Advances in Science, Technology and Engineering Systems Journal Vol. 4, No. 6, 147-157 (2019)
www.astesj.com
Special Issue on Multidisciplinary Sciences and Engineering

ASTESJ
ISSN: 2415-6698

Coordination between Heterogeneous Models Using a Meta-model Composition Approach

Naima Essadi*, Adil Anwar

SIWEB, E3S, EMI, Mohammed V University in Rabat, Rabat, Morocco

A R T I C L E I N F O

Article history:
Received: 17 September, 2019
Accepted: 26 October, 2019
Online: 25 November, 2019

Keywords:
Coordination
Transformation
Composition
DSML
Heterogeneity
Model
Meta-model
Offered Interface
Required Interface
Realization
Referencing
Relationship
Bridge Pattern

A B S T R A C T

The technological advance as well as needs of human beings made that systems became more and more complex. In contrast, the use and creation of new modelling languages became simple and no more reserved for a handful of language experts. Consequently, many new practices of systems implementation emerged, among them, the use of different domain specific modelling languages (DSMLs) to represent the same system. Indeed, complex systems are composed by many components sometimes belonging to various domains. Thus, many teams of experts collaborate to develop such systems. Moreover, teams tend to use different DSMLs to design their concerns. This new practice generates an accidental heterogeneity due to production of various heterogeneous models representing a same system. However, those heterogeneous models need absolutely to be coordinated to facilitate communication between stakeholders and of course to ease implementation and validation of systems. This paper proposes a composition interface–based approach to coordinate and integrate heterogeneous DSMLs in order to coordinate their models. The proposed composition interface is defined according to Bridge Design Pattern. To illustrate this approach two DSMLs are used: An Indoor Service Transport Modelling Language and an Internet of Things Modelling Language.

1. Introduction

Model Driven Engineering (MDE) has as main goal a deep separation between business and technological concerns. It makes models in center of software and systems engineering. Hence, it provides concepts and tools assuring to bridge the gap between problem-level abstractions and implementation-level concepts.

However, the emergence complexity of nowadays systems raises numerous new conception and implementation challenges. Indeed, these systems involve many different domains. Consequently, many teams of experts contribute to implement a same system.

Moreover, the need and the large use of MDE as well as the popularization of software language engineering (SLE) induce the new practice of using specific hand maid modelling languages baptized Domain Specific Modelling Languages (DSMLs)

*Corresponding Author: Naima Essadi, SIWEB, E3S, EMI, Mohammed V University in Rabat, Rabat, Morocco, Email: naimaessadi@gmail.com

Recently, the use of DSMLs rather than Unified Modelling ones increases due to many reasons. Actually, Domain Specific Modeling languages are expressive and allow a concise and accurate specification in addition of a high level of abstraction [1, 2]. These highlights have a direct influence upon productivity and costs. Indeed, the use of domains vocabulary means less time and effort in modelling phase. It also means a high quality of communication between stakeholders thereby decreasing error rate and modelling iterations.

Actually, every domain has specific vocabulary, concepts and paradigm. This difference implies the use of a different DSML for every different domain involved in the same system.

Consequently, the use of multiple DSMLs to design systems induces an accidental heterogeneity. Actually, as result we get many heterogeneous models for a same system as illustrated by Figure 1. The diagram of Figure 2 summarises problematic causes and effect.

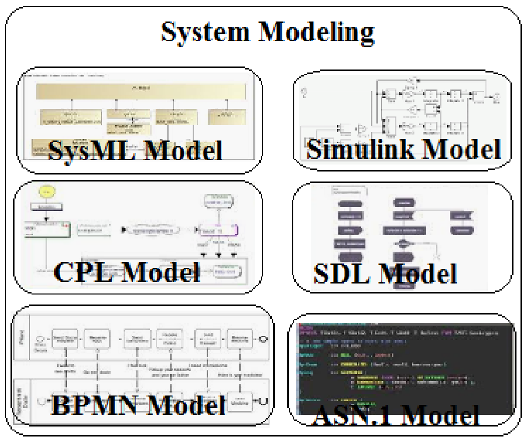


Figure 1: Heterogeneous Models for a same system

These models need inevitably to be coordinated and integrated to get a whole view of systems and, by the way to ease implementation and validation. Coordination between these models is also motivated by the need of getting relevant information scattered in different models as well as the concern to maintain the consistency of systems in case of evolution.

Many leads had been explored by current researches to resolve this issue [2, 3, 4, 5]. Specifically, this work fits in researches operating at meta-level [6, 7, 8, 9]. Actually, we choose to coordinate DSMLs used to elaborate heterogeneous models rather than coordinating models themselves. The need of a generic, reused and less error prone approach motivated this choice.

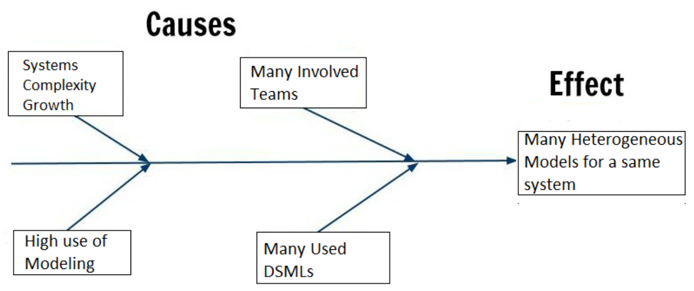

Causes
Systems Complexity Growth
Many Involved Teams
Effect
Many Heterogeneous Models for a same system
High use of Modeling
Many Used DSMLs

Figure 2: Causes effects diagram of current problematic

Adopting interface concept to coordinate the use of multiple DSMLs has been proposed by few interesting works [10, 11, 8, 12, 13]. Some of this works didn't give clear specification of how to define interfaces for languages, others didn't give any concreate process or tool to fulfil the proposed approach. Furthermore, those approaches didn't explain how to deal with pre-existent models.

Therefore, in this paper, we introduce a composition interface-based approach to coordinate heterogeneous DSMLs. This approach involves the application of a process in two steps: Coordination Meta-Model (CMM) elaboration and Models coordination. While, the first step aims to define interfaces for involved DSMLs and their composition to get a coordination meta-model (CMM), the second one involves the import of heterogeneous models in a model conforming to CMM. This import enables coordination between pre-elaborated heterogeneous models.

Our approach is in fact a structural interface-based composition of pre-elaborated models. It first gives a global view of system on hand and secondly affords the possibility to exchange important information between heterogeneous models as well as insuring their interoperability.

The remainder of this paper is organized as follows: we begin by giving an overview for DSMLs specification as well as heterogeneity in section 2. In Section 3, a motivating example is presented to advocate the need of resolution. Then, in following section, we introduce interface-oriented design basics for languages as well as Bridge Pattern Design. Section 5 illustrates the applicability of the approach using the motivating example where heterogeneous models are coordinated. Section 6 presents related works and finally, section 7 concludes this paper.

2. Heterogeneous DSMLs

2.1. DSMLs Overview

Domain Specific Modeling Languages aims to abstract problems and solution using domains vocabularies and concepts instead of generic and unified objects and shapes.

Contrary to a General-Purpose Modeling Language like UML, a DSML is tailored to a specific specialized domain. It allows stakeholders to contribute in modeling using notations close to their knowledge of respective domains [1]. DSMLs could be visual such as SDL, IFML, BPMN or textual like Data Modeling Languages: Abstract Syntax Notation (ASN.1) [14, 15], YANG [16, 17] and others.

