Behavioral and Structural Model Composition Techniques: State of Art and Research Directions

NISRINE EL MARZOUKI[1, 2], YOUNES LAKHRISSI [2], OKSANA NIKIFOROVA[3], MOHAMMED EL MOHAJIR[1], KONSTANTINS GUSAROVS[3]

[1] LIMS Laboratory, Faculty of Sciences Dhar el Mehraz, Sidi Mohammed Ben Abdellah University
Fez, MAROCCO
[2] ERSI Laboratory, ENSA of Fez, Sidi Mohammed Ben Abdellah University
Fez, MAROCCO
[3] Faculty of Computer Science and Information Technology, Riga Technical University
Riga, LATVIA

elmarzoukinisrine@gmail.com, younes.lakhrissi@usmba.ac.ma, m.elmohajir@ieee.ma,
Oksana.Nikiforova@rtu.lv, konstantins.gusarovs@rtu.lv

Abstract: - MDA [1] allows developers to build models without knowledge of other models in the system and then combine those models to create a system community in order to handle the complexity of a model-driven design process. In this context, we assert that developers need support for composing and manipulating their models to expose how elements of functionality relate to one other. To address this need, we present in this paper the state of art of model composition techniques based on earlier works by focusing on t he various parameters that governed and characterize their behavior, then we describe the results of this survey on the future use of composing UML class diagrams based on the two-Hemisphere Model driven approach [2,3]. The motivation for this study derives from the desirability of discovering more and new effective ways in reusing or adapting the existing methods to create a novel composer framework involving in the MDA concept and answering the key criteria of model composition.

Key-Words: - Model Driven Architecture, Model Composition, Multi-modeling, Two-Hemisphere Model Driven Approach, Behavioral Composition, Structural Composition

1 Introduction

In many disciplines, when a problem becomes more complex, there is a natural tendency to try to break it down into smaller, distinct but connected pieces. The concept of breaking down a system into smaller components is generally referred to decomposition.
Indeed, the activity of decomposing problems needs a step of composition at a specific time to get a global representation of a system under construction and to reason about the system as a w hole for verification, validation and consistency checking purposes. That's why model composition is a challenging topic of interest in which the definition of new approaches should benefit from existing composition model techniques.
This paper, therefore, aims at classifying, identifying publication fora, and performing thematic analysis of the current literature in the model composition for creating an extensive and detailed understanding about this area, thereby determining gaps by graphing and pinpointing in which research areas and for which study types a shortage of publications still exits. We have conducted this study to first scrutinize the model composition contributions produced over time, and then use this study to create a n ew composer framework for composing the UML class diagrams available from the two-hemisphere model driven approach, which is presented in the form of business process model and the concept model.
The outline of the paper is the following. Section 2 provides the background of our research in the form of some common concepts and definitions related to model composition activities and MDA approach. The different types of multi-modeling approaches are given in Section 3. Section 4 exposes the design approaches and the mechanism of composition implemented in those approaches. Section 5 gives a

global overview of existing model composition, and discussing their main behavior and structure. Section 6 draws some conclusions and points out the two-hemisphere model driven approach as a direction for future work.

2 Background

2.1 Introduction
There is little compromise in model composition and MDA jargon, and even less on t he basic specifications of a model composition solution. To address this need, the authors present in this section a common set of concepts and definitions for model composition in order to use this characteristics to identify the assessment criteria of existing model composition techniques.

2.2 Complexity
The increasing complexity of systems has led in recent years to numerous proposals of expressive structuring mechanisms such as m odules, viewpoints, components, software architecture, or models. The corresponding entities are designed separately, which increases their reusability while making their integration more complicated.
Indeed, a real software system is too much complex to be described in a single model. Multiple models should be created for the system specification. Increased system complexity typically brings with it the following problems:

- Longer development times;
- More complex assembly due to number of components and number of people involved;
- Increased cost and time for testing;
- Increased maintenance costs.

Overall, this results in an increased time to market for any system, and increased development and maintenance costs in order for there to be any confidence that the quality of the system is not compromised.
For software systems, as well as the problems outlined above which relate to the fundamental increase in lines of code, there is an additional qualitative difference to the systems being developed today compared to those of decades past. Modern systems are increasingly distributed in nature, as demonstrated by the ubiquity of enterprise applications.
This adds another dimension to software complexity, and brings added challenges of communication and security to those listed above. Since the challenge of managing complexity is the main topic

2.3 Diversity
The challenge of diversity reflects how developers have to manage in a non-homogenous environment. Life would be much easier if there would be only one programming language and one deployment platform, but of course this is not the case, and for very good reasons.
Therefore, this complexity has an obvious consequence on the developed systems, which hampers their development and reuse. This problem is characterized by symptoms known under the names of scattering and tangling (identified for the first time by [4] and used as justification for the programming approach aspects).

