Journal of Computer Science and Information Technology
December 2014, Vol. 2, No. 3 & 4, pp. 55-81
ISSN: 2334-2366 (Print), 2334-2374 (Online)
Copyright © The Author(s). 2014. All Rights Reserved.
Published by American Research Institute for Policy Development
DOI: 10.15640/jcsit.v2n3-4a4
URL: http://dx.doi.org/10.15640/jcsit.v2n3-4a4

Generating the PIM Behavioral Model from the CIM using QVT

Najiba Addamssiri[1], Abdelouhaed Kriouile[2], Youssef Balouki[3] & Gadi Taoufiq[4]

Abstract

The software process based on the Model Driven Architecture (MDA) is constructed from a set of transformation sequences. In the context of MDA, we have defined an approach based on two kinds of transformation: The first one is the horizontal transformations in the Computation Independent Model (CIM) level between the Business Process Model and Notation and the Use Case Diagram (UC-UML) with her textual description (TD). These transformations provide two entry points into MDA and ensure the refinement of the CIM high level. The Second type is the vertical transformation from CIM to behavioral model of Platform Specific Model (PIM) level represented by the System Sequence Diagram (UML-SSD). We have developed a set of rules using Query/View/Transformation language, and we have automated these steps to automatically generate the UML-SSD diagram from the UC-UML and its textual description structured with Semantics of Business Vocabulary and Business Rules standard which are in turn obtained automatically from the BPMN. Our approach was applied in an e-library books system. The application of our proposal shows that our automatic process can be used to obtain a set of useful artifacts for software development processes. The applicability of the approach is exhibited via one case study.

Keywords: MDA; Model transformation; QVT; CIM; PIM; SBVR

1. Introduction

The term Model-Driven Engineering (MDE) is typically used to describe software development approaches in which abstract models of software systems are created and systematically transformed to concrete implementations (Robert & Bernhard, 2007).

[1] Lavete Laboratory, Univ Hassan 1, 2600 Settat, Maroc. Email: addam.naji@gmail.com
[2] Lavete Laboratory, Univ Hassan 1, 2600 Settat. Email:kriouile1970@gmail.com
[3] Lavete Laboratory, Univ Hassan 1, 2600 Settat. Email:Balouki.youssef@gmail.com
[4] Lavete Laboratory, Univ Hassan 1, 2600 Settat. Email:gtaoufiq@yahoo.fr

The Model Driven Architecture (MDA) (OMG, Model Driven Architecture, ormsc/2001-07-01, July 2001) is a specific variant of MDE that aim at elaborating different models and model transformations which are used to generate implemented level models. In the context of the MDA, model is a viewpoint on a system with regard to the architectural concepts and structuring rules that it tries to abstract.

The model should be conformed to an abstract model named meta-model. As seen in the Figure 1, a transformation model is a process that receives input from the source model which conforms to source meta-model, and then produces an output target model that itself conforms to a target meta-model.