DSMLs are first of all languages and consequently adhere to languages norms. Commonly, languages are specified using grammars with the famous Context-Free Grammars EBNF (extended Backus-Naur Form) [18]. Indeed, many other forms of specification exist [10] such as: Attribute Grammars [19], Graph Grammars [20], UML Profiles [21] and Meta-modeling [22]. Specifically, this work emphasizes meta-modeling specification.

2.2. DSMLs Specification

DSMLs are defined by two manners: white box specification and black box specification. The white box provides all necessary information about the language: the exhaustive list of including concepts with their allowed inter-relations as well as their concrete representation and optionally semantics giving the related meaning.

A white box classical specification reflects the language complexity. It is for example used to elaborate a compiler for the language or either to integrate languages to improve their expressiveness. Many recent researches use this specification to reuse existing DSMLs. Table 1 summarizes elements of white box specification.

However, a DSML black box specification is optional and aims to hide complexity and irrelevant information about a language. It could be defined by providing offered and required interfaces for a language. Table 2 describes offered and required interfaces.

Table 1: DSML White Box Specification

Parts	Relevance	Description
Abstract Syntax	Required	Provides concepts of a language with their exact rules of combinations [23].
Concrete Syntax	Required	Provides textual or graphical notation and symbols of language concepts [10]. A language could have one or more concrete syntax [23].
Semantics	Optional	Provides meaning and subjective understanding of a language expressions and concepts combination [10, 23]. Many types of semantics exist: Translational, Operational Semantic, Denotational and Extensional [23].
Semantics Mapping	Optional	Provides correspondences and relations between Semantic and Abstract syntax elements [23].
Syntactic Mapping	Optional	Assigns syntactic constructs to the abstract syntax [24]

The definition of interfaces for a DSML allows dealing with the language without knowing all detailed information and specification about it. The black box specification gives just restricted and needed elements. Actually, the definition of a DSML as a black box exposing specific interfaces allows language reuse in different contexts. Hence, a language could be substituted by another as long as it respects the same interface contract.

Table 2: DSML Black Box Specification

Parts	Relevance	Description
Offered Interface	Optional	Exposes elements to be used or referenced by other languages.
Required Interface	Optional	Represents elements and references needed by the language from other languages.

2.3. Heterogeneous DSMLs

We qualify, as heterogeneous, elements with different nature. In MDE, the heterogeneity of models is usual due to various domains, tools, concepts and paradigms of modelling. Many types of heterogeneity have been identified [2, 3, 25]. Nevertheless, we notice three levels of heterogeneity between models:

- Heterogeneity of points of views: includes models with different purposes and scopes. For example a model representing static point of view is considered as heterogeneous with a model describing a dynamic point of view or a requirements model.

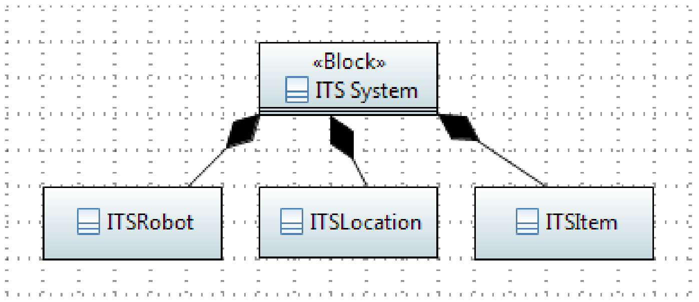
- Heterogeneity of meta-model (language): means that models are elaborated with different DSMLs. We can also say that they are conform to different meta-models, e.g., language heterogeneity between a BPMN model and an SDL model.

- Heterogeneity of meta-meta-model means that models are elaborated with different DSMLs, and those DSMLs themselves are specified using different meta-models. An example of that form of heterogeneity could be a CPL model elaborated using CPL DSML conforming to GOPPRR [26] meta-meta-model, the second model could be an SDL model elaborated using SDL DSML which is specified with Ecore meta-model.

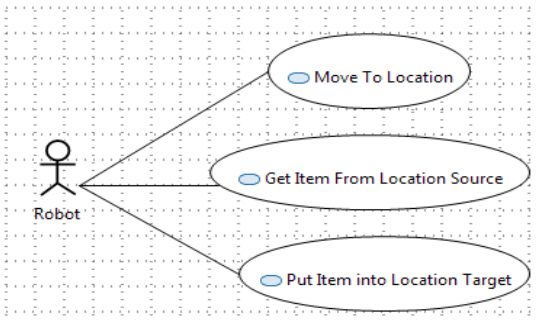
In this work we are concerned by heterogeneity of DSML which is the second level of heterogeneity. The first and third levels of heterogeneity are beyond the scope of this paper.

3. Motivating Example

In this paper we illustrate our approach using an Indoor Transport Service System (ITS). The ITS system is composed by Robots, Locations and Items as described by diagram in left of Figure 3. Robots execute tasks within a defined world [27]. It transports Items from source to target locations. This kind of systems is of course very useful and is more and more used for different purposes: In hospitals for providing patient rooms with medications and medical supplies, in restaurants to deliver ordered meals to customers, etc.



(a)



(b)

Figure 3: ITS System Main Elements and Features

However, the ITS system needs to be supervised. System users need to be informed about Robot's location and activity while accomplishing their transportation service. They also need to get informed about item's location in real time. A good solution for Robot and Items supervision is to consider them as connected objects.

This means that we need to compose our ITS system with an Internet of Things system.

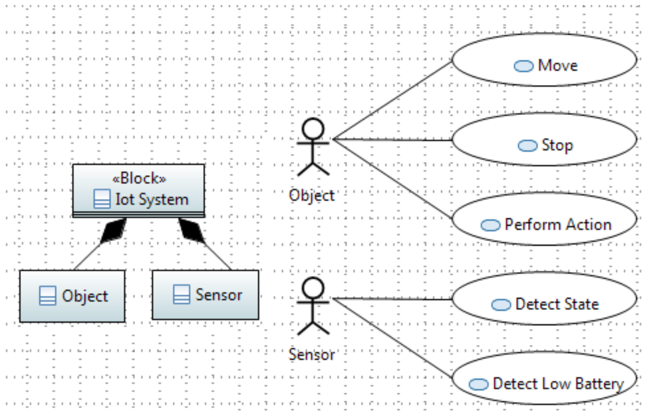


Figure 4: IoT System Main Elements and Main Features

An IoT system as described by left part of Figure 4 is composed by Objet and Sensor. Many kinds of sensors exist: sensors to detect state change, location or battery thresholds. A sensor is related to an object to detect events over them. The right part of Figure 4 exposes object and sensor use cases. Sensor detects event over object that performs actions. Consequently, in this case study, we will deal with two heterogeneous domains: ITS domain and IoT Domain. Subsequently, the composition of these two systems will involve two experts' teams: ITS team and IoT team. Every team needs a dedicated DSML for modelling its concerns, ITS DSML and IoT DSML for instance.

3.1. ITS DSML

We have elaborated a specific DSML for the ITS System. This DSML uses domain vocabularies and concepts. In this section, we will give an excerpt of the white box specification of this language. The abstract syntax is illustrated in Figure 2 and described by semantics column of table 3. The graphical representation of DSML concepts and semantics are given in Table 3.