- Scattering: in this case a concern (functional or transversal) is distributed throughout the system, and not placed in a cl early identified unit. For example, if a feature is distributed over several components, the cost of an update of this feature can be considerable.
- Tangling: in this case a unit contains several elements from different concerns. We have in the same component an inter-connection between multiple features, and therefore the management cost of the various interactions between these features can be important. [4]

To address this need, several approaches adopting the principle of separation of concerns have been proposed. They allow a decomposition of modeling, especially in the design phase where several models can be developed separately to represent a particular perspective of the system. [5, 6].

2.4 Decomposition
The idea of decomposition can be considered as an effective strategy for changing the representation of a classification problem. Indeed, [5] considers decomposition as the most useful form of transformation of data sets. [2, 3]
The decomposition approach is frequently used in economics, finance and engineering. For instance, decomposition of systems is considered to be a practical way to improve forecasting. The usual decomposition into trend, cycle, seasonal and irregular components was motivated mainly by MDA, who wanted to improve computational efficiency and robustness of systems.
Although decomposition is a promising technique and presents an obviously natural direction to follow, since there are any works in MDA literature that consider the subject directly. Instead, there are abundant practical attempts to apply decomposition methodology to specific, real life applications. Decomposition approaches are not the object of the current paper but it is one of our future

investigation. Decomposition alone is never enough: it is always necessary to recombine the decomposed parts. At first sight it may seem that subproblem requirements can be combined by logical conjunction, and subproblem machines by concurrent execution: subproblem domain projections need no composition because they are projections of an already composed physical reality, and software implementation demands no more than a mechanism for appropriate distribution of shared events. However, this optimistic view is far too simple. Wherever two subproblems have problem domain phenomena in common there is a potential interaction that must be appropriately handled in the composition. The composition task, then, may demand introduction of an appropriate communication mechanism, or the choice and enforcement of requirement precedence to resolve conflict, or even the recognition and analysis of the composition itself as an additional subproblem.

2.4 Behavior
In RM-ODP [7] specification, the behavior is defined as a collection of actions with a set of constraints that may occur.

2.5 Behavioral composition
In RM-ODP specification, the behavioral composition is defined as an operation that creates a new behavior from a combination of two or more behaviours. The characteristics of this new behavior depend on:

- The features of each combined behavior
- The way in which these behaviors are combined

3 Multi-Model Approaches

Separation of concerns paradigm is an essential key to ensure the smooth running of a composition process, it can be done in different manners, but with the same goal— be able to identify relatively independent “parts”, so that they can be distributed among different actors of the process, be designed independently, and at the end, be integrated in a way which allows future maintenance and evolution.
However, in this section, we present the four major multi-modeling approaches: Views modeling, Aspects Modeling, Subject Modeling and Role modeling.

3.1 Views Modeling
The concept of views has been studied in several areas related to databases, knowledge representation, modeling, programming languages and software engineering. In the field of databases, the views are operated by the query language as a selection function on the data [8, 9]. In knowledge representation, the views are used to represent the taxonomic classificatory reasoning and knowledge representation. In the views approaches [14, 15, 16], the level of decomposition is different from that adopted by the aspects approach. The decomposition is done according to the actors views. The views are developed independently of each other and without making any distinction between the basic functionality and cross-functionality.

3.2 Aspects Modeling
The Aspect-Oriented Modeling (AOM) is an approach of multi modeling based on the separation between functional concerns and preoccupations called "cross" in the software development. The idea of modeling aspects results from the AOP (Aspect Oriented Approach) and proposes to consider aspects in models [14].
The aspects approach decomposes the system into functional units and non-functional units and also separates the core functionality (or trades) of an application from the business requirements.

3.3 Subject Modeling
Subject Modeling or SOP (Subject Oriented Programming) is another separation of concern technique introduced by [10, 11]. This approach is based on a multidimensional separation of concerns, to cover different types of concerns (business, technology, business rules, etc.). It identifies a set of specifications and behaviors that reflect the perception of the real world corresponding to a generic vision of an actor. The subject approach [8] extended by MDSoC approach (Multidimensional Separation of Concerns) offers a decomposition of the system into more arbitrary dimensions, where each dimension is a collection of particular concerns.

3.4 Role modeling
The concept of role modeling comes from the need that extrinsic properties of an object can change over time [9]. Indeed, an object can be a subject of multiple classifications during its life cycle. Therefore, [8] Defines a role as a temporary viewpoint.
The decomposition in role approach aims to represent an entity of the model through multiple objects. Each object models a particular role. Unlike modeling by views, roles are objects resulting from local entities without being linked to actors.

4 Design approaches

In this part we present a n on-exhaustive set of approaches for designing infrastructure. These infrastructures allow applications to benefit from a set of technical services. The ability to design adaptable infrastructure is based on t echniques that make the infrastructure modular and compostable. These techniques are based on the separation of concern principle.