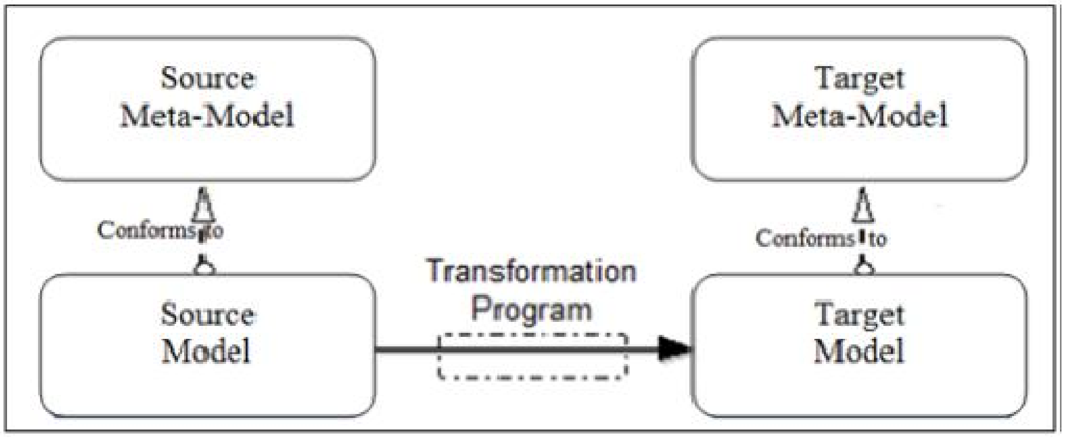


Figure 1: Model Transformation Process

MDA distinguishes among three different models of the process of software development (Miller & Mukerji, 2003). It's initiated with the development of the Computation Independent Model (CIM), and transforms it into the Platform Independent Model (PIM). The PIM is also transformed into the Platform Specific Model (PSM), and at the end, the PSM is used to generate the code of application.

The model transformation approaches proposed in the context of MDA treat, in general, the transformations between PIM, PSM and the Code. However, few researches, which are still not ripened, have enclosed the construction and the modeling of CIM and its transformation to PIM.

Our approach consists of modeling the CIM level and transforming it automatically to the PIM level. In this paper we have focused, firstly on the representation of the CIM level by both the Business Process Model and Notation (BPMN) (OMG, Business Process Model and Notation (BPMN), Version 2.0.1, September 2013) and the Use Cases Diagram (UC-UML) with its Textual Description (TD), and secondly on automatically generating, from the CIM level, the Behavioral PIM Model which is represented by the System Sequence Diagram (UML-SSD) (Larman, 2004).

While any model take part of the transformation should be modeled and conformed to one meta-model, we have proposed to formalize the textual description (TD) of use cases by the Semantics of Business Vocabulary and Business Rules (SBVR) (OMG, Semantics of Business Vocabulary and Business Rules (SBVR), 2013).

At the CIM level, we begin by elaborating the BPMN Model, and then we transform it using the QVT (Query/View/Transformation (OMG, QVT, Meta Object Facility (MOF) 2.0 Query/View/Transformation Specificatio, 2011)) transformation rules to a Use Cases Diagram (UC-UML) with its Textual Description (TD) formalized by the Semantics of Business Vocabulary and Business Rules.

In order to have the same vision between, on one hand the experts of the domain and analysts of the requirements and the other hand, the experts of the system design and development, we propose a refinement of the highest level (CIM) based on a bidirectional transformation between BPMN and UC-UML with its TD-SBVR.

To obtain the Behavioral Model of the PIM from the CIM model, we transform by using a set of QVT transformation rules the Use Cases Diagram with its textual description to a System Sequence Diagram (UML-SSD).

This paper is organized as follow: In section 2, we present a background and related works. Section 3 will cover our proposal model transformations. In section 4, we propose an evaluation based, firstly on one case study and secondly on criteria assessment. Finally, in section 5 we present the conclusions and our future perspectives.

2. Background and Related Works

2.1. CIM and PIM in Short

According to (Streekmann, Steffens, Möbus, & Garbe, 2006), the CIM is the initial point in MDA approach since it includes the business processes used to execute the business of the enterprise, the domain model that represents the intra- or inter-organizational understanding of the domain the application operates in, and the requirements of the system.

The CIM level has a principal role to connect and to facilitate the communication between the domain expert analysts, the business analysts or domain users and the software analysts. This level contains several distinct models that depict system requirements, business processes and business objects (Kriouile, Gadi, & Balouki, CIM to PIM Transformation: A criteria Based Evaluation, July-August 2013). The models represented in the CIM must be understandable by the domain experts and must represent the static, behavioral, and functional aspect of CIM.

The PIM level shows the information system in hiding the details of concrete technology. The models representing this level should describe its static and dynamic aspects. These models must also be productive because they are the foundation of the whole process of code generation defined by the MDA (Kriouile, Gadi, Addamssiri, & El Khadimi, 2014).