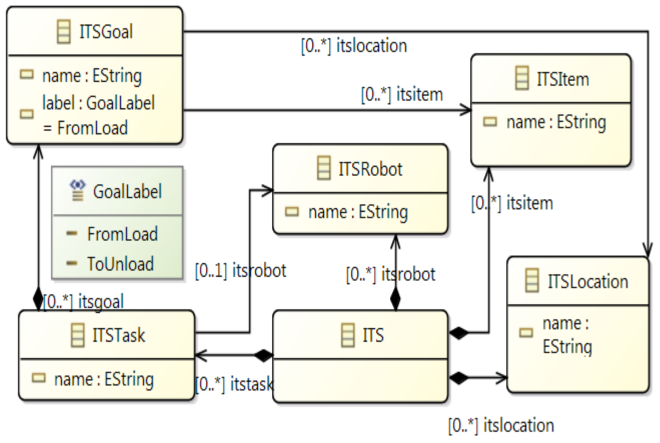


Figure 5: ITS DSML Abstract Syntax

As we can see in Figure 5, ITS DSML defines following new concepts: ITSRobot, ITSTask, ITSGoal, ITSItem and ITSLocation. ITSRobot has to achieve ITSTasks. Every ITSTask is composed by two ITSGoals: an ITSGoal to get an ITSItem from an ITSLocation source and an ITSGoal to put an ITSItem into an ITSLocation target.

Table 3: ITS DSML Concrete Syntax and Semantics

Abstract Concepts	Graphical Representation	Description
ITSRobot		Concept responsible for indoor transportation.
ITSGoal		Concept describing a mission to be executed by robot. It includes an item as well as two locations source and target.
ITSTask	Task	Concept regrouping a list of goals to be executed by a single robot.
ITSItem		Concept representing elements transported by robots.
ITSLocation		Concept representing site of departure and end points for robots missions.
Relation ITSTask-ITSRobot	AssignedTo →	Concept relating an ITSTask to an ITSRobot
Relation ITSGoal-Location	ToUnload → FromLoad →	Concept relating a goal to Location concept.
Relation ITSGoal-Item	ToUnload → FromLoad →	Concept relating a goal to Item concept

We have used Obeo Designer [28] to create and design a new DSML for ITS domain. Obeo Designer is an eclipse modeling tool that enables the creation of new DSMLs and their editors. It is based on the frameworks EMF, GEF [29] and GMF [30].

Figure 6 illustrates the elaborated ITS language editor. It illustrates also a model of this DSML produced using the graphical editor. In this model the instance "Hospital-Robot" of ITSRobot has the ITSTask "Hospital_Task" as mission to achieve. This task is composed by two ITSGoals: a goal "Source" to load ITSItem "oxygenBottle" from ITSLocation "Storage" and a second ITSGoal "Target" to transport "oxygenBottle" to "PatientRoom".

A second model of this DSML is given in Figure 7. This model concerns food delivery where an "FooDeliveryRobot" has to accomplish the ITSTask "Pizza Delivery Task" of delivering ordered food "Pizza" to a specific ITSLocation "CustomerAddress".

3.2. IoT DSML

We have also elaborated a second DSML to cover internet of things domain [25]. This DSML introduces several basic concepts of IoT domain such as: Object, Sensor, Event and Action concepts. Sensors are associated to real word objects to detect events over them. Subsequently, actions are triggered according to detected events. The abstract syntax of IoT DSML is illustrated in Figure 8.

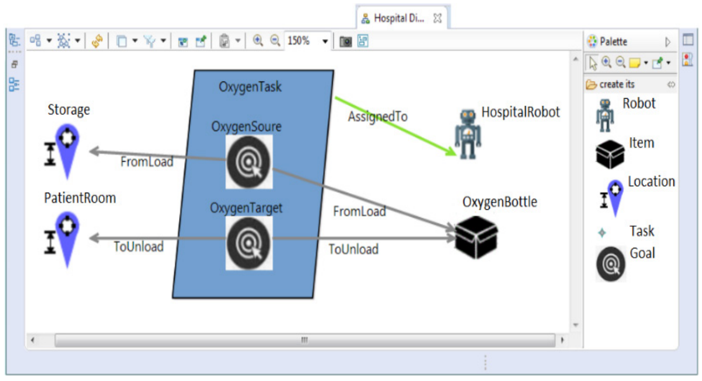


Figure 6: ITS Model for Hospital Use

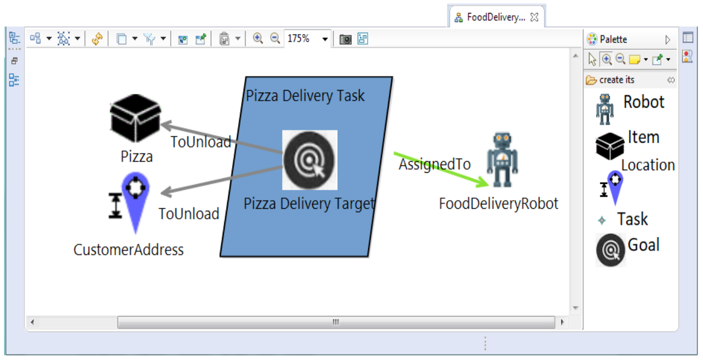


Figure 7: ITS Model for Food Delivery

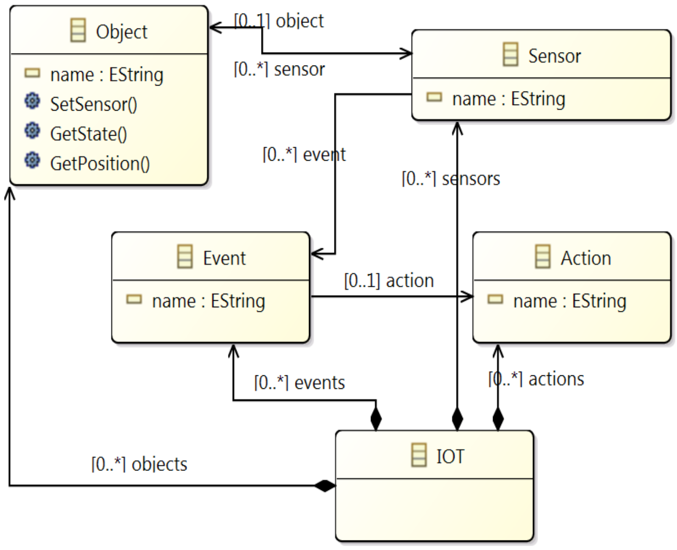


Figure 8: IoT DSML Abstract Syntax

The graphical representation and semantics of this DSML are described in Table 4.

An IoT model is illustrated in Figure 9. In this example, the instance of Object "Device" has two sensors "SpeedSensor" and "LocationSensor". The "SpeedSensor" detects event "Moving" over "Device" which trigger the action "SendSms". However, the sensor "BatterySensor" detects the event "LowBattery" and then triggers the action "SendSms".

Table 4: IoT DSML Concrete Syntax and Semantics

Abstract Concepts	Graphical Representation	Description
Object		Concept representing connected objects.
Sensor		Concept sensor responsible of detecting events over connected objects.
Event	🔔	Concept event detected by sensors over objects.
Action		Concept describing action done when a sensor detects an event or an environment change.
Relation Event-Action	→	Concept relating an Event to a triggered Action.
Relation Sensor-Event	→	Concept relating a Sensor to detected Event.

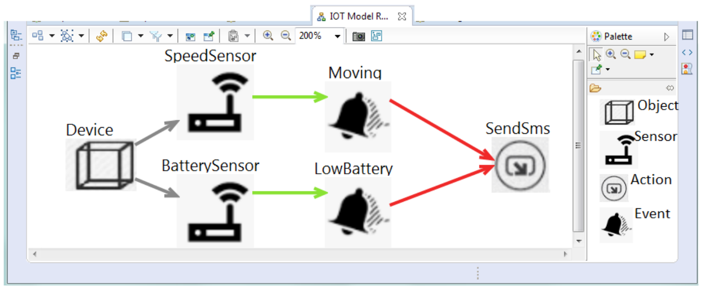


Figure 9: IoT DSML Model and Editor

A second model for IoT is given by Figure 10. The instance "Building" has "PositionSensor" as sensor. The action "SendSms" is triggered when the event "PositionChanged" is detected on "Building" Object.