For this purpose we will go through six design approaches that have an impacts in Software Engineering in order to clarify some of their criteria. These criteria only concern the composition, they allow us to ignore the other less relevant details of the approach. The proposed criteria are the following:

- Concept: The composition is always inherent from the previous phase of decomposition. In this phase a system is divided into entities, called bean, component or bundle.
- Coupling: This property measures the degree of coupling between two units. Often we talk about strong coupling and weak coupling. In the strong coupling, it is difficult to understand the isolation units; any change may force a unit to change all the associated units; reuse of these units is difficult. However, the weak coupling overcomes all these drawbacks.
- Communication: This concept relates generally to the sending and receiving of data, events, or messages.
- Customizing modules: This criterion characterizes the ability of the approach to allow the variation of an existing module.
- Mechanism of composition: This criterion refers to the composition mechanism used in order to obtain the final composite model
- Type of composition: This criterion allows to determine the type of composition used by the approach. Referring to the previous sections, we identify two types of composition—structural and behavioral.

4.1 The modular approach

This approach is based on the module concept that is defined as a task manager (responsibility assignment) [12]. The construction of a system is to set all modules that perform different tasks. A module is characterized by the following features:

- A module is associated with a set of interfaces: interfaces expose the components (resources) provided and required by the module. These resources could be global variables or procedures with parameters and without the implementation.
- A module has an implementation portion which is a set of sub programs and data structures that are accessible through the interfaces.
- A module can be compiled separately: this allows work in parallel and allows easy replacement of a module with another in a system. Generally, a module communicates with another through the procedure calls and access to global variables declared in the interfaces of the other. The idea of the modular approach is the assembly of modules through their interfaces. This work resulted in the interconnection modules languages (MILs).

4.2 Architectural Description Approach

The Architectural Description approach is the successor of the modular approach. It focuses on the modeling of the architecture of software in terms of abstract system specification consisting primarily of functional components Described in terms of Their Behaviors and component-component interfaces and interconnections [12, 13].

In the approach of Architectural description, architectures are expressed by the architecture description languages (ADL 10) [13]. ADL provides a formal notation for specifying architectural bricks. An architectural brick is a conceptual software unit, which shows parts of a system regardless of their implementation.

The system is constructed by assembling these bricks. This will then enable the design of applications by detaching the details specific to the environment techniques.

The components are connected either by the connector or by the direct connection interface.

In the first case, the component provides and / or requires one or more ports. The connector connects the ports of components.

4.3 The software engineering component-based

The software engineering based on components (CBSE) is based on the construction of complex systems by integrating prefabricated software components. The principle of this approach is simple: "do not reinvent reuse purpose" [10]. "Components are for composition. This approach is based on the concept of Component-namely that “A component (composition) is an artifact that allows you to group and isolate a graph of objects in the model, defining explicit responsibility and needs with respect to the rest of the application, allowing it

to evolve independently [46]. This depends on the component technology, for example, CORBA or DCOM RPC use, EJB uses RMI, Web Services uses SOAP RPC-28 etc.
The composition is made by connecting each component when analyzing (declarative), when designing (scripting and programming) and at runtime (visual).

4.4 Aspects Oriented Programming
Aspect-Oriented Programming (AOP) [14] is a software development technique that aims to improve software modularity through the separation of crosscutting concerns into modular units called aspects.
In an object-oriented application, it is common for application features that they are scattered in different places and do not receive adequate encapsulation at both design models of programming languages. Such functionality is called a crosscutting concern.
Aspects Oriented Programming aims to solve this problem by proposing to write the program into two parts: a functional part that encapsulates the core business application code, and a secondary part which includes cross-functionality disseminated.
Communication is done by the event invoked implicitly calls between the core curriculum and aspects. In this approach, the basic program code and code aspects are completely separate. A third language is used to establish relationships between them.

4.5 Reflexive approach
Reflexivity is the ability of a system to reason and act on itself in its own execution [12]. A reflective system is divided into two levels: a base level that corresponds to the functional application and a meta-level corresponding to the non-functional properties.
Introspection is the ability of a program to observe its own state and therefore to reason about it. Intercession is the ability of a program to modify its own state of reification execution. Reification is the mechanism that gives the program the ability to act on its execution state.
A Meta level can be seen as the interpreter running baseline and it's indicated by the meta-objects that implement the functionality of the interpreter. The base level and meta-level are connected by a meta-link symbolizing the relationship between objects of the base level and meta-objects. This link is represented by the definition of a MOP (Meta-Object Protocol) [12, 13].

4.6 Interaction composition Approach
This approach focuses on the problem of dynamicity in the composition. It can provide an architecture for dynamic integration (the execution) of components in infrastructure such as infrastructure-based EJB.
The interaction composition is based on t he mechanisms of Behavioral fusion allows generating the behavior resulting from the interactions composition, the fusion occurs only where the interactions are enabled on a same trigger message and the fusion is a mechanism placed in the work to resolve the non-orthogonal paradigm.
Behavioral fusion is composed from rewriting rules applied on the patterns interactions and based on the semantics operators.
For simple illustration, one of the rules is that a fusion may result in parallelization of calling services. The composition in the approach oriented interactions is behavioral. The rewrite rules define the interleaving of the execution interactions. The grain of the scheduling is performed at the level of the instructions defined in the pattern of interaction.
In the implementation of this model, these rules can be executed dynamically.
The current prototype is limited in the sense that it does not offer the ability to define and apply (statically / dynamically) new fusion rules of sequential interactions; the result is a competitive runtime behavior interaction.
Therefore, there is no mechanism to set this fusion; such as execute in sequential way the behavior interactions (according to a defined order) [15].
The composition in the interaction approach is based on a composition of interaction (Called fusion) that allows the composition of the technical services.
The interaction software approach differs from other approaches in that the fusion interaction allows interleaving their executions.