2.2. Transformation Language: QVT

In order to implement the various transformations, we have to use a transformation language that takes a model as input, according to the rules, to produce an output model. It is currently possible to find many model transformation languages such as BOTL (Braun & Marschall, 2003), Kermeta1 (Falleri, Huchard, & Nebut, 2006), GReAT (Agrawal, 2003) and ATL (Jouault & Kurtev, 2005). However, the QVT language is the unique proposal from the Object Management Group (OMG).

We picked the QVT language since it supports bidirectional transformations, both horizontal and vertical transformation, solves transformational problems within the OMG/MDA Technical Space, and assures automatic traceability; especially Operational QVT (QVTo (OMG, QVT, Meta Object Facility (MOF) 2.0 Query/View/Transformation Specificatio, 2011)) which has a mature and stable tooling.

2.3. SBVR

SBVR appeared to share the business semantics between the business community and the IT community. It represents an abbreviated of ”Semantics of Business Vocabulary and Business Rules” which is a publicly available specification from the Object Management Group (OMG) (OMG, Semantics of Business Vocabulary and Business Rules (SBVR), 2013) intended to be the basis for a formal and detailed natural language declarative description of business vocabularies and rules.

SBVR allows making business rules accessible to software tools that support the business experts in creating, finding, validating and managing business rules. It also makes these rules accessible to tools that support the information technology experts in converting them into implementation rules for automated systems. SBVR is compatible with MDA and behaves as a Computational Independent Model. This Compatibility with MDA makes it adopted by several business organizations.

The basic principle of SBVR is: “SBVR rules are built on of fact types and facts types are built of terms”.

In our research we use the business vocabulary which has two major types of elements: Concepts and Fact Types.

• A concept is a key term that represents a business entity in a particular domain. The basic types of concepts are (Bajwa, Lee, & Bordbar, 2011): - Noun concept (Term): represented by a word or a group of words represented a business entity.

- Individual concept (Name): represented by a word or a group of words. It represents an instance of a particular term.
- Verb concept: represents the notion of relations and is defined as “a concept that is the meaning of a verb phrase”.

Typically, the common nouns are classified as noun concepts while the proper nouns or quantified nouns are denoted as individual concepts. A verb concept can be an auxiliary verb or action verb or both.

• A fact type is a combination of a verb concept and noun concepts. A Fact type specifies the relationship among different concepts in a business rules.

Every fact can be represented in the form of a term/Name-verb-term/Name template.

2.4. Related Works

According to the evaluation in (Kriouile, Gadi, & Balouki, CIM to PIM Transformation: A criteria Based Evaluation, July-August 2013) which examines the approaches dealing with the modeling and transforming the MDA in high levels CIM and PIM described in the papers (Kherraf, Lefebvre, & Suryn, 2008), (Rodríguez, Fernández-Medina, & Piattini, 2008), (Zhang, Mei, Zhao, & and Yang, 2005), (Kardoš & Drozdová, 2010), (Bousetta, El Beggar, & Gadi, 2013), (Wu, Shin, Chien, Chao, & Hsieh, June 2007), (Fatolahi, Somé, & Lethbridge, 2008), (Sharifi & Mohsenzadeh, 2012), and (Osis, Asnina, & Grave, 2008). It has deduced that the current methods studied were not ripened and did not cover all of the transformation stages.

According to (Kherraf, Lefebvre, & Suryn, 2008) the transformation CIM to PIM is presented as disciplined approach. Business processes and system requirements are modeled in a CIM using two activity diagrams. System requirements are specified from the detailed activity diagrams, and system components are created from the model of requirement elements. Finally, a set of business archetypes helps to transform the system components to the PIM layer in details. This approach is based on modeling the CIM using the UML 2.0 Activity Diagrams as a single technique, and the PIM behavioral aspect is not specified. In (Rodríguez, Fernández-Medina, & Piattini, 2008) it is presented as an approach in which CIM level is represented by business processes in BPMN notation.