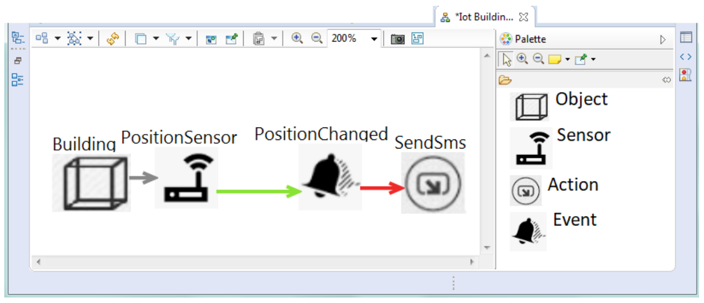


Figure 10: IoT DSML Second Model

3.3. ITS and IoT DSMLs Coordination Motivations

Both models given in Figure 6 and Figure 7 need to be related to model in Figure 9 for the following purposes: (i) To get activity and position information of "Hospital_Robot" and "FooDeliveryRobot". (ii) To get items' position and situation of orders. However, coordinating ITS and IoT DSMLs is more beneficial than coordinating their models. As, the coordination at language level holds to all its models [10]. It actually guarantees less elaboration time, less effort and less error prone. The coordination is done once instead of for every model.

Moreover, this operation is validated once for all which decreases bugs occurrences and coordination iterations. However, to achieve coordination we need to answer the following questions:

How can we coordinate the two DSMLs to get a global view of this system?

How can we coordinate the two DSMLs to achieve the whole system goals described by (i) and (ii)?

How can we represent and describe coordination between the two DSMLs?

4. DMLs Coordination

The recent design practices, for instance the use of different DSMLs to describe domains concerns, induce an accidental heterogeneity. Indeed, system designers from different teams produce heterogeneous models to describe a single system. These models are heterogeneous as they are expressed in different DSMLs. Consequently, coordination between involved DSMLs is needed to facilitate systems elaboration.

Actually, coordination could be seen as a combination between languages [10] to work together. It aims to join DSMLs's separate concepts to achieve a common goal. This join could however be structural to get hole static and global view of systems and either to get needed information and data among systems. On the other hand, the join could also be behavioral to achieve simulation to validate systems features and execution in early implementation stages.

Furthermore, coordination is considered as a less invasive form of integration [31]. It is done posteriori to get global analysis or global simulation of a system [32]. For same purposes, many interesting works [33, 6, 7] invoked instead globalization concept. This term is used as analogy with world globalization relationships between countries to regulate interchange, interaction and communication [6].

4.1. Interface-Based Coordination Approach

We propose to consider heterogeneous meta-models as component Figure 11. We relate them using interfaces considered as coordination points. Interfaces must be defined and related internally to their own meta-models and externally to other heterogeneous meta-models using coordination relationships.

4.2. Interface Description

DSMLs interfaces are part of black box modeling languages specification. It aims to define concepts exposing possible coordination join points and relationships to be established between languages.

Indeed, the use of an offered interface of a DSML1 by a DSML2 supposes that DSML1 exposes an offered interface and in the other side the DSML2 in its turn exposes a required interface.

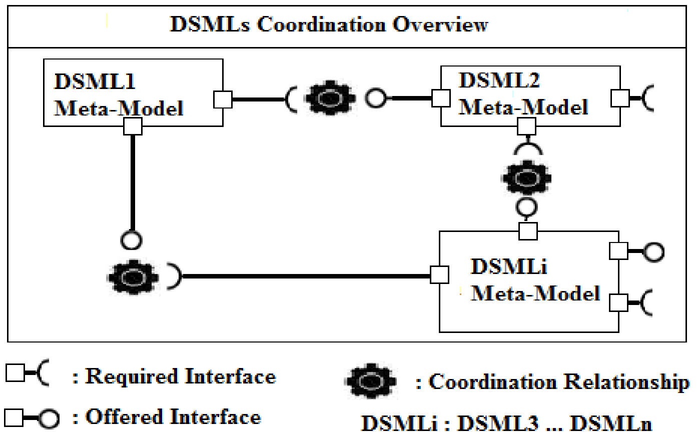


Figure 11: Heterogeneous DSMLs Coordination Overview

Our idea is described in Figure 11, where we define interfaces over DSMLs abstract syntax (meta-model). These interfaces are exposing concepts to other DSMLs and hiding details about exact structures and implementations.

In coordination, a DSML could be passive or active. The active DSML is the one exposing a required interface and thus uses interfaces offered by other DSMLs qualified as passive.
In this paper we consider external coordination between involved DSMLs. The external coordination has been defined in our previous work. It is done externally and is assured by a specific framework or workbench.

Many recent works [11, 8] discussed definition of DSMLs interfaces for various needs. However, they gave different proposition for defining it. In this work, we propose to create interfaces according to the Bridge Design Pattern [34].

4.3. Bridge Design Pattern Overview

Bridge Design Pattern has been first introduced by E.Gamma et al [34]. This design pattern aims to decouple an abstraction from its implementation so that the two can vary independently [34].

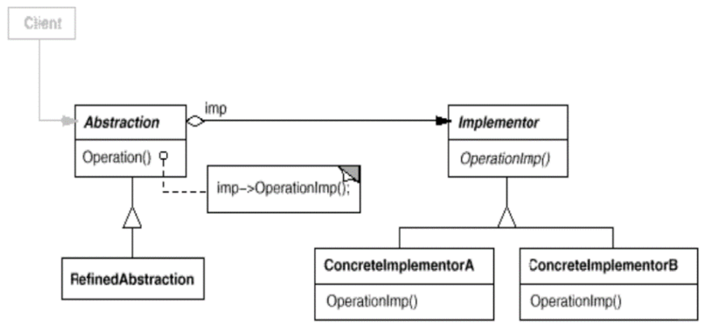
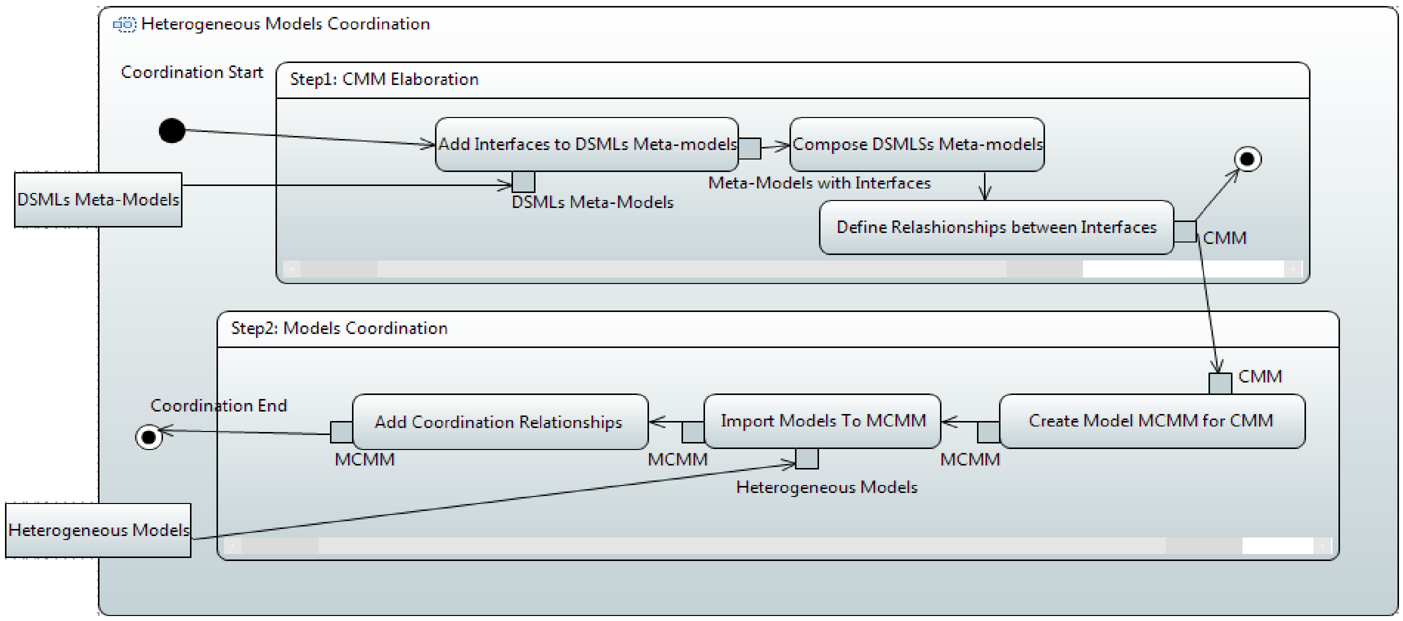


