XCOMP: A Multidimensional Approach for Composing Specific Domains

Asmaa Baya, Bouchra EL Asri, Mahmoud Nassar
IMS Team, SIME Laboratory. ENSIAS
ENSIAS
Rabat, Morocco
bayaasmaa@gmail.com, {elasri, nassar}@ensias.ma

Abstract—**Despite the interest of DSM (Domain Specific Modeling), the definition of new specific domains is still expensive and difficult to achieve. To overcome this problem, several research works propose the composition of existing domains in order to get the maximum benefit from domains that have already been developed and tested. In this context, this article proposes a new approach for composing specific domains models. First, we analyze some related works. On the basis of the key findings and conclusions drawn from the analysis, we propose a multidimensional approach based on the composition of crosscutting concerns contained in the source domain models.**

Keywords-Composition; DSM; separation of concerns; modularization; abstraction

I. INTRODUCTION

The main interest of DSM (Domain Specific Modeling) is to capitalize on the concepts of a given domain and to promote the reuse of domain artifacts. In fact, DSLs (Domain Specific Languages) [15][16] help to reduce the complexity introduced by the specification of general-purpose modeling languages such as UML, since their concepts are aligned with the problem domain [10][11].

Although the importance of DSM is obvious, the development of new specific domain is still expensive; because it is time-consuming as it requires experience and expertise in the new field. To solve this problem, the research community focuses on finding alternative orientations such as Configurable Domain-Specific Development Environment (CDSDE) [1] and domain composition [7][8]. In this paper, we are interested in domain composition as a practical solution to get the maximum benefit from domains that have been already developed and tested.

Developing domains by composition is intensified through a wide range of research to extend the use of specific domains [7][8][9][12][13]. Indeed, the trend is to build libraries of subdomains metamodels that are reusable. These basic domains (like interconnection of modules, specialization, finite state machines, Petri networks, etc.) are used in the construction of most domains. Thus, modeler can construct new domains just by extending and defining rules of composition between these basic subdomains. On this basis arises the need to define a new domain composition approach which provides clear and practical mechanisms to extend and compose domains.

In this paper, we treat domain composition issues. The paper is structured as follows: The first section presents some composition approaches and their comparative study. The second section presents our approach to compose specific domain. We begin this section by explaining the context of the work and then present the different phases of this approach.

II. COMPOSITION APPROACHES FOR SPECIFIC DOMAINS: THE STATE OF ART

In this section, we analyze some current composition approaches for specific domains, as we draw up a summary of the criteria that guide our contribution.

A. *Current domain composition approaches*

Several research studies focused on the problem of composition for specific domains. This contributed to the emergence of several orientations. In what follows, each paragraph presents a specific orientation.

Conceptual composition: The approach of Vega [7] proposes to compose domains at the conceptual level rather than components infrastructure, an additional level is inserted between the two levels in order to ensure their synchronization. The first step of this approach is to materialize the concepts of the composite domain by classes of metamodel. The metamodel of the composite domain become an extension of the source domain metamodels.

Transformation to pivot language: “Multi-modeling views” approach offers a solution to the composition of heterogeneous models (called high-level models) by transforming both high-level models into low level models that conform to the same existing metamodel, or conform to an extension of it [8]. A correspondence model ($CM_{high\text{-}level}$) is used to align high-level models by describing the relationships of correspondence between the composing elements. Then high-level models specified in different domain specific languages submit a sequence of transformation steps in order to translate them into a common low level language. A $CM_{high\text{-}level}$ needs also to be propagated through the complete transformation chains in order to automatically derive $CM_{low\text{-}level}$ (correspondence model of low level models). Finally, this approach proposes to make a homogeneous model composition like “Package merge” [14].

Composition by adopting Template: “Template instantiation” is a metamodel composition approach [9]. This approach is based on the reuse of common metamodeling patterns in a single composite metamodel. Those patterns are saved as abstract metamodel templates and will be instantiated in domain-specific metamodels. This approach of composition does not bring any changes to the source metamodels; however, it automatically creates new relationships between the pre-existing entity types in a target metamodel to make them play roles in a common metamodeling pattern.

