S_3)={8,4,0}

f(*Entity1*, S_4)={7}

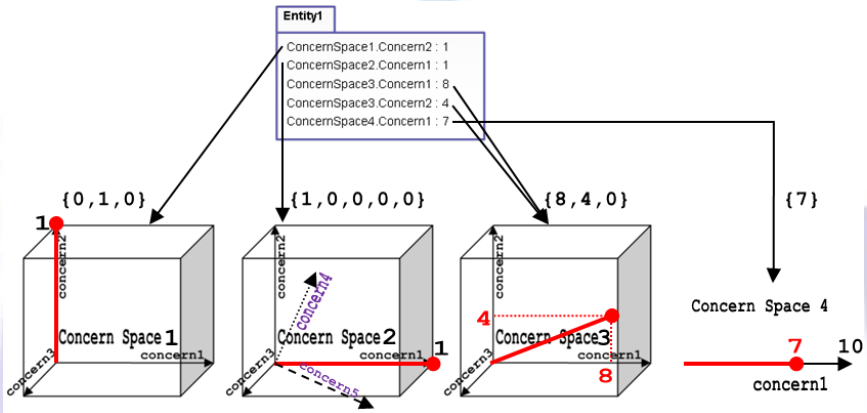


Figure 5. Example of an entity linked to multiple concerns in different concern spaces with the respective values

On refinements of entities such as requirements, the sub-requirements are inheriting all the upper levels (concern spaces) of the original requirement, as presented in Figure 6. Also a new concern-space, based on the original requirement is created, to show the relation in between the newly created sub-requirements, as presented in the next figure.

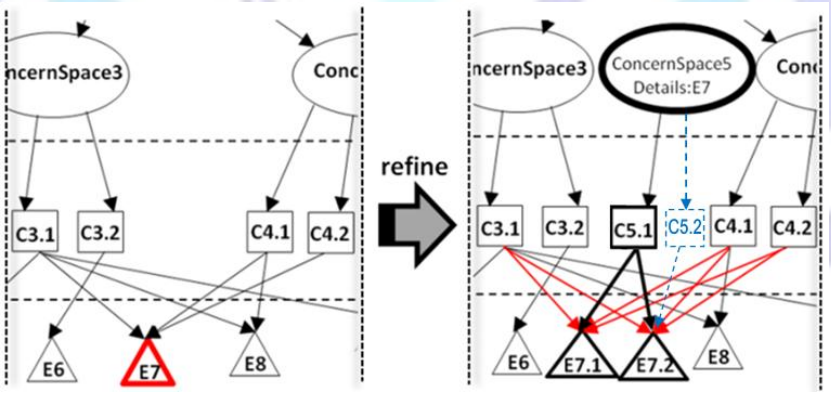


Figure 6. Refining an entity in sub-entities

In this Figure, we can consider the entity *E7*, as being a requirement in the web application Requirements Space with the description: “the members can edit their profile”. This entity can be refined in two entities, entity *E7.1* “the regular members can edit their profile” and entity *E7.2* “the premium members can edit their profile”. The newly created *ConcernSpace5* gets as details the description of the former entity *E7*:” the members can edit their profile” and one dimension is automatically generated: the concern C5.1 “edit the profile”. Following this refinement other concerns can be added to the *ConcernSpace5*, (such as concern C5.2 “upload video profile”) that would be linked to entities (the concern C5.2 can link to entity E7.2 with the sense “premium members can upload a video profile”)

This refinement capacity offers stability (the use of same model structure) and flexibility (anytime entities can be changed and detailed).

ISSN 2277-3061

4. CASE STUDY

This part will investigate the main existing web development methodologies and their tools, with respects to the tools supporting RE.

A wide use of the concerns is for describing requirements. Given a problem to be solved we will describe the requirements using the concern spaces created for the problem. The concern spaces proposed do not have to be used just for requirements. Once the problem is understood the concern spaces can be created together with the stakeholders to make clear different attributes of the product.

> "A web application selling online songs and albums is required. Online users can listen songs and can buy songs. In order to buy songs the user had to be logged into the application and to pay using credits. Credits can be purchased by Credit Card, received as promotion (special holidays/contests) or as loyalty (long period member/purchase volume/ friend invites). Credits also can be donated/transferred from one user to another. Promotions are offered on the homepage. The application had to record browsing history for authenticated users in order to determine their client-profile. For administrative purposes a multi-user back-office will be created to administrate user accounts, music, promotions, and generate reports such as purchases/trends/profiles. Front-end had to be fast and to look professional."