Figure 12: Bridge Design Pattern Concept [34]

The main idea of this design pattern illustrated in Figure 12 consists in the definition of an abstract class in the first part ("Abstraction" of Figure 12). This abstract class is inherited by the class requiring a specific implementation ("RefinedAbstraction" of Figure 12).


Heterogeneous Models Coordination
Coordination Start
Step1: CMM Elaboration
Add Interfaces to DSMLs Meta-models
Compose DSMLSs Meta-models
DSMLs Meta-Models
DSMLs Meta-Models
Meta-Models with Interfaces
Define Relashionships between Interfaces
CMM
Step2: Models Coordination
CMM
Coordination End
Add Coordination Relationships
Import Models To MCMM
Create Model MCMM for CMM
MCMM
MCMM
MCMM
Heterogeneous Models
Heterogeneous Models

The defined Abstract class ("Abstraction") references an interface defined in the other side (see "Implementor" in Figure 12) which is implemented by classes of this same side (see "ConcreteImplementorA", "ConcreteImplementorB" in Figure 12). In case of DSMLs, we propose to define interfaces according to the Bridge Design Pattern.

Consequently, the abstraction part of the Bridge is the required interface and is defined by the first DSML while the second DSML defines the second part of the Bridge which is the interface to be referenced. This interface is exactly the offered interface of the second DSML. This means that an abstract meta-class must be defined in the first DSML. This meta-class is the super meta-class for meta-classes requiring coordination. While this meta-class represents a required interface for the first DSML, an interface is defined in the second DSML to represent its offered interface. This interface is implemented by elements of the second DSML.

The use of this pattern gives a generalized nature to coordination. Indeed, one of the two coordinated DSML could be easily replaced by another DSML as well as interfaces contracts are respected. In the proposed coordination process, a DSML could be coordinated to as many as needed DSMLs. It can be active or passive or the two at the same time.

4.4. Coordination Process Description

We introduced in Figure 11 a high-level overview of our approach. In this paragraph we propose to use this approach as a part of a coordination process composed by two main steps: CMM Elaboration and Models Coordination Figure 13.

"CMM Elaboration", represented by Fig. 13 and Fig. 14, is done at DSML level. This step is done in three stages. We begin by adding interfaces to involved DSMLs's meta-models and then we compose them to be able to define coordination relationships between their interfaces.

Interfaces are added using a transformation, while composition is done by creating a new element to be the root element of the CMM. The composition consists in including DSMLs's meta-models root elements in this new element.

We assume that in a meta-model every meta-class must have at most a parent. At least one meta-class doesn't have a parent. This meta-class is considered as the root element of a meta-model.

Subsequently, the result of this first step is coordination meta-model (CMM) that has a root element and coordinated meta-models as leaves

The second and final step is "Models coordination" illustrated by Figure 13 and Figure 15. The second step in this process enables us to achieve coordination.

We start by creating a new model MCMM conforms to CMM. The former will be composed by all input models and their mutual relationships. We first import input heterogeneous models then we relate input models interfaces using relationships defined in step 1.

The import operation is in fact a transformation that aims to transform input models to models that are conform to CMM. The result of this step is a model representing a global view of the whole system as well as its inter-relationships.

5. Application: Connected Indoor Transport Service System

To demonstrate the applicability of the proposed approach, we use the motivating example described in section 3. Actually, in this example we have two heterogeneous DSMLs to coordinate in order to get activity, position and battery threshold of ITSRobot. This coordination aims also to get position of Item and Room elements.