Extending languages: Many approaches propose to extend language in order to support domain composition [17] [12]. For example the approach of Ledeczi proposes to extend UML with new operators in order to combine source metamodels [12]. Another approach proposes to extend the UML metamodel with behavior describing symmetric, signature-based composition of UML model elements [17].

Coordination: the composition by coordination was adopted in [13]. This approach proposes architecture of coordination where every participant preserves its autonomy, and the coordinator organizes the collaboration of all participants. Another work proposes to compose an inter domain specific languages coordination [18]. This work introduces an inter-model traceability and navigation environment based on the complementary use of megamodeling and model weaving.

Graphical composition: Some works solve the problem of composition at the graphical level. In [19], researchers define a layer for graphical syntax composition. This work provides formally defined operators to specify what happens to graphical mappings when their respective metamodels are composed.

B. *Review of domain composition approaches*

The proposed approaches are non-exhaustive; Model Driven Engineering (MDE) contains a lot more approaches. However, we chose these methods because each one illustrates a particular orientation. The approaches above have several advantages and disadvantages that we present in what follows.

In Vega’s approach, the conceptual stage before the composition of metamodels aims at conceptualizing the concepts of composite domain. Thus, the composition is expressed at a high level abstraction. The disadvantage is that the approach remains complex as long as this high level abstraction is not supported by simple and practical methods.

The approach "Multi-modeling views composition" is based on a simple principle which is the transformation of a high-level language to a low-level language. However, it follows a long process and goes through several transformations and intermediate models before getting the final model. This increases the complexity of this approach and its margin of error.

“Template instantiation” approach ensures a high-level abstraction through the use of abstract metamodel templates. However, these templates have several limitations such as validity, adaptation and instantiation, which greatly minimize their context of use.

Extending languages maximizes reuse and makes profit of existing languages. However, this orientation is not generative because of its limitation to specific languages.

The composition by coordination presents a lot of advantages like the independence of source domains and a high level of abstraction. However, it proposes usually many languages and techniques which are not easy to use by domain experts.

A graphical composition is a limited orientation, because it solves the problem of composition at the syntactical level only.

Based on these results, we can conclude that high-level abstraction approaches often provide a high level reuse and a good operating range. However, this advantage is often accompanied by a high level of complexity. Thus, the challenge to overcome is to ensure a high level of abstraction while avoiding both introduction of complex mechanisms and limitation of the context of use.

III. Multidimensional composition approach

Looking through the approaches of domain composition presented in section II, we can conclude that every approach is designed for a specific context or language. In addition, approaches are not enough flexible to support modifications in domain models structure. For these reasons, we propose a generative approach applicable in all contexts and languages. This approach pays particular attention to the composition of concerns contained in the domain models. So, it allows the composition of domain models or just a composition of domain models and concerns (security, persistence, etc.). This approach allows also the remodularization on demand; because every designer organizes the source models according to his point of view so the result of composition changes.

In this section, we describe the context of this work, as we present our approach.

A. *Work context*

In this work, we will focus on developing large-scale systems. This context presents additional challenges to overcome, namely, working simultaneously on several specific domains or extensive domains, heterogeneity of models and languages, integration of new non-functional concerns throughout the system life cycle.

To meet these requirements, this work will be based on three strategic axes (see Figure 1):

- Functional: Deals with the Composition of the business part of domains. Indeed, the composite domain must capitalize all the concepts contained in the source domains.

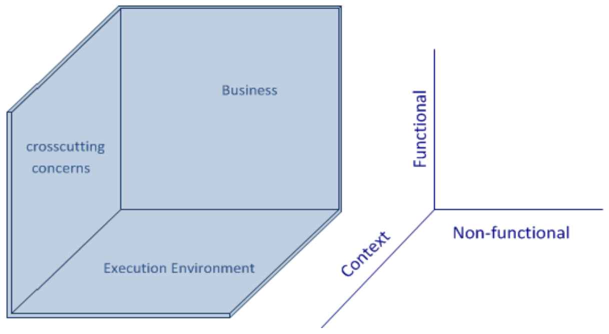

Figure 1. Axes of work.