It proposes an approach based on the transformation of business process diagrams to analytical UML 2.0 Class Diagrams and UML 2.0 Use Case Diagrams. The CIM is composed of a business process model using the secure business process in BPMN and by UML 2.0 Activity Diagram. The CIM is transformed, with the help of QVT rules, checklists, and refinement rules into two models that are part of the PIM: a Use Case Diagram and a Class Diagram. Use Cases Diagram is moved in this method at the PIM level. In addition, the diagrams of the PIM that are obtained by transformation of the CIM do not cover the PIM behavioral structure. In (Zhang, Mei, Zhao, & and Yang, 2005), the approach is based on features and components which are adopted as the key elements of CIM and PIM building. In this paper, the requirement in CIM is represented by feature model which includes a set of features and relationship between them. The PIM is represented by software architecture that includes a set of components and interaction between them. This method uses an intermediate model that is neither CIM nor PIM. The paper (Kardoš & Drozdová, 2010) represents the CIM level by business processes using the Data Flow Diagram (DFD), and the PIM level by four UML diagrams: Use Cases Diagram, Activity Diagrams, Sequence Diagrams, and Domain Models. While, (Bousetta, El Beggar, & Gadi, 2013) provides a method to build the CIM that can be transformed (semi-) automatically later to lower levels of abstraction in PIMs. The CIM is represented by the BPM and use case model whereas the PIM level is represented using the Sequence Diagram of System's External Behavior (SDSEB) and DCD. This method is based on the business rules to generate the DCD PIM level. In paper (Wu, Shin, Chien, Chao, & Hsieh, June 2007) the CIM is composed of use case diagram, activity diagram and robustness diagram, while the PIM is modeled by two parts: the behavioral part which is presented by using the sequence diagram and the structural part which is depicted using the class diagram.

Other methods that we have found in the literature do not repose on the business processes such as (Fatolahi, Somé, & Lethbridge, 2008), or do not propose how to transform CIM to PIM like in (Sharifi & Mohsenzadeh, 2012) and (Osis, Asnina, & Grave, 2008).

We can conclude that the CIM level doesn't cover, in general, its static, dynamic and behavioral aspects. The traceability doesn't assured in any level. Moreover the proposed approaches can't generate automatically the behavioral aspect of the PIM from the CIM.

3. Our Proposal Approach

In this paper we present an approach that allows, firstly to represent a complete view of a system from the computation independent viewpoint which covers the static, functional and behavioral aspect of the CIM level. This level is represented by the BPMN model, the UC-UML and its textual description formalized by SBVR. And, secondly to represent the behavioral view of the PIM level represented by the System Sequence Diagram (UML-SSD). This approach also assures automatic transformations inside the CIM level and between the CIM level and the PIM level.

Thus, our proposal consists of representing the CIM artifacts that satisfy its static, dynamic and functional views. As shown in figure 2, the construction of CIM level begins by the elaboration of the model of business processes and business objects, using the Business Process Model and Notation (BPMN). Then, by transforming at the same level the BPMN diagram to a Use Cases Diagram (UML-UC) with its Textual Description based on SBVR standard (SBVR-TD). Next, we transform the Use Cases Diagram with its textual description to a System Sequence Diagram (UML-SSD) representing the behavioral aspect of the PIM-level.