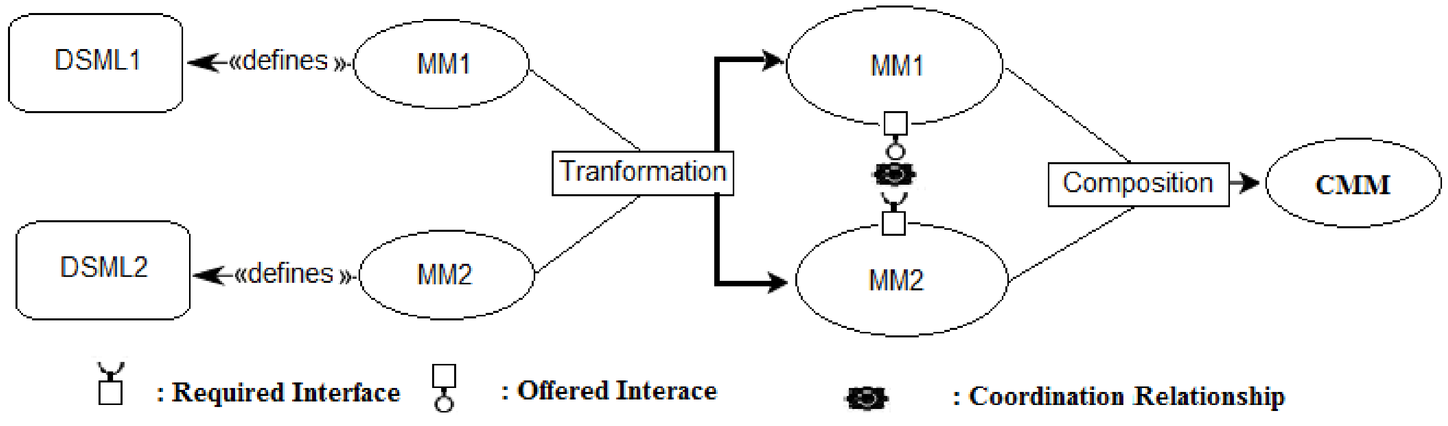


Figure 14: Step1: Coordination Meta-model (CMM) Elaboration

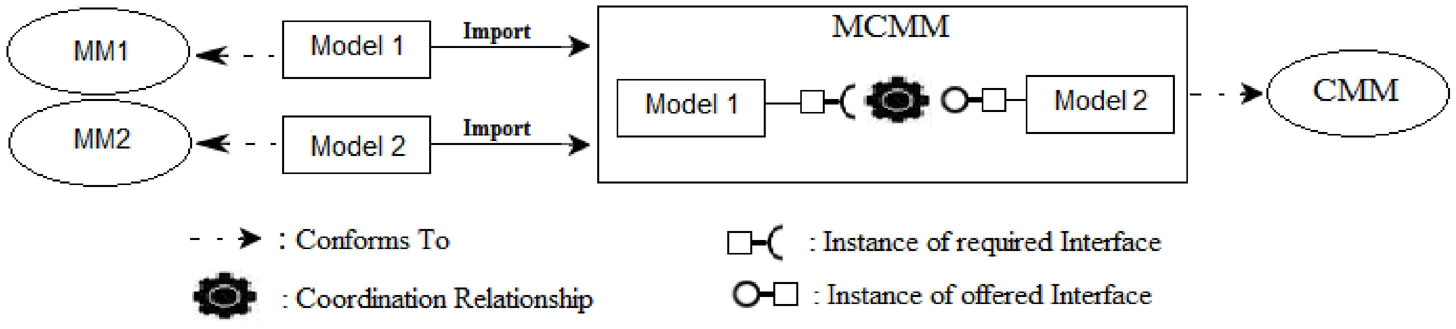

MM1
MM2
Model 1
Model 2
Import
Import
MCMM
Model 1
Model 2
CMM
: Conforms To
: Coordination Relationship
: Instance of required Interface
: Instance of offered Interface

Figure 15: Step2: Models Coordination

5.1. ITS DSML Required Interface

The main feature of ITS DSML is the modeling of an Indoor Transport Service.

However, ITS team needs to get a real time activity of ITSRobot as well as position and battery thresholds. They also need to get location of Item and Room concepts. As said earlier, needs of a DSML are translated to required interface, for that we propose to define a required interface for ITS DSML. This interface is represented by the abstract meta-class named "ConnectedElement" see Figure 16.

According to Bridge Design Pattern, this meta-class is considered as super class of ITSRobot, Item and Room concepts. Moreover, this meta-class must use an element that provides its needs. The used element must be defined by another DSML offered interface. In the current example, the IoT DSML represents the other language.

5.2. IoT DSML Offered Interface

The IoT DSML offers connecting ability to objects by relating them to sensors able to detect events upon them. Subsequently and according to Bridge Pattern design, we propose to define the interface "IObject" as offered interface see Figure 17. This interface is implemented by Object concept. Actually, referencing "IObject" enables connection to sensors.

5.3. CMM Elaboration

The coordination meta-model (CMM) of ITS and IoT DSMLs assures the composition of the two languages (Figure 18).

It has as root "ITS_IOT" element that contains involved languages root elements "ITS" and "IOT" for instance.

According to proposed approach, the coordination between the two languages must be done by defining coordination relationships between both required interface of ITS and offered interface of IoT, described in earlier paragraph. Subsequently, 'ConnectableElement' the super class of both 'ITSRobot', 'Item' and 'Room' must reference the interface 'IObject' of IoT DSML. This later is implemented by 'Object' meta-class belonging to IoT DSML.

While coordination elaborated between ITS and IoT DSMLs is qualified as external [25], we notice that relationship used between those languages interfaces is a composition relationship and more specifically a referencing relationship [25]: "ConnectedElement" references "Iobject".

While coordination elaborated between ITS and IoT DSMLs is qualified as external [25], we notice that relationship used between those languages interfaces is a composition relationship and more specifically a referencing relationship [25]: “ConnectedElement” references “Iobject”.

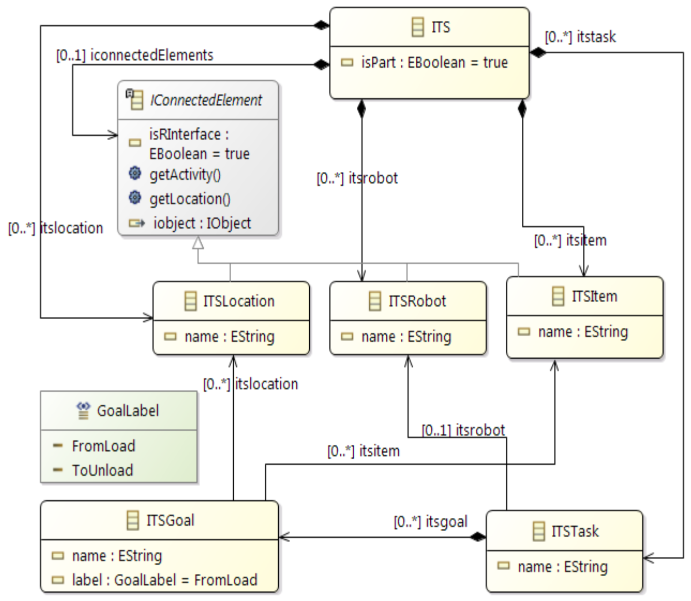


Figure 16: ITS Required Interface

The last step of the coordination process is the coordination between respective Models.

Figure 19 illustrates an ITS Model of a hospital robot transporting medical items from storage to patient rooms. This model has been linked to two IoT models.

The ‘Hospital-Robot’ has the ability to reference an Object due to the previous elaborated coordination. Hence, we associate it to a device Object having two sensors: Speed sensor to get activity and state of the ‘Hospital-Robot and the Battery sensor to monitor battery thresholds of the ’Hospital-Robot.

On the other side, The ‘PatientRoom’ and ‘Storage” have also the ability to be related to an Object element. Thereby, it has been associated to ‘Building’ object.

These associations provide connection to a Location sensor that gives information about Patient rooms and storage localization.

The coordination done at language level is also valuable for all conformed models. Thus, the same CMM could be used to coordinate models belonging to coordinated DSMLs. Consequently, to supervise elements of the second model illustrated in Figure 6 and Figure 7, we use the same CMM. Thereby, in Figure 20 ‘FoodDeliveryRobot’ and ‘CustomerAddress’ have both the possibility to reference an ‘Object’ element of IoT DSML.

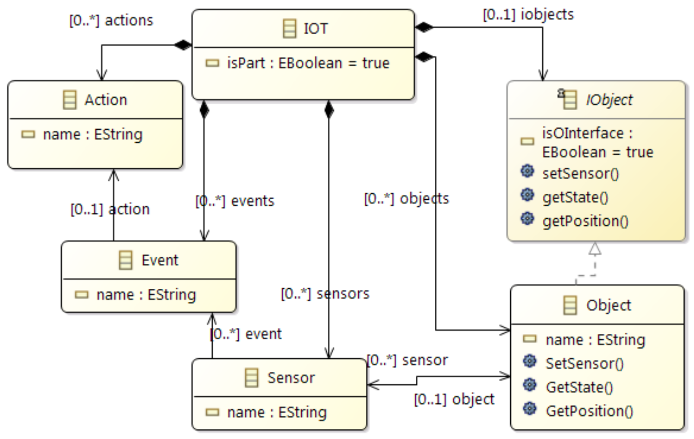


Figure 17: IoT Offered Interface

6. Related Works

Coordination between heterogeneous DSMLs is an active research topic. Although, many research papers discussed this subject, their objectives and motivations are different. Most of them emphasize reutilization and development of DSMLs.