4.7 Synthesis
In this part, we made a cl assification of the approaches we have studied through six criteria.
The Table I summarizes the characteristics of these approaches.
This classification will be a good way for us to create an extensive and detailed understanding about this area, thereby determining gaps by graphing and pinpointing in which research areas and for which study types a shortage of publications still exits.

TABLE I. BEHAVIORAL AND STRUCTURAL COMPOSITION APPROACHES REVIEWS

	Modular	**Architectural**	**CBSE**	**AOP**	**Reflexive**	**Interaction**
Concept	Module	Brick	Component	Aspect	Meta-Object	Model object
Coupling	Low	Low	Low	Low	Low	Low
Communication	calling function and access of global variables	defined by the types of connector and / or links	defined by the types of connector and / or links	Calling event between the basic program and aspects	Meta link symbolizing the relationship between objects of the base level and meta-objects	calling services
Customizing modules	No	No	Yes	Yes	Yes	No
Mechanism of composition	Connection of the modules after the design phase.	Connection of the components during the analysis phase.	connection of the component during the analysis phase	Weaving	static and dynamic reification	Fusion
Type of composition	Structural	Structural	Structural	Behavioral	-	Behavioral

5 Model composition approaches

5.1 Introduction

The model composition is a new research topic in the MDA. The work is ongoing development and evolution. So there is still no mature foundation to date for this. Our goal through this part is to study existing model composition approaches by analyzing and identifying 1) what are the elements involved in the composition process, and 2) how the model composition is made in these approaches. The ultimate goal is to arrive at an understanding of what is done for model composition in these approaches.

5.2 Classification criteria

The approaches mentioned below are evaluated according to several assessment criteria, which are selected based on the need to promote the reusability and the automation of model elements, as well as building a generic composition operator. However, the assessment of how a composition approach proceeds to manage conflicts and ensure model consistency during the composition process is also an important coefficient. The criteria are the following:

- The area: currently there are three major areas that are actively working on the model—Aspect-oriented modeling: The uniqueness of this area is the application of separation of concerns principle; the weaving operation which is central key in the composition process of aspect-oriented modeling and the relationship between the aspect model and the base model which is relative in most cases.
- Management Model: This criterion interested in providing a generic MDA platforms manipulation operators such as merging, comparison and conflicts.
- Metamodeling: Meta-modeling approaches allow the definition of metamodels. As the relationship between model / metamodel is relative [30], it is necessary to ensure that if these mechanisms are applicable to the meta-level, it should also be applied on the model level.
- The composition effect: It can be transformed or preserved. The compositions can transform the structure of the source models or preserve it.
- Type of composition: We identified two types of composition—by operators or relationships. By operators can be a melting, weaving or replacing the union; by relationships establish relations such as association, aggregation and inheritance. In the first type, the composition is prepared by performing the composition of operators on the source models; whereas in the second type, the source models are composed

by using the relationship in order to connect them. The difference is that the operators are not part of the final model; while relationships really are part of this model.

- Mechanism of composition: melting, replacing the union, weaving etc.
- Composition element: Defines the additional elements involved in the composition. There are two classification axes: the type and formality of these elements.
- Language of composition: The composition of elements need formalisms to express them. These formalisms are very diverse because each approach has its own elements of composition. It can be a w eaving language, a m etamodel of composition rules [31] or a UML profile for model composition.
-

5.3 Classification of Model Composition Techniques

6.3.1 AMW (Atlas Model Weaver)

AMW uses a l anguage called weaving language (weaving language) (formalism), which has a core part of providing basic generic concepts to create structural links between the models. These links are saved in the weaving patterns (weaving models) (composition element). The basic weaving metamodel AMW contains the basic weaving concepts. WElement is the basic element of all weaving metamodel elements; WModel represents the root of the weaving pattern. WLink represents links between elements of the models. WLinkEnd indicates the type of items that can be dialed [16]. This approach offers a weaving generic metamodel that defines the composition of links to a higher level of abstraction, the extension of this metamodel for defining the semantics of links depending on the application domain.

6.3.2 EMF (Eclipse Modeling Framework)

is an open source modelling framework, integrated into Eclipse. It allows to specify the structural models, single, platform independent, and from which the code can be generated. One of the supports default EMF is the generation of publisher's code for models. EMF editors are able to compose models by reference [17].