- Non-functional: The crosscutting concerns contained in each source domain must remain valid in composite domain
- Context: Additional concerns may appear in the composite domain to meet the changing context.

B. XCOMP: A multidimensional composition approach

In this context, particular attention to crosscutting concerns is required. For this, we thought to divide the domains to compose into dimensions. This notion of dimensions was introduced in the multidimensional separation of concerns, which is a general and very ambitious approach: It provides scalability and modularity [1][2].

Multidimensional separation of concerns suggests introducing as many arbitrary dimensions as needed. Separation of concerns will be carried out according to defined dimensions. This allows great flexibility because each actor can define its own dimensions depending on his own vision, as it can add dimensions (not provided a priori) throughout the system life cycle. In principle, concerns are considered as independent and orthogonal, but it is not fully applicable in practice. That is why we will have to define integration rules in order to specify inter-concerns interactions and overlaps in a later step.

The basic principle of our approach is to organize the domain models to compose in independent dimensions, where each dimension is represented by a part of the model. Then compose these dimensions in order to have a composite domain model. In what follows we will detail the steps of our approach.

➢ Step 1:

The first step is to define the dimensions contained in each source domain, where each dimension contains a set of concerns. The choice of dimensions is not governed by rules; however, care should be taken to minimize the inter dimensions relations in order to obtain orthogonal dimensions. Then, decompose the source domain models following these dimensions. Each dimension must be represented by a part of the model (called block). To keep the consistency of the model, we must trace the relationships between the blocks.

Thus, each source domain model is represented by the following triple **M = < D, B, R >**

where:

- $D = \{D_1, D_2 ..., D_n\}$ is the set of dimensions ,and D_i are defined dimensions
- $B = \{B_1, B_2, ..., B_n\}$ is the set of blocks corresponding to the dimensions
- $R = \{R_{1,1}, R_{1,2}, ..., R_{n,n}\}$ is the set of relations which rely the different blocks of domain model. $R_{i,j}$ is the relation that rely the block i to the block j.

The new architecture of the domain model will be presented in a matrix as shown in Figure 2.

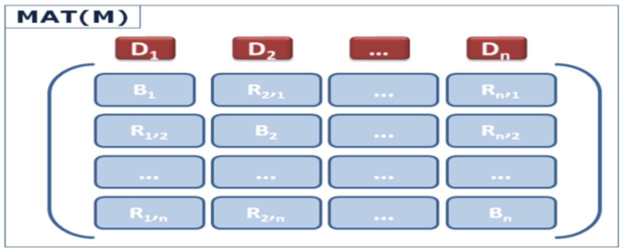

Figure 2. Matrix of domain model.

➢ Step 2:

Once we have defined the source model matrices, we

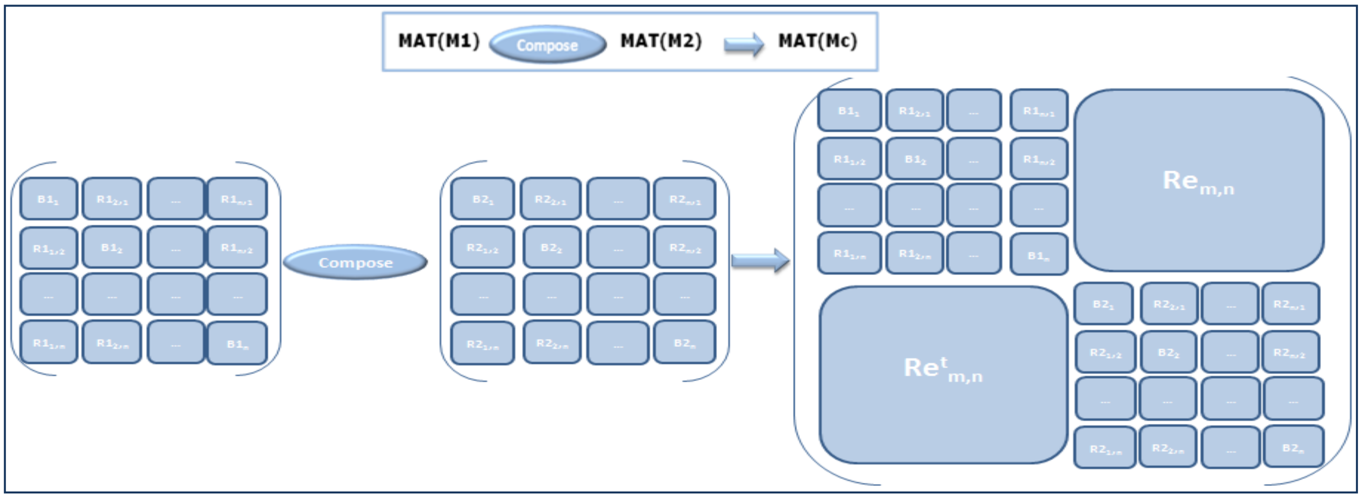

Figure 3. Composition of domain model matrices.

must eliminate repeated dimensions in the models so that they appear only once in the entire source models. Then compose source matrices (MAT (M1) and MAT (M2)) in order to obtain the composite matrix (MAT (Mc)), using the relation in Figure 3.

The diagonal of MAT (Mc) contains blocks which represent the dimensions of source models. The $R_{i,j}$ represent the relationships between blocks from the same models. To ensure coordination between blocks from different models, we will define new relationships that will meet the composite domain requirements. These emergent relationships are represented by $Re_{m,n}$ in MAT (Mc).

$Re_{m,n}$ is a matrix which organize relationships between blocks of the first domain model and blocks of the second domain model. Relations can be expressed in one of the domain specific languages used in source domain models.

The matrix of the composite domain contains all dimensions contained in the source models. At this level, new concerns can be inserted into composite domain in order to adapt it to specific context.

➢ Step 3:

At this stage, we obtain a matrix of composite domain model. This matrix is represented by a set of blocks and relationships. However, the orthogonality of dimensions is not necessarily assured because of the introduction of new relationships ($Re_{m,n}$). In this step, we will make transformations to MAT (Mc) in order to have a domain with independent dimensions.

To achieve this, we propose to represent all relations $Re_{m,n}$ in a new dimension instead of dispersing them throughout the model. The new dimension must make reference to all dimensions and its sub-elements that are members of relations, as it must contain the detailed specification of relationships. To model this new dimension, we can use one of the source domain specific languages or another language that meets requirements of inter-blocks relations.

The resulting model is constituted by a set of blocks from different models, so they are described in various domain specific languages. Our approach stops at this level, since all proposals to homogenize blocks must specify the target language (DSL or a GPL generative programming language). However, our goal is to propose a generative approach applicable to any type of language. Indeed, to adapt the composite domain model, there are other alternatives such as keeping models heterogeneous and proceeding directly to the code generation, or applying one of the transformation methods from DSLs to DSL or from DSLs to GPL to unify the language used in the model [3][4].

IV. USE CASE

To illustrate what we explain above, we present a simple example of composition of data domain and workflow domain. Figure 4 shows the model of data domain and Figure 5 shows the model of workflow domain.

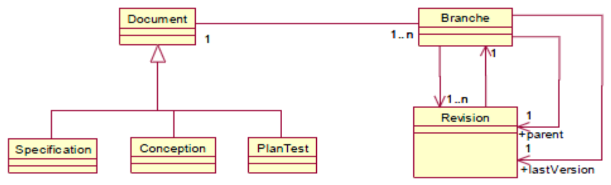

Figure 4. Model of data domain

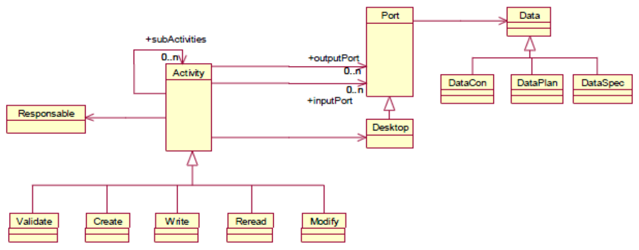

Figure 5. Model of workflow domain

The first step is to identify dimensions, blocks, and relations of domain models. The model of data domain can be organized in two dimensions of concerns: Data and organization. Each dimension is represented by a block. Figure 6 shows the blocks and relations contained in the model of data domain.