Part of the entities in the Requirements Space will be introduced in Table 1, and will make the subject of our case study.

Table 1. Requirements sample from requirements space

Id	**Requirements**
R1	User listens online music
R2	User purchases music
R3	System can retrieve information
R4	System can record user browsing history
R5	Media have to load fast adjusting to the system capabilities
R6	Front end have to use latest design trends
R7	Front end have to update fast
R8	Application have to be portable on various platforms

Different concern spaces can be applied to the requirements determined in this example. Two concern spaces will be introduced now in the study, in Table 2 and Table 3.

Table 2. Concern Space *Information Retrieval*.

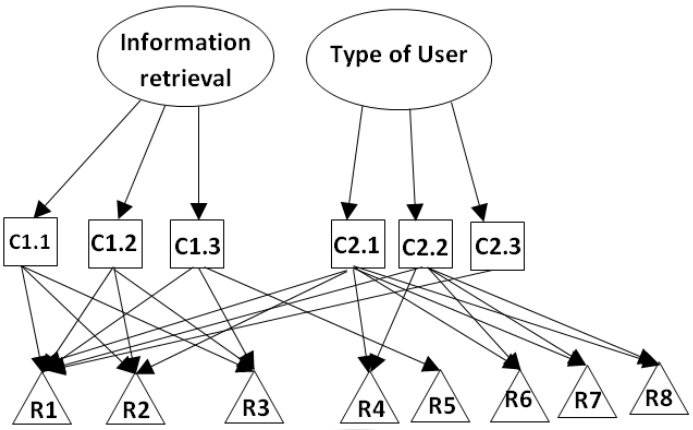
Id = CS1	**Concernspace *Information retrieval***
C1.1	Database retrieval
C1.2	WebPage retrieval
C1.3	HostMachine retrieval

Table 3. Concern Space *Type of User*.

Id = CS2	**Concernspace *Type of user***
C2.1	Client
C2.2	Guest
C2.3	Admin

As associations are made in the model.

ISSN 2277-3061



we can see the graph dependencies in between the requirements and the concern spaces in Figure 7.

Figure 7. Modelling the selected concern spaces over the requirement space of the case study

We can see that, regarding the type of user, most of the requirements are not the concern of the user “administrator”. On the other hand, most of the requirements are connected to “information retrieval” activities or to users “guest” or “client”.

For another view of the relations, a composition table for the mapping function can be made:

Table 4. Composition Table of the relations between entities and Concern Spaces.

Entity	**Concern Space CS1 InformationRetreival**	***f*(Ri,CS1)**	**Concern Space CS2 TypeOfUser**	***f*(Ri,CS2)**
R1	C1.1, C1.2, C1.3	{1,1,1}	C2.1, C2.2, C2.3	{1,1,1}
R2	C1.1, C1.2	{1,1,0}	C2.1	{1,0,0}
R3	C1.1, C1.2, C1.3	{1,1,1}		{0,0,0}
R4		{0,0,0}	C2.1, C2.2	{1,1,0}
R5	C1.3	{0,0,1}		{0,0,0}
R6		{0,0,0}	C2.1, C2.2	{1,1,0}
R7		{0,0,0}	C2.1, C2.2	{1,1,0}
R8		{0,0,0}	C2.1, C2.2	{1,1,0}

This model can also be used on representing the concern spaces used in other approaces. In Model-Driven Software Development (MDSD) information that is captured in artifacts (documents, diagrams, etc.) that are created during the development of systems. These artifacts are regarded as models of the system and are integrated by means of transformation and composition. For example, ModelSoC[3] is introduced as an extension of the hyperspace model for multi-dimensional separation of concern defined by Ossher and Tarr [9] that can handle replication of information in different formats and usage of DSMLs for composing information. During the process, information is transported between different models—i.e., different views on the system—by model transformations. Five different types of viewpoints are supported by the approach: OpenOffice use case documents, UML use case models annotated with invariants, UML class models, Value Flow models and Java. Also, 12 concern dimensions were identified and defined as composition systems. The relation between viewpoints (y-axis) and concern dimension (x-axis) is presented below [Table 5].