6.3.3 EML (Epsilon Merging Language)

Is a rule language based on the Epsilon platform (Extensible Platform for Integrated Specification of Languages for model management). It allows the composition of models according to different metadata models. Epsilon is a basic platform: it is a core on w hich it is possible to define models management languages, focused on specific tasks (task-specific language) such as validation, transformation, generation, comparing and merging models. [18]

6.3.4 GME

Generic Modeling Environment (GME) is a generic modeling environment that using modeling paradigms (modeling languages dedicated to DSML areas). A modeling paradigm formalizes a metamodel by defining the syntax, semantics and concrete presentation of DSML [18].

6.3.5 Kompose

Kompose is a framework that implements an aspect-oriented modeling approach for model composition through a set of guidelines. The composition process is structured in two parts: the mapping, which identifies the model elements describing the same concept and composition for creating new items. The mapping phase is based on the implementation of a set of operations that are specialized depending on the field and the composition phase is based on a conventional method that is implemented in a generic way using introspection. The elements having the same signature are compounds and their content (properties and methods) are compared in turn then compounds. The items with no corresponding are simply copied to the model compound. [19]

6.3.6 XMF-Mosaic

XMF-Mosaic (eXecutable Metamodeling Facility) is a meta modeling framework invented by the company Xactium 14. XMF provides two mapping types: unidirectional mapping and synchronized mapping. Unidirectional mapping is based on the vision processing. The unidirectional mappings take one (or a set) model (s) as input (s) and generates an output model. Unidirectional mappings are often used in code generation, for example to convert a model to Java or C ++. From the perspective of composition, we are not interested in this kind of mapping. Synchronized mapping which is a mapping for managing synchronization between two models. As we said above, the synchronized mappings can be used for several purposes; the consistency management is only one of several applications of this type of mapping. Other typical applications are synchronized mapping multiple models for the management of a system, the support of the "round trip engineering" etc. [20].

6.3.7 ECL

ECL (Epsilon Comparison Language) is a language based on rules for building links, based on the Epsilon platform. Epsilon is a platform on which it is possible to define models for managing languages, focused on specific tasks (task-specific language), such as validation, transformation, generation, comparing and merging models. ECL rule takes as input two parameters referring to the model items to compare. It is performed on the set of pairs of Meta class instances that satisfy these parameters. The body of an ECL rule consists of three parts: a comparison (compare), a compliance (conform) and an optional third party "guard" [20].

6.3.8 AML

AML (AtlanMod Matching Language) is an extension of AMW for obtaining a matching pattern. Assembles different mapping strategies that are implemented as sets of model transformations. Each of these transformations takes a set of input and produces an output matching pattern models. [20]

TABLE II. MODEL COMPOSITION APPROACHES REVIEWS

	Composition domain	**Composition Types**	**Element composition**	**Composition Language**	**Mechanism composition**	**Composition effect**	**Inputs**
AMW	Management models	by operators	Weaving and transformation Hot Model	Weaving AMW language	Weaving and transformation	structure of source models transformed	2
EMF Editor	Meta Modeling	By relationship	References	Language Reference Ecore	Establishment references	structure of source models preserved	2
EML	Management models	Specification of composition rules	Weaving	Melting language EML	Weaving	structure of source models transformed	2
GME	Meta Modeling	References	Establishment references	References of FCOs	Establishment references	structure of source models preserved	2
Kompose	Aspect oriented modeling	Specification Of compositional directives	Weaving	Fusion language kompose	Weaving	structure of source models transformed	2
XMF-Mosaic	Meta Modeling	By relations	Mappings	xSync	Establishment references And execution mapping	structure of source models preserved	2
AML	Management models	By operators	Weaving	AML Language	Weaving and transformation	structure of source models transformed	2
ECL	Meta Modeling	By rules	Validation, Transformation, Generation, Comparison	ECL Language	Comparison Conformity Guard	structure of source models transformed	2

6.3.9 Synthesis

It is clear that with the model composition there is a huge potential and a lot of possibilities that have not been exploited or even investigated until now.

The approaches mentioned above are evaluated according to several assessment criteria, which are selected based on the need to promote the reusability and the automation of model elements, as well as building a generic composition operator. However, the assessment of how a composition approach proceeds to manage environmental constraints and ensure model consistency during the composition process is also an important coefficient. The criteria are the following:

- Heterogeneity: This criterion checks the ability of an approach to stow the parameters of heterogeneous models. We consider that two specific models are heterogeneous if their modeling languages are themselves heterogeneous. Expressivity: The choice of this criterion is justified by the need to identify approaches allowing designers to express and customize different types of links.

- Reusability: This criterion is analyzed on two different planes. The foreground is used to study whether the approach provides ways for defining a g eneric part / variable in the modeling elements used. This property promotes reuse and structuring systems. The second plan involves the mechanisms offered by an approach to define generic operators' models of the composition process.
- Pre-alignment phase: This criterion verifies whether the approach implements precedent phases.

Table III summarizes the ability of approaches in expressing the criteria above.