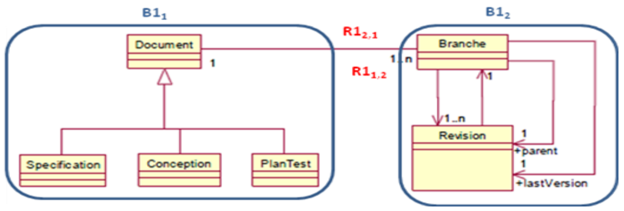

Figure 6. XCOMP applied to model of data domain

The model of workflow domain can be organized in three dimensions: Data, control, resource. Blocks and relations of this model domain are shown in Figure 7.

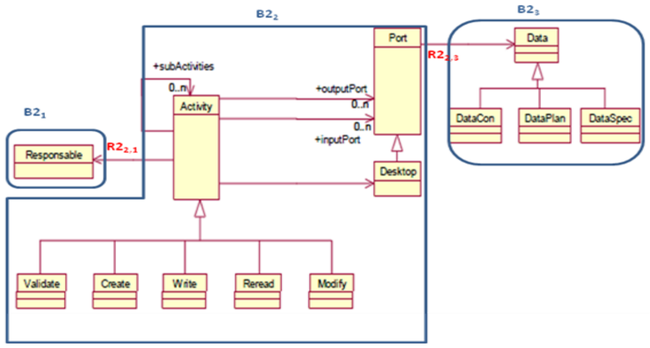

Figure 7. XCOMP applied to model of workflow domain

As we can see through the models; the dimension data appear in two models, so we must eliminate it from one model (for example from workflow domain model). The relation between this dimension and the rest of the model will appear in the matrix ($Re_{m,n}$). To simplify, we suppose that we have no emergent relationships.

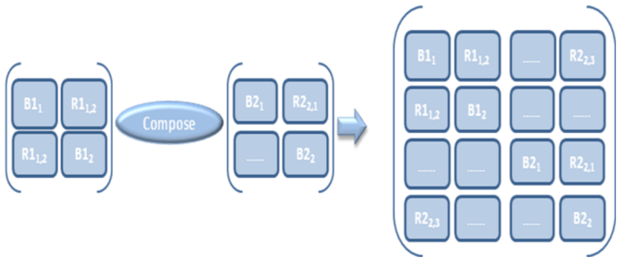

Figure 8. Composition of data model domain matrix and workflow model domain matrix

The Figure 8 presents the source model matrices as it presents the result of composition.

V. CONCLUSION AND FUTURE WORK

This paper presented a generative approach to compose domains. The main motivation behind this proposal is to promote the composition of crosscutting concerns, and to allow the insertion of new concerns throughout the domains life cycle. However, this approach uses several theoretical basis that must be implemented in order to have a practical approach. Our future work will focus mainly on proposing practical methods to decompose domains on orthogonal dimensions. We consider this point as a prerequisite in our approach, but it is not often easy to achieve it. So the approach must be supplied with additional mechanisms to facilitate decomposition of models on orthogonal and independent dimensions.

The last step of the approach should also be extended. In fact, the new dimension representing the relationships between the blocks can be defined in any language. Whether we choose a generative language or one of the source specific languages to model this dimension, we recognize that they are not suitable for all purposes, especially when they do not provide enough abstraction to reference dimensions, blocks, and describe the relationships. So, we propose to define a new specific language dedicated to the description of inter-dimensions relations. This DSL will be used also to formally define the matrix $Re_{m,n}$.

To build this approach, we used the basic principle of the multidimensional separation of concerns. This type of separation has several advantages: The separation is done according to multiple concerns, which deal with the problem of the dominant decomposition according to only one kind of concern at a time. In addition, it allows on-demand remodularization. Indeed, it is possible to add or omit concerns throughout the life cycle without having to change the entire model.

This domain composition approach begins first with a multidimensional decomposition of concerns and then, composes all dimensions to obtain a composite domain. Finally, it proposes to insert a new dimension which includes the relationships between dimensions.