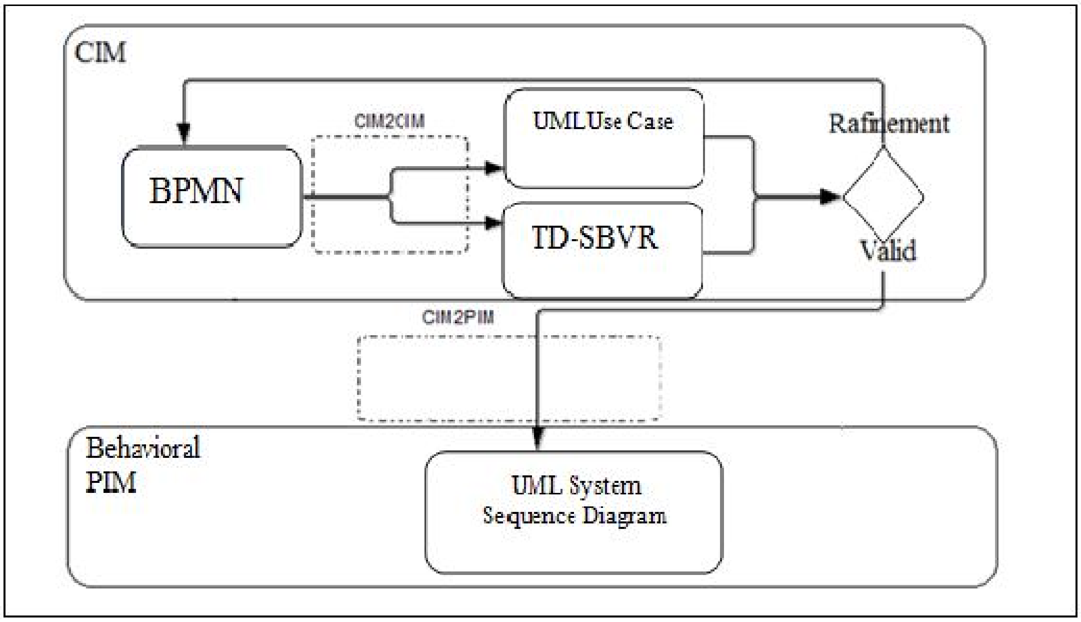


Figure 2. Overview of the Approach

3.1. Modeling of CIM and PIM

According to our previous (Kriouile, Gadi, Addamssiri, & El Khadimi, 2014), we have reached that the static aspect of the best CIM is described by the business objects and the behavioral aspect is described by the business process. Both of them apply the BPMN for their representations. The functional aspect of the CIM is described by the requirement system that is represented by UC Diagram alongside the textual description formalized by the SBVR. Besides the best PIM that covers the behavioral and the static aspect must be represented by the SSD diagram and the DCD diagram. The static aspect will be represented in future work.

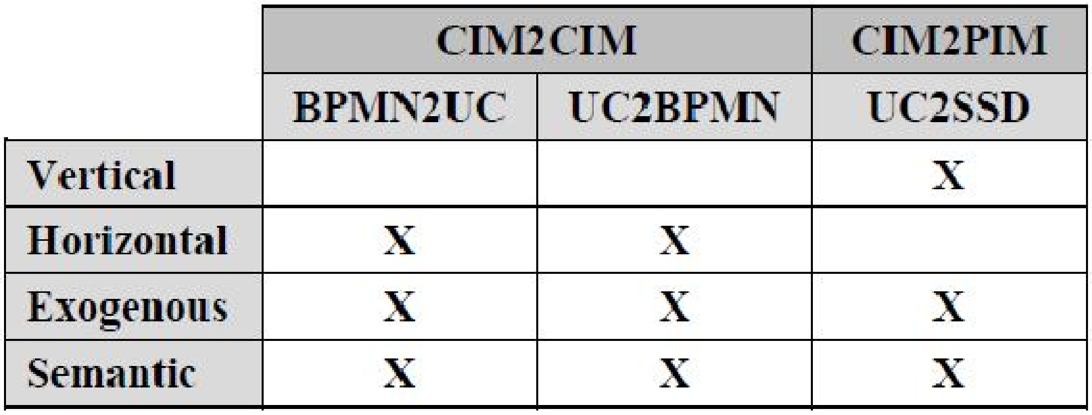
3.2. Models' Transformation Based on QVT

By using the QVT transformation language, we can define transformation rules that map elements of one meta-model to the elements of another metamodel. Once the transformation rules are defined, a transformation process uses these rules and transforms an instance of source meta-model (Model source) into an instance of target meta-model (Model target).