TABLE III. THE ABILITY OF MODEL COMPOSITION APPROACHES

	Heterogeneity	Expressivity	Reusability	Pre-alignment phase
AMW	Yes	Yes	No	No
EMF editor	Yes	Yes	No	No
EML	Yes	Yes	No	No
GME	Yes	Yes	No	No
Kompose	No	Yes	No	Yes
XMF-Mosaic	No	Yes	No	No
AML	Yes	No	Yes	No
ECL	Yes	Yes	No	Yes

5.6 Discussion

The Table III shows that the fields of investigation are still very broad and the possibilities of implementing the approaches in different ways are still important. The modular approach allows to build systems by assembling modules. The idea is to connect these modules through their interfaces. The main interest of the modular approach is to facilitate the construction of systems by allowing 1) to write the module with little knowledge about the other code modules, and 2) to replace one or more modules without reassemble all the system. Therefore, with this technique the construction of the system is more understandable, manageable, and maintainable but this approach doesn't allow to personalize a module which may be a real limitation.
The Architectural Description approach focuses on designing systems by assembling architectural bricks and we can use these portions when the system is in the analysis phase to allow create the architecture description. This description can help later to make simulations on the system or allows other people to analyze the system.
The software engineering based on components approach strongly promotes the concept of composition. The applications are built by assembling components through their interfaces. The main interest of this approach is the high reuse of components but the concept of component is not clearly defined, which make difficult to define composition standards and mechanisms. Indeed, the composition of the components is a fundamental and important activity in the component approach but there are not enough research efforts for achieving the composition, compared to the efforts to define new models components.
The AOP approach [21] enables to encapsulate crosscutting code of an application in the separate software units called aspects and build these units within the core program. The interest of the AOP approach is to avoid redundant codes that frequently appear in several places in the application, and thus increase the reuse of these in different applications. By cons, he resulting code can be very complicated, difficult to understand, test and debug. However, the composition mechanisms still need lots of improvements to better manage discrepancies and conflicts during the weaving process.
After analyzing the different approaches in the context of model composition [22, 23, 24], we can notice that there is a correlation between the power of the approach and its complexity: approaches that have a satisfactory result are difficult to handle and require in most cases h uman intervention; the simplest approaches do not produce accurate results.

6 Cnclusion and Future Work

By studying the different approaches, we have been identified some criteria that influence their implementation. Some approaches deal only with the matching problem, while others feel this issue pre-board as a step in a larger process.
To synthesize, we can defined the composition as a model management operation, which generate a

single model by the combination of the contents of at least two models.

We think that this article is a good way to find a new and more difficult effective way of action. In the light of these six composition techniques, we intuitively identify four similarities as follows:

- Every technique composes a pair of models.
- Every technique proposes a m echanism for detecting similar or equivalent model element.
- Every technique proposes a m echanism that uses matching for combining models.
- Every approach proposes a m echanism of composition based on the components detected in the beginning.

So, this study allows us to realize that there are two categories of composition: the white box composition which is involved in the internal structure of the components, and the black box composition which comprises components as they are without any change. We can find it even in the model composition in which there are also two types of composition: one allows to compose component as they are, and the other composes them but after that their structure is transformed.

This part was interested in assessing different approaches by different criteria. In conclusion, each approach has its own design model for implementing services.

According to this assessment, structural composition mechanisms are clearly defined in the approaches oriented components. Behavioral composition mechanisms are very different depending on t he approaches and are based on the notion of scheduling.

As we mentioned before it is observed that some composition techniques already proposed various operations on a set of models. In special cases, reusing or adapting these techniques seems an interesting path to build a new composition model operation. Indeed several techniques of composition method have been suggested in the literature. However, there is no work that considers the coexistence of these different composition methods in order to answer practical questions such as: when should we prefer one composition method over the other? Is it possible to solve a given problem of a several composition methods?

So in our future work we will the focus on MDA approach as a whole concept through a novel methodology for automatic model composition based on the two-hemisphere model driven approach, which is an approach involved in the context of model driven architecture and proposes to create the UML class diagram from initial presentation of problem domains. The idea of the two-hemisphere-model driven approach comes from the necessity to implement the concept of separation—be able to create specifications that capture requirements in a form that is understandable by less technical stakeholders; for example, the project manager; these people were not comfortable with UML class diagrams, but were perfectly able to understand the required information represented in a simple graphical manner. Indeed, our conceptual prototype start from the point where we have several UML class diagrams made by several team development process and we need to combine them in order to have a g lobal representation of the system through one class diagram. If we take the example of two operations in two models that appear with the same signature (name, type, parameters), so to remedy this problem, it is necessary to include a step of reconciliation between the separate designs or strengthen semantics associated with the input metamodel, so that we can implement finer comparison strategies that address the behaviors described by the methods.

In this context we are working in collaboration with Riga Technical University of Latvia in this taxonomy to create a novel composer framework based on T wo Hemisphere Model Approach in order to compose several UML class diagrams. The approach is a sequence of the two-hemisphere model driven approach and answers in its turn the standards defined by MDA. The first phase will be a general analysis to build the system requirement which. The second phase is decentralized on design phase, during this phase several teams can work separately to achieve design templates for blocks belonging to the same system. The third phase is a conflict resolution phase between design models which aims to identify and treat addresses conflicts of modelling between models in the frame of "Multi-modelling paradigm" [25]. We are primarily concerned with this syntactic conflicts over naming modelling elements problems, and structural inconstancies. The last step is to merge bricks in order to achieve the overall model. The two-hemisphere model driven approach on w hich our future work is based proposes using of business process model and concept model to represent systems in the platform independent manner and describes how to transform these models into UML diagrams. The strategy supports gradual model transformation from problem domain models into program components, where problem domain models reflect two fundamental things: system functioning (processes) and structure (concepts and their relations).