REFERENCES

[1] P. L. Tarr, H. Ossher, W. Harrison, and S. Sutton, “N degrees of separation: multi-dimension separation of concerns,” In Proceedings of the ICSE 99, International Conference on Software Engineering, Los Angeles, CA, USA, pp. 107-119, May 1999

[2] H. Ossher and P. Tarr, “Multi-dimensional Separation of Concerns in hyperspace,” ECOOP’99 workshop on Aspect-Oriented Programming, Lisbon, June 1999.

[3] M. Brambilla, P. Fraternali, and M. Tisi, “A transformation framework to bridge domain specific languages to MDA,” Models in Software Engineering: Workshops and Symposia at MODELS 2008, Toulouse, France, pp.167-180, Sept. 2008.

[4] J. Bezivin, G. Hillairet, F. Jouault, I. Kurtev, and W. Piers, “Bridging the ms/dsl tools and the eclipse modeling framework,” International Workshop on Software Factories at OOPSLA, 2005

[5] M. Wimmer, A. Schauerhuber, M. Strommer, W. Schwinger, and G. Kappel, “A semi-automatic approach for bridging DSLs with UML,” 7th Workshop on Domain- Specific Modeling at OOPSLA, Montreal, Oct. 2007

[6] A. Hovsepyan, S. V. Baelen, Y. Berbers, and W. Joosen, “Specifying and composing concerns expressed in domain specific modeling languages. Objects, components, models and patterns,” 47th International Conference, TOOLS EUROPE 2009, Zurich, June 2009.

[7] G. Vega, “Développement d'applications à grande echelle par composition de méta-modèles, ” Ph.D. thesis, University Joseph Fourier, December 2005.

[8] A. Yie, R. Casallas, D. Deridder, and D. Wagelaar, “A practical approach to multi-modeling views composition,” Proceedings of the 3rd International Workshop on Multi-Paradigm Modeling, Denver, Colorado, USA, jan. 2009

[9] M. Emerson and J. Sztipanovits, “Techniques for metamodel composition” OOPSLA – 6th Workshop on Domain Specific Modeling, pp. 123-139, October, 2006

[10] M. Harsu, “A survey on domain engineering” Report 31, Institute of Software Systems, Tampere University of Technology, pp. 26, Dec. 2002.

[11] DSM Forum, Why DSM, http://www.dsmforum.org/

[12] A. Ledeczi, G. Nordstrom, G. Karsai, P. Volgyesi, and M. Maroti, “On metamodel composition,” Proceedings of the 2001 IEEE International Conference, Mexico, pp. 756-760, Sept. 2001.

[13] T. Le-Anh, “Fédération: une architecture logicielle pour la construction d’applications dirigée par les modèles,” Ph.D. thesis, University Joseph Fourier, january 2004.

[14] J. Dingel, Z. Diskin, and A. Zito, “Understanding and improving UML package merge,” Software and Systems Modeling, Vol. 7, pp. 443-467, Oct. 2008.

[15] J. L. Bentley, “Programming pearls: little languages,” Communications of the ACM, Vol. 29, N. 8, pp. 711–721, August 1986.

[16] A. V. Deursen, P. Klint, and J. Visser, “Domain-specific languages: an annotated bibliography,” ACM SIGPLAN Notices,Vol. 35, N . 6, pp. 26–36, june 2000

[17] F. Fleurey, R. Reddy, B. Baudry, and S. Ghosh, “Providing support for model composition in metamodels,” Proceedings of EDOC 2007, Annapolis, MD, USA, October 2007

[18] F. Jouault, B. Vanhooff, H. Bruneliere, G. Doux, Y. Berbers, and J. Bezivin, “Inter-dsl Coordination support by combining megamodeling and model weaving,” In: Proceedings of the 2010 ACM Symposium on Applied Computing, Suisse Sierre, pp. 2011-2018, Mar 2010.

[19] L. Pedro, M. Risoldi, D. Buchs, B. Barroca, and V. Amaral, “Composing visual syntax for domain specific languages,” 13th International Conference, HCI International 2009, San Diego, CA, USA, pp. 889-898, July 2009.