According to the taxonomy in (Yashwant & Manu, 2009) when the source and the target models reside at the same abstraction level the transformation is named horizontal, otherwise the transformation is called vertical. Then once the source and the target models share the same meta-model the transformation titled the Endogenous or else Exogenous. Moreover it's possible to take the semantic of the source model into account in the semantic transformation.

In our approach, at the CIM level it is identified bidirectional horizontal transformations that establish the correspondence between the business process' models defined with the BPMN-BPD and the UC-UML model. We ensure the validity of this level by establishing the refinement process. Then we have defined a vertical transformation which permits to move from CIM using UC-UML model into the behavioral PIM Model represented by the SSD-UML model. The both transformations are exogenous and semantic. Table 1 resumes the different characteristics of all transformations in our approach.

Table 1. Transformation Description



	CIM2CIM		CIM2PIM
	BPMN2UC	UC2BPMN	UC2SSD
Vertical			X
Horizontal	X	X	
Exogenous	X	X	X
Semantic	X	X	X

3.3. CIM to CIM Transformations (CIM2CIM)

In our research, we attempted to model the CIM by diagrams that efficiently represent its different views: static, behavioral and functional. We elaborate a diagram that represents the different business' process (BPMN-BPD) and we transform it to a Use Case Diagram (UML-UC) and its textual description formalized by SBVR (SBVR-TD). Afterwards, we define the transformation from UML-UC and TD-SBVR to UML-UC to refine the level and allow changes of the input model of the approach. The refinement process aimed at enriching, filtering and specializing the CIM level.

3.3.1. From BPMN to Use Case and TD-SBVR

Several main mapping are used to transform the BPMN Model that conformed to the BPMN Meta-model (Figure 3) into the Use Case Model that conforms to the Use Case Meta-model (Figure 4) and its TD-SBVR that conforms to SBVR Meta-model (Figure 5).