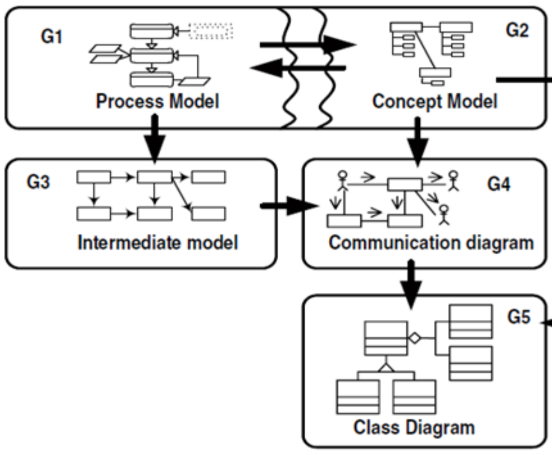

Fig. 1. Transformations from two hemisphere model into class diagram under two hemisphere model driven approach [47]

As shown in Figure 1 the two-hemisphere model driven approach proposes to start process of software development based on the representation of problem domain by two models, where one model reflects functional (procedural) aspects of the business and software system, and another model reflects the corresponding concept structures. The co-existence and inter-relatedness of these models enables use of knowledge transfer from one model to another, as well as utilization of particular knowledge completeness and consistency checks [26].

Therefore, when the models are small enough and developed by a single or a couple of designers, they can be composed manually. However, in most cases, the models are too large to be composed manually and it's necessary to develop an automatic composition method to ensure that all the elements in the model are handled. Indeed, our methodology can be used to build the model for a large system, where the modellers identify different class diagram. However, they model each piece separately to deal with complexity. Once all these models have been correctly built in isolation, it is necessary to compose them, however four main ideas for composition are identified:

- Better understand the interactions between the elements to compose: Model comparison.
- Match equivalent: weaving.
- Analyze interactions to identify conflicts and undesirable emergent behaviors: Repository for conflicts management.
- Check the global consistency of the system's model (as shown in Fig.2).

Our future approach intend to create the interactions of structural and behavioral type of composition: when there is a structural dependency between two components, a link is created between the interfaces and when there is a behavioral dependency between two components, an order relation can be applied.

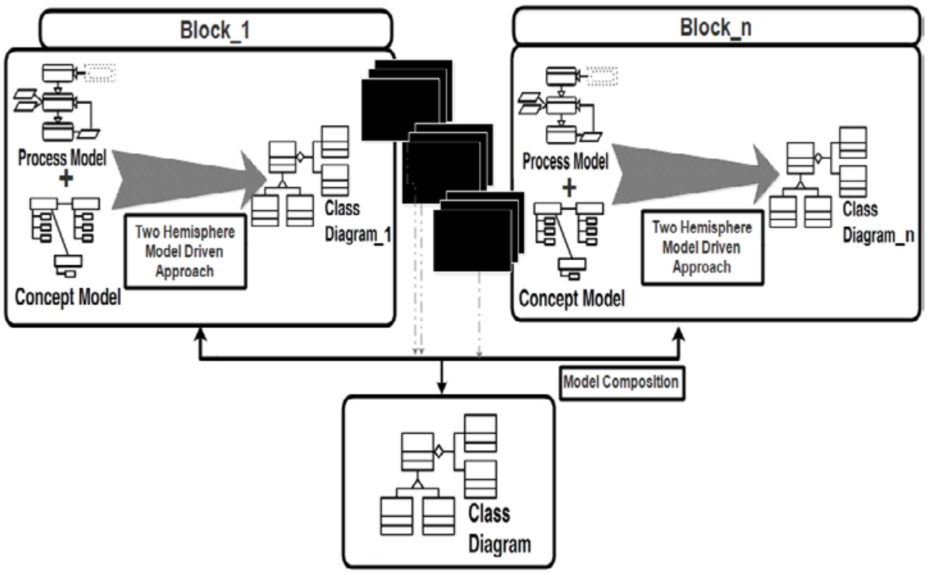

Fig. 2. The application of model composition on two hemisphere model driven approach

The definition of these interactions makes possible to have a high degree of composability. Furthermore, considering the change in a level of abstraction as a criteria of comparison, in Two Hemisphere Model Approach the transformation is vertical because it causes a ch ange in level of abstraction, it is the case of refining PIM to PSM in MDA, and in our composition approach the transformation will be horizontal because it includes changes designed to incorporate models from multiples sources.