Table 5. Viewpoints supported by concerns dimensions in ModelSoC

	usecase	particip.	exchange	flow	trigger	factory	class	dataclass	associate	typebind.	app	security
OpenOffice	x	x										
UML use case	x	x	x									
Value Flow	x	x	x	x								
UML class							x	x	x			
Java	x	x	x	x	x	x[1]	x	x	x	x	x	x[2]

Using our Multi Dimensional Separation of Concerns, we can consider concern dimensions in ModelSoC as concerns in the Composition Systems Concern Space (S_0), and the five viewpoints as entities of the ViewPoints System Space. S_0 = {0, "Composition Systems", "Composition Systems of concerns", (usecase, participation, exchange, flow, trigger, factory, class, dataclass, associate, typebind, app, security), 1}The mapping function *f* using entities from the ViewPoint System Space as the first argument and the Concern Space S_0 as the second will give us the following:

f(*"OpenOffice"*, S_0) ={1,1,0,0,0,0,0,0,0,0,0}

f(*"UML use case"*, S_0) ={1,1,1,0,0,0,0,0,0,0,0}

f(*"Value Flow"*, S_0) ={1,1,1,1,0,0,0,0,0,0,0}

f(*"UML class"*, S_0) ={0,0,0,0,0,1,1,1,0,0,0}

f(*"Java"*, S_0) ={1,1,1,1,1,1,1,1,1,1,1}

The composition systems in ModelSoC concern space S_0 are composing the so-called "concerns". Examples of such "concerns" are given in the ModelSoC example: (a) Customer participates in Book Ticket (b) Bank participates in Book Ticket or (c) Account is exchanged between Customer and Bank. These "concerns" are considered to be "entities" in our approach.Considering the three entities as being part of a System Space called Features, we will be able to mapp the relations in between the entities in Features System Space and the S_0 Concern Space. As mentioned in the ModelSoC example [3], the entities (a), (b) are linked to *participation* and to *usecase* concern dimensions, while (c) is linked to *exchange* concern dimension.

f("(a)Customer ...", S_0) ={1,1,0,0,0,0,0,0,0,0,0}

f("(b)Bank...", S_0) ={1,1,0,0,0,0,0,0,0,0,0}

f("(c)Account...", S_0)={0,0,1,0,0,0,0,0,0,0,0}

Also another Concern Space can be defined here as the Actor Type Concern Space S_1. Concerns in this space can be "Customer", "Bank", but, if needed, can be extended to other dimensions (like "ticket", "account").S_1 = {1, "Actor Type", "types of actors in the system", (Customer, Bank), 1}Considering the Features System Space, we mapp with the *f* mapping function the relations to S_1 Concern Space.

f("(a)Customer ...", S_1) ={1,0}

f("(b)Bank...", S_1) ={0,1}

f("(c)Account...", S_1) ={1,1}

So we can see in this example 2 system spaces and also 2 concern spaces conected by using the multi-dimensional separation of concerns. This way we can track the relations [Figure 8]from features entities to viewpoints entities using the conceptul binding by relating the system spaces to the same concern space S_0, and from one concerns space to another - from Actor types S_1 to composition systems S_0 using the conceptual binding through the Features system space.

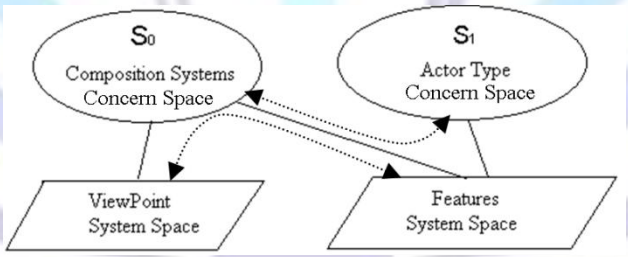


Figure 8. Conceptual bindings in between system spaces and in between concern spaces

So, a transformation map of the primitives from ModelSoC to MultidimensionalSoC will convert as follows [Table 6]:

Table 6. Primitives transformation map from ModelSoC to MultidimensionalSoC

ModelSoC	**MultidimensionalSoC**
Not existent	Multispace
Concern Space	Concern Space
Concern dimension	Concern dimension
Concern	Entity
Exists – not defined	System Space

Different aspects of software development can be captured using concern spaces. The prioritisation analysis made in the Volere project [14] provides guidance on prioritising requirements. The prioritization factors that commonly affect prioritisation decisions are: Minimise Cost of implementation (how much cost to develop?), Value to customer (how much does the customer want it?), Time to implement (how much time to deliver?), Ease of technical implementation (how technologically difficult?), Ease of business implementation (how organisationally difficult?), Value to the business (how much will the business benefit?), Obligation to some external authority (necessity to obey law?). An example of prioritizing the requirements calculating a Priority Rating is provided as a table [Table 7] where a score out of 10 is assigned to every requirement for each prioritization factor.