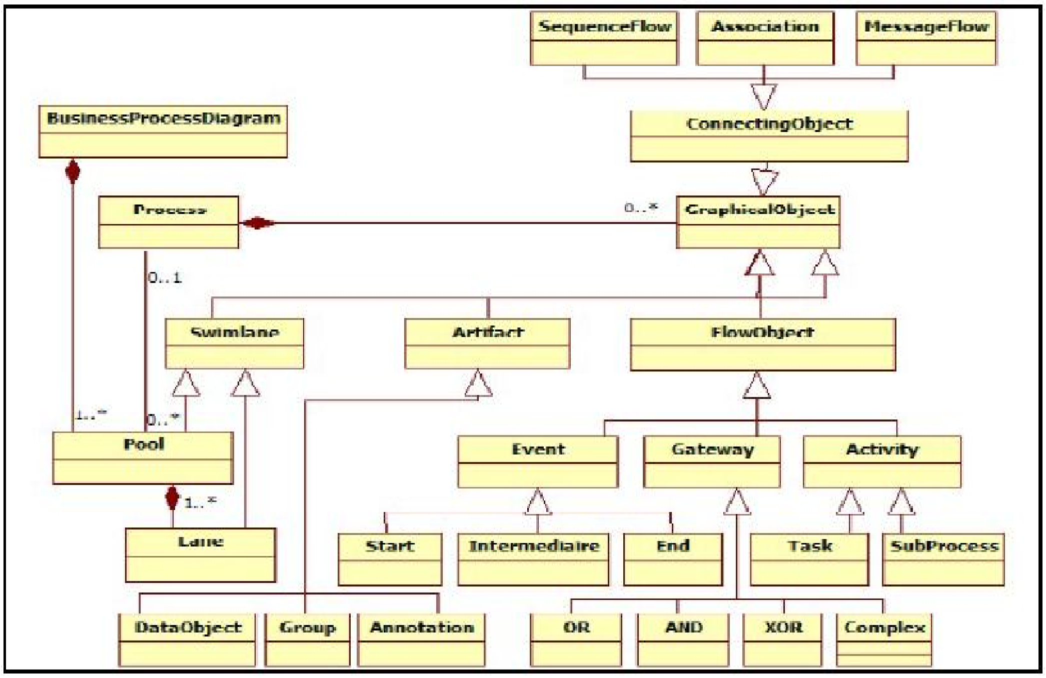


Figure 3: Principal Fragment of the BPMN Meta-Model

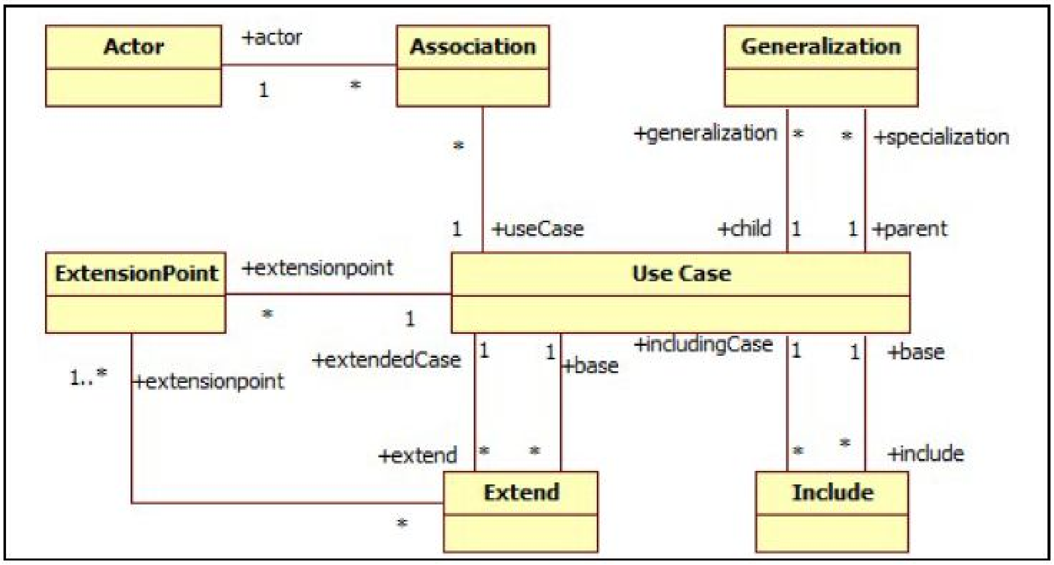


Figure 4: Principal Fragment of the UC Meta-Model

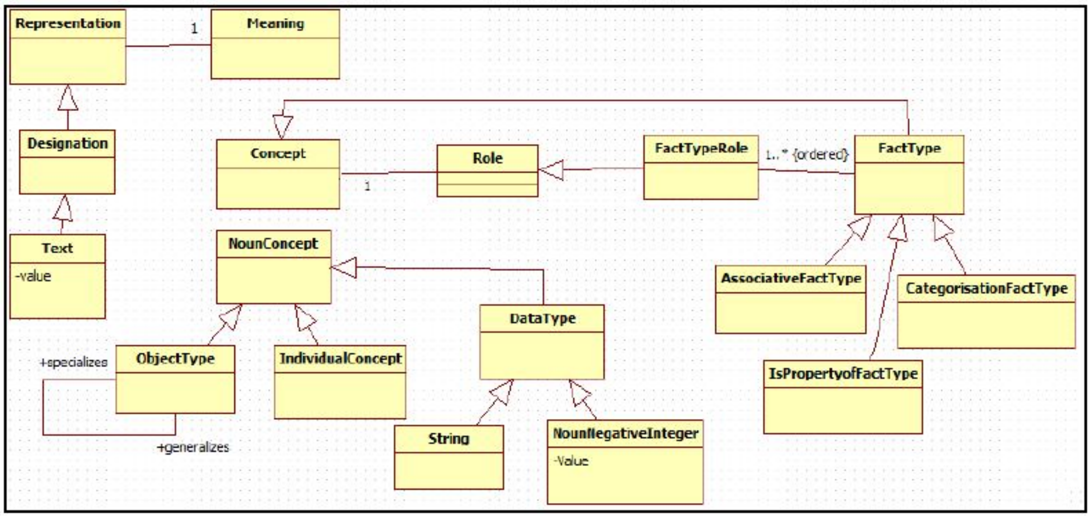


Figure 5. Principal Fragment of the SBVR Meta-Model

As shown in the table 2, we present above the different mappings:

- The source model element “Pool” or “Lane” is transformed into the target model element “Actor” with the same name.
- The “lanes within Pool” are transformed to “generalization relationship”.
- The Pool’s “activities” are mapped to “use cases”.
- The associations are established between actor corresponding to lane and the different Use Cases corresponding to Activities.
- The “SequenceFlow” and “Messageflow” are mapped with “Include” relationship that associate their use cases correspondents.
- The “Gateway” is mapped with “Extend” relationship which associates their use cases correspondents, and defined the “condition”.
- The element “SuccesfulFlow” is mapped with the fact type “SuccessfulScenario”, and then it’s associated with the concept noun “UseCases”, and “Actor”.
- The element “AlternativeFlow” is mapped with the fact type “Alternativescenario”, and then it’s associated with the concept noun “UseCases” and “Actor”.
- The element “ErrorFlow” is mapped to a target model “Error scenario”, and then it’s associated with the concept noun “UseCases” and “Actor”.
- The sub process type “loop” is mapped with the fact type “loop”.

Table 2 also shows in the column “QVT rules” each transformation rule with their code source.

Table 2. BPMN to Use-Case Transformation QVT Rules

Transformation Rule	Source Model Element	Target Model Element	QVT Rules

Pool2Actor	Pool	Actor	mapping Pool::PooltoActor(): Actor