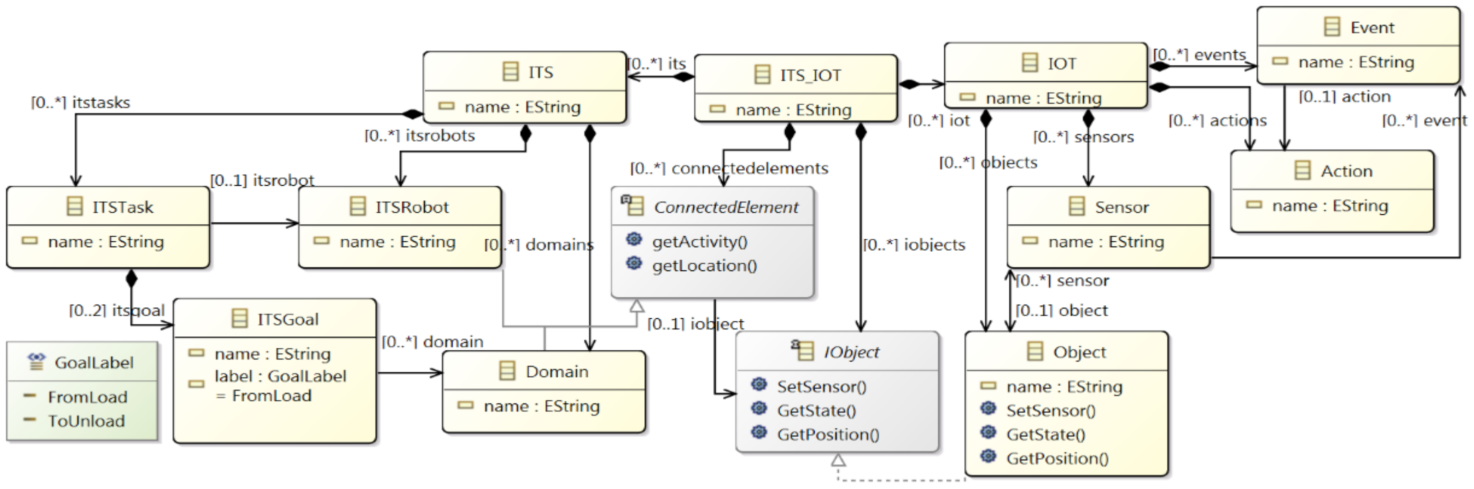


Figure 18: IoT Offered Interface

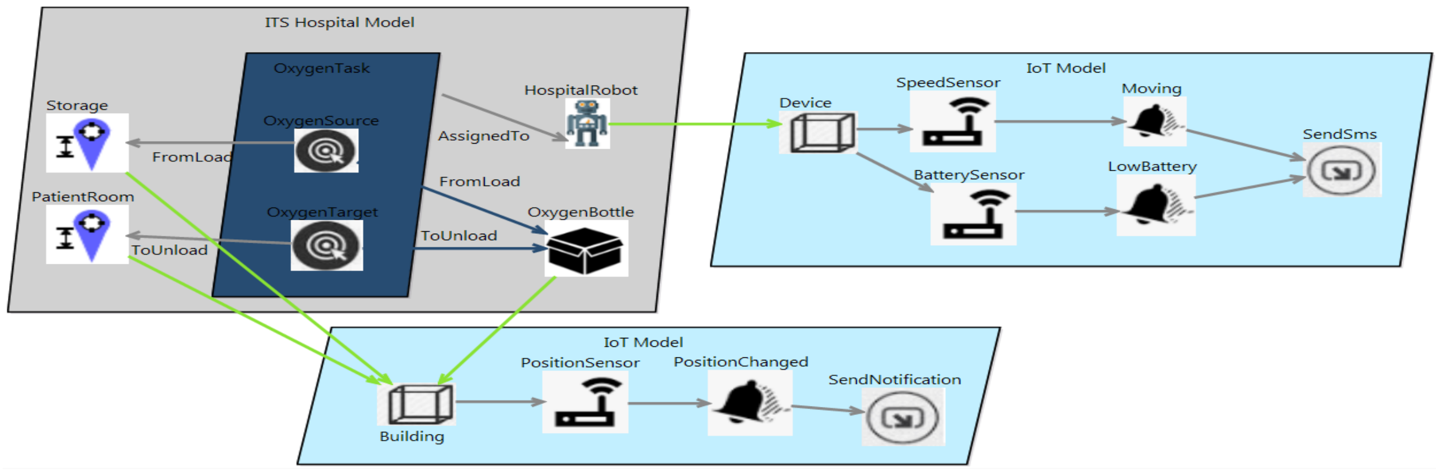


Figure 19: ITS and IoT Models Coordination

This coordination aims to help teams to cooperate with each other in different phases of systems implementation: specification, development, test, maintenance and evolution.

Modularization and interface-based techniques to compose and integrate DSMLs has been proposed by recent papers. A.Klepee [10] discussed the idea of language interfaces to combine languages and like this paper the author gave also a way to define a language interface using attributes and methods. However, our proposition is more explicit. Either, in [8] authors introduced an interface-based approach to define modular meta-models. The approach advises the use of a black-box meta-model definition based on interfaces and corresponding composition operators. This is done using an abstract meta-modeling language enabling the applicability of the approach for various meta-modeling languages. In the same line, our work proposes a composition technique based on offered and required interfaces.

In accordance with this work, Paper [11] advocates the definition of provided and required interfaces for languages as well as their syntactical and semantical composition operators to support DSLs modularity. However, author doesn't talk about how interfaces must be defined. In contrast, in [35] author proposes "Model Types" as example of DSLs interfaces defined on the abstract syntax of languages. Here, models are linked to Model Types using a typing relationship, while Model Types are related to each other with a subtyping relationship.

In [9], author proposes to coordinate heterogeneous behavioral models by specifying coordination patterns between languages. This is done using language behavioral interfaces as well as correspondence and coordination rules. Language behavioral interface in this approach is a set of Domain Specific Events (DSEs) defined in the context of language syntax meta-classes. DSEs are used as coordination points allowing both observation and control of models execution. Actually, coordination is enrolled using BCool the coordination language. That language enables the definition of operators containing matching and coordination rules over DSEs. This proposition is similar to ours as language interface is defined on top of language's abstract syntax even if author didn't give a precise design for interface specification.

In [36] Marco Di Natal et al. use a mapping language to integrate a functional language and a platform language. For that, a set of connectors are used to specify the mapping between the two languages. Once the mapping model is elaborated, communication code between functional model and platform model is generated. We can say that there are some similarities between this proposition and ours. Connectors and mapping model play respectively the same role of interface and coordination model in our proposition.

Finally, in [37] authors define composition interface for model fragments as a set of ports representing reference and variation points of a language. Reference points are root nodes for model fragments, where variation points are nodes that may be replaced during composition. Ports have unique names and map to one or more elements of the fragment. This work is interesting as we ca consider model fragments as separate models that authors coordinate using ports.

7. Conclusion and Future Works

The use of divers DSMLs for modeling a same complex system becomes a common practice. Indeed, systems are more and more complex inciting experts to use their own specific modeling languages.

Albeit, while many language workbenches exist and allow an easy creation of DSMLs, the coordination and integration between them is not proposed or poorly provided.

Subsequently, many recent researches discussed this issue. In the same line, this paper introduces an approach based on both composition and interface concepts to coordinate heterogeneous DSMLs.

Actually, a detailed process composed by two main steps has been proposed. The first step consists in composing involved DSMLs and adds an interface layer to them. This layer is defined according to Bridge Pattern Design.

The proposed approach has been applied to a connected Indoor Transport Service system. The former involves two heterogeneous DSMLs.

For our future works, we plan to explore more relevant cases involving various coordination relationships. Hence, further case studies will help us to improve our coordination process. We also plan to provide a wizard enabling the use of proposed approach and assisting integrators to achieve coordination.

Conflict of Interest

The authors declare no conflict of interest.

Acknowledgment

Acknowledge your institute/ funder.