Table 7. An example of the Volere Prioritisation Spreadsheet

Requirement/Product Use Case/Feature	Value to Customer	Value to Business	Minimise Implementation Cost	Ease of Implementation	Time to Implement
Requirement R1	2	7	3	8	3
Requirement R2	8	8	5	7	6
Requirement R3	7	3	7	4	5
Requirement R4	6	8	3	5	9
Requirement R5	5	5	1	3	7
Requirement R6	9	6	6	5	4
Requirement R7	4	3	6	7	6

Considering the present approach for using concern spaces, the requirements are part of the Requirements Space for a particular application and the prioritization factors are the dimensions in a Concern Space, as presented below:

S_2={2, “prioritization”, “prioritization factors for requirements”, (Value to Customer, Value to Business, Minimise Implementation Cost, Ease of Implementation, Time to Implement), 10},

Given this concern space we can use the mapping function *f* with the requirements as the first argument and the Concern Space S_2 as the second.

$f(R1, S_2)$={2,7,3,8,3}

$f(R2, S_2)$={8,8,5,7,6}

$f(R3, S_2)$={7,3,7,4,5}

$f(R4, S_2)$={6,8,3,5,9}

$f(R5, S_2)$={5,5,1,3,7}

$f(R6, S_2)$={9,6,6,5,4}

$f(R7, S_2)$={4,3,6,7,6}

According to the Eclipse Process Framework (EPF) [15], during the Project Plan phase of developing an application, the team members are assigned roles they play in the project. The work is divided into a number of content areas. Each content area is lead by a committer that is the content lead and responsible for that content area. The content lead is working closely with a number of regular committers. In an example of project plan covering content and enablement portions of the EPF 1.0 project, the work is divided into the following content areas:

- Project management: Chris Armstrong (lead), Jochen Krebs, Per Kroll
- Requirements: Chris Sibbald (lead), Paul Bramble, Ana Paula Valente Pereira, Leonardo Medeiros, Kurt Sand, Bruce MacIsaac, Jim Ruehlin, Ricardo Balduino.
- Change management: Chris Sibbald (lead), Kurt Sand.
- Development: Brian Lyons (lead), Scott Ambler, Ricardo Balduino.
- Architecture: Mark Dickson (lead), Jim Ruehlin, Ana Pereira, Chris Doyle.
- Test: Brian Lyons (lead), Nate Oster, Jeff Smith, Dana Spears.
- General: Steve Adolph (lead).
- Developer outreach: Per Kroll / Naveena Bereny (leads), Ricardo Balduino, Scott Ambler, Kurt Sand.

As noticed, the content lead and the regular commiters are mentioned for each content area. Considering the System Space formed out of the team members, and S_2, a concern space of content areas we can express the relations among

these two spaces using the mapping function *f*. Different weights are given in the concern space to content leads and to regular commiters, such as a content lead has a maximum value of 2, and a regular commiter a lesser value of 1.

S_3 = {3, “content areas”, “team members involvment in the content areas”, (Project management, Requirements, Change management, Development, Architecture, Test, General, Developer outreach), 2},

f(“Chris Armstrong”, S_3) ={2,0,0,0,0,0,0,0}

f(“Jochen Krebs”, S_3) ={1,0,0,0,0,0,0,0}

f(“Per Kroll”, S_3) ={1,0,0,0,0,0,0,0}

f(“Chris Sibbald”, S_3) ={0,2,2,0,0,0,0,0}

f(“Ricardo Balduino”, S_3) ={0,1,0,1,0,0,0,1}

f(“Kurt Sand”, S_3) ={0,1,1,0,0,0,0,1}

f(“Brian Lyons”, S_3) ={0,0,0,2,0,2,0,0}

f(“Steve Adolph”, S_3) ={0,0,0,0,0,0,2,0}

...