References:

[1] OMG: Object Management Group – MDA (Model Driven Architecture) Guide Version 1.0.1; 2001 Available at (http://www.omg.org/mda/)

[2] Oksana Nikiforova Two Hemisphere Model Driven Approach for Generation of UML Class Diagram in the Context of MDA- e-Informatica Software Engineering Journal, Volume 3, Issue 1, 2009

[3] [24] Nikiforova O., Kozacenko L., Ahilcenoka D., Gusarovs K., Ungurs D., Jukss M., Comparison of the Two-Hemisphere Model-Driven Approach to Other Methods for Model-Driven Software Development, Scientific Journal of Riga Technical University: Applied Computer Systems, Grundspenkis J. et al. (Eds), Vol.18, 2015, pp. 33-42

[4] G.KICZALES."Aspect-Oriented Programming". European Conference on Object-Oriented Programming (ECOOP), Springer-Verlag LNCS 1241, Finland, June 1997.

[5] Fischer, B., "Decomposition of Time Series - Comparing Different Methods in Theory and Practice", Eurostat Working Paper, 1995.

[6] Fredrik Milani et al, "High-order statistics in global sensitivity analysis: Decomposition and model reduction-Computer Methods in Applied Mechanics and Engineering", Volume 301, 1 April 2016, Pages 80-115

[7] ISO.ITU/ ISO Referebce Model of Open Distributed Processing –Part 2: Foundations, International Standards ISO/IEC 10746-2,ITU-T Recommendation X.902.Technical report ISO, 1995.

[8] M. Acher, P. Collet, P. Lahire, and R. France. Comparing approaches to implement feature model composition. Modelling Foundations and Applications, pages 3–19, 2010.

[9] A. Anwar, S. Ebersold, B. Coulette, M. Nassar, and A. Kriouile. A rule-driven approach for composing viewpoint-oriented models. Journal of Object Technology, 9(2):89–114, 2010.

[10] P.L. Tarr, H. Ossher, W. Harrison, M. Stanley, Jr. Sutton. "N Degrees of Separation: Multi-Dimensional Separation of Concerns". International Conference on Software Engineering, pp. 107-119, 1999.

[11] H. Ossher, P. Tarr. "Using multidimensional separation of concerns to (re)shape evolving software". Communications of the ACM, Vol. 44, No. 10, pp. 43-50, October 2001.

[12] J. Klein. "Behavioral Aspects and weaving". Thesis of Rennes University, December 2006.

[13] S. Clarke. "Composition of Object-Oriented Software Design Models". PhD thesis, Dublin City University, 2001.

[14] G. KICZALES. "Aspect-Oriented Programming". European Conference on Object-Oriented Programming (ECOOP), Springer-Verlag LNCS 1241, Finland, June 1997.

[15] L. Cavallaro, E. Di Nitto, C.A. Furia, and M. Pradella. A tile-based approach for self-assembling service compositions. In Engineering of Complex Computer Systems (ICECCS), 2010 15th IEEE International Conference on, pages 43–52. IEEE, 2010.

[16] Eclipse AMW plugin, In: Eclipse Modeling Symposium, Eclipse Summit Europe 2006 (Esslingen, Germany), 2006.

[17] C. Clasen, F. Jouault, J. Cabot, et al. Virtual Composition of EMF Models. In 7èmes Journées sur l'Ingénierie Dirigée par les Modèles, 2011.

[18] M. Dahchour, H. Rayd, Y. Lakhrissi, A. Kriouile, "Extension d'UML par les rôles". Proc. of the 9th Maghrebian Conference on Information Technologies (MCSEAI 2006), Agadir, Morocco, December 2006.

[19] F. Fleurey. Kompose: a generic model composition tool. 2007. Available from: http://www.kermeta.org/kompose/

[20] Jeanneret, C., France, R., & Baudry, B. (2008, April). A reference process for model composition. In Proceedings of the 2008 AOSD workshop on Aspect-oriented modeling (pp. 1-6). ACM.

[21] Reddy, Y. R., Ghosh, S., France, R. B., Straw, G., Bieman, J. M., McEachen, N & Georg, G. (2006). Directives for composing aspect-oriented design class models. In Transactions on Aspect-Oriented Software Development I (pp. 75-105). Springer Berlin Heidelberg.

[22] Alberto Rodrigues da Silva, Model-driven engineering: A survey supported by the unified conceptual model- Computer Languages, Systems & Structures, Volume 43, October 2015, Pages 139-155

[23] D. Calegari, N. Szasz, Institution-based foundations for verification in the context of model-driven engineering, Science of Computer Programming. 107 (2015) 41–63.

[24] D.Calegari et al, Heterogeneous verification in the context of model driven engineering-Science of Computer Programming, In Press, Accepted Manuscript, Available online 26 February 2016.

[25] G.Coutinho Sousa Ferreira et al, On the use of feature-oriented programming for evolving software product lines — A comparative study-Science of Computer Programming, Volume 93, Part A, 1 November 2014, Pages 65-85

[26] Hanine Tout-AOMD approach for context-adaptable and conflict-free Web services composition Computers & Electrical Engineering, Volume 44, May 2015, Pages 200-217