References

[1] M. Fowler, R. Parsons, Domain Specific Languages, Addison-Wesley Professional, 2010.

[2] C. Hardebolle, " Composition de modèles pour la modélisation multi-paradigme du comportement des systems," Ph.D Thesis, Université Paris-Sud XI Orsay, 2008.

[3] D. Simon-Zayas, "A framework for the management of heterogeneous models in Systems Engineering,", Ph.D Thesis, Ecole Nationale Supérieure de Mécanique et d'Aérotechnique, 2012.

[4] M. El hamlaoui, "Mise en correspondence et gestion de la cohérence de modèles hétérogènes évolutifs," Ph.D Thesis, Université Toulouse-Jean-jaurès, 2015.

[5] M. Chechik, S.Nejati, M.Sabetzadeh, "A Relationship-Based Approch to Model Integration" Innovations in Systems and Software Engineering, 8(1), 3-18, 2012. https://doi.org/10.1007/s11334-011-0155-2

[6] B. Combemale, J. Deantoni, B. Baudry, J. Jézéquel and J. Gray, "Globalizing Modelling Languages" Computer 47.6:68-71, 2014. https://doi.org/ 10.1109/MC.2014.147

[7] J. Deantoni, C. Brun, B. Caillaud, R.B. France, G. Karsai, O. Nierstrasz and E. Syriani, "Domain globalization: using languages to support technical and social coordination", Globalizing Domain-Specific Languages (pp. 70-87). Springer, Cham, 2015. https://doi.org/10.1007/978-3-319-26172-0_5

[8] S. Živković and D. Karagiannis, "Towards metamodelling-in-the-large: Interface-Based composition for modular metamodel development" in Enterprise, Business-Process and Information Systems Modeling. Springer, Cham, 2015. https://doi.org/10.1007/978-3-319-19237-6_26

[9] M.E.V Larsen, J. Deantoni, B. Combemale and F. Mallet, "A Behavioral Coordination Operator Language (BCOol)" in Model Driven Engineering Languages and Systems (MODELS), ACM/IEEE 18th International Conference, 2015. https://doi.org/10.1109/MODELS.2015.7338249

[10] A. Klepee, Software language engineering, Addison-Wesley Professional, 2008.

[11] T. Degueule, "Towards Language Intefaces for DSLs Integration", 2015. https://hal.inria.fr/hal-01138017

[12] D. Strüber, G. Taentzer, S. Jurack & T. Schäfer. "Towards a distributed modeling process based on composite models" in International Conference on Fundamental Approaches to Software Engineering. Springer, Berlin, Heidelberg, 2013. https://doi.org/10.1007/978-3-642-37057-1_2

[13] D. Strüber, S. Jurack,, T. Schäfer, S. Schulz & G. Taentzer. "Managing Model and Meta-Model Components with Export and Import Interfaces." in Big- MDE Workshop on Scalability in Model Driven Engineering. p. 31-36, 2016

[14] ITU-T Recommendation X.208, "Specification of Abstract Syntax Notation One (ASN.1)", 1988.

[15] ITU-T Recommendation X.209, "Specification of Basic Encoding Rules for Abstract Syntax Notation One (ASN.1)", 1988.

[16] M.Bjorklund, "YANG a data modeling language for the network configuration protocol (NETCONF)" (No. RFC 6020), 2010.

[17] M. Bjorklund, "The YANG 1.1 Data Modeling Language (No. RFC 7950)", 2016.

[18] ISO/IEC, "Standard EBNF Syntaxt Specification". Ebnf: Iso/iec 14977, 1996 (e). URL http://www. cl. cam. ac. uk/mgk25/iso-14977. pdf, 1996, vol. 70.

[19] J. Paakki. Attribute grammar paradigms—a high-level methodology in language implementation. ACM Computing Surveys (CSUR), vol. 27, no 2, p. 196-255, 1995.

[20] G. Rozenberg. "Handbook of Graph Grammars and Comp", World scientific, 1997.

[21] L. Fuentes-Fernández & A. Vallecillo-Moreno, "An introduction to UML profiles". UML and Model Engineering, vol. 2, p. 6-13, 2004.

[22] T. Kühne, "Matters of (meta-) modeling". Software & Systems Modeling, vol. 5, no 4, p. 369-385, 2006. https://doi.org/10.1007/s10270-006-0017-9

[23] T. Clark, P. Sammut and J. Willans, Applied metamodeling: a foundation for language driven development, Middlesex University Research Repository, 2008.

[24] K. Chen, J. Sztipanovits and S. Neema, "Toward a semantic anchoring infrastructure for domain-specific modeling languages" in the 5th ACM international conference on Embedded software (pp. 35-43). 2005. https://doi.org/10.1145/1086228.1086236

[25] M. Klein, "Combining and relating ontologies: an analysis of problems and solutions". in Workshop on Ontologies and Information Sharing, IJCAI, 2001

[26] J. P. Tolvanen, R. Pohjonen & S.Kelly. " Advanced tooling for domain-specific modeling: MetaEdit+ " in the 7th OOPSLA Workshop on Domain-Specific Modeling, Finland, 2007. ISBN: 978-951-39-2915-2

[27] R. Heim, P.M.S Nazari, J.O. Ringert, B. Rumpe and A. Wortmann, "Modeling robot and world interfaces for reusable tasks" in IEEE Intelligent Robots and Systems (IROS), IEEE/RSJ International Conference on (pp. 1793-1798), 2015. https://doi.org/10.1109/IROS.2015.7353610

[28] E. Juliot and J. Benois, Viewpoints creation using obeo designer or how to build eclipse dsm without being an expert developer?, Obeo, Tech. Rep., obeo designer whitepaper, 2010. http://www.obeo.fr

[29] Graphical Editing Framework (GEF), http://www.eclipse.org/gef/.

[30] Eclipse Community Forums: GMF (Graphical Modeling Framework): http://www.eclipse.org/forums/eclipse.modeling.gmf.

[31] K. Hölldobler, A. Roth, B. Rumpe and A. Wortmann, "Advances in modeling Language engineering" in Proc of International Conference on Model and Data Engineering (pp. 3-17),Springer,2017. https://doi.org/10.1007/978-3-319-66854-3_1

[32] T. Deguele, "Composition and interoperability for external domain-specific language engineering", Ph.D Thesis, Rennes 1,2016.

[33] T. Clark, M. Van den Brand, B. Combemale and B. Rumpe, "Conceptual model of the globalization for domain-specific languages" in Globalizing Domain-Specific Languages(pp.7-20). Springer, Cham, 2015. https://doi.org/10.1007/978-3-319-26172-0_2

[34] E. Gamma, R. Helm, R. Johnson, J. Vlissides, Design patterns: Elements of Reusable Object-Oriented Software, Pearson Education India ,1995.

[35] J. Steel, J.M Jézéquel, "On model typing" Software & Systems Modeling 6(4), 401–413, 2007. https://doi.org/10.1007/s10270-006-0036-6

[36] M. Di Natale, F. Chirico, A. Sindico and A. Sangiovanni- Vincentelli, "An MDA approach for the generation of communication adapters integrating SW and FW components from Simulink" in International Conference on Model Driven Engineering Languages and Systems (pp. 353-369), Springer, Cham, 2014. https://doi.org/10.1007/978-3-319-11653-2_22

[37] J. Johannes, R. Samlaus and M. Seifert, "Round-trip support for invasive software composition systems" in July 2009 Springer, Berlin, Heidelberg, International Conference on Software Composition (pp. 90-106), 2009. https://doi.org/10.1007/978-3-642-02655-3_8