In these examples we were able to observe the versatility of this approach over different types of system spaces. Also, we have seen that modeling of concerns in various multidimensional Concern Spaces can be a solution in different existing approaches, and the modelling is clear and can handle the representations of both simple and complex systems.

5. FUTURE WORK

A methodology is necessary in order to systematically investigate and determine the proper concern spaces to be considered and used for different types of entities. Such work can extend the valuable contribution of Poshyvanyk et all in [10].

Samples of concern spaces and multispaces databases along with entities spaces (system spaces) should be created in order to support the development processes based on separation of concerns methodologies.

A tool to support for the implementation of concern spaces is under development. Conversion patterns from types of concern spaces introduced in other approaches towards the model introduced here is needed to prove model generality and acceptance.

Oher visualizations tools can provide graphical renditions, but may also offer query facilities. The manually mapping of code to concerns is difficult and time-demanding, as proved in [6]. Such activities should be assisted by and, if possible, partially automated with specialized tool.

The impact of this approach on web application requirements and product development will be also a subject of future research, as one direction can be to drop concern information into the web application source code for reverse engineering purposes. A valuable contribution is this direction is made by Marcus and Rajlich in [8].

6. CONCLUSION

The approach presented here is meant to introduce a generalization of the concern spaces and of the separation of concerns modelling. The model presented has multiple possible applications, such as requirements engineering, informational systems, reverse engineering. The model can be expressed visually as a graph, can handle complex scenarios and is flexible, reliable through the life cycle of the scope.

ACKNOWLEDGMENTS

This work was possible with the financial support of the Sectoral Operational Programme for Human Resources Development 2007-2013, co-financed by the European Social Fund, under the project number POSDRU/107/1.5/S/76841 with the title „Modern Doctoral Studies: Internationalization and Interdisciplinarity”.

REFERENCES

[1] Castro J., Kolp M., and Mylopoulos J.2002. “Towards requirements-driven information systems engineering: the Tropos project” - Information Systems 27 (2002) 365–389

[2] Gal-Chis C.E.N., 2013.A Multi-Dimensional Separation of Concerns of the Web Application Requirements, Studia Universitatis Babes-Bolyai, Series Informatica, Volum LVIII, nr. 3, 2013, pp 29-40

[3] Jendrik J., Uwe A. 2010. Concern-Based (de)composition of Model-Driven Software Development Processes, Model Driven Engineering Languages and Systems – 2010 p.47-62

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

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

[6] Lai A., Murphy G. C., 1999. The Structure of Features in Java Code: An Exploratory Investigation. Workshop on Multi-Dimensional Separation of Concerns in Object-Oriented Systems, OOPSLA '99,

[7] Moreira A., Rashid A., Arajo J., 2005. Multi-Dimensional Separation of Concerns in Requirements Engineering. IEEE.

[8] Marcus, A., Rajlich, V., 2005. Panel: Identifications of Concepts, Features, and Concerns in Source Code, the Proceedings of the 21st IEEE International Conference on Software Maintenance (ICSM2005), Budapest, Hungary, September 25-30, 2005, p. 718

[9] Ossher H., Tarr P., 2002. MultiDimensional Separation of Concerns and the Hyperspace Approach

[10] Poshyvanyk, D., Gethers, M., and Marcus, A., 2012. Concept Location using Formal Concept Analysis and Information Retrieval, ACM Transactions on Software Engineering and Methodology (TOSEM), 21(4), 2012.

[11] Sutton Jr., S. M., Rouvellou, I. 2002. Modelling of Software Concerns in Cosmos, 1st International Conference on Aspect-Oriented Software Development, Enschede, The Netherlands, April, 2002

[12] Tarr P., Ossher H., Harrison W., Sutton Jr. S.M. "N-degrees of separation: Multidimensional separation of concerns"

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

[14] *** - Prioritisation Analysis - http://www.volere.co.uk/prioritisationdownload.htm - 2009

[15] *** - Project Plan - http://epf.eclipse.org/wikis/openup/

Author' biography with Photo

Calin Eugen Nicolae Gal-Chis is a PHd student in Computer Science at Universitatea Babeş-Bolyai Cluj-Napoca, Facultatea de Matematică şi Informatică Department, Cluj-Napoca, Romania