			{ result.name :=self.name;
			result.UseCase += self.Activities.map
			ActivitytoUseCase();
			result.ChildActor += self.lanes.map
			LanetoActor(); }

Lane.within.Pool 2Generalization	Pool that contain Lanes	Actor(Pool) is a generalization of Actor(Lane)	
Lane2Actor	Lane	Actor	mapping Lane::LanetoActor() : Actor

Activity.In.Swimlane 2Association	Activity within a swimlane	Association between actor corresponding to swimlane and UC corresponding to Activity	{ result name := self name;
			result.UseCase += self.Activities.map
			ActivitytoUseCase();
			result.ParentActor :=
			self.Pool.resolveone(Actor); }

Activity2UC	Activity (Sub-Process or Activity of type task)	Use Case	mapping Activity::ActivitytoUseCase() : UseCase
			{ result name:=self name;
			if(self Pool lanes = null) then {
			result.Actor:= self.Pool.resolveone(Actor);}
			endif;
			if(self.Pool.lanes <> null) then {
			result.Actor:= self.Lane.resolveone(Actor}endif;}

Sequence Flow Or Message Flow 2include	Flow	Include	mapping SequenceFlow::SFlowtoInclude() :Include
			when { self.ActivitySource <> null and
			self.ActivityTarget <> null} {
			result.SourceUseCase.–
			self.ActivitySource.resolveone(UseCase);
			result.name:=self.name;
			result.TargetUseCase :=
			self.ActivityTarget.resolveone(UseCase);}
			
			mapping MsgFlow::MsgFlowtoInclude() :Include
			when {self.ActivitySource <> null and
			self.ActivityTarget <> null} {
			result.SourceUseCase:=
			self ActivitySource resolveone(UseCase);
			result.name:=self.name;
			result.TargetUseCase :=
			self.ActivityTarget.resolveone(UseCase);}

Gateway 2extend	Decision Gateway	Extend between the related activities	mapping Gateway::SuccGatewaystoExtend(): Extend {
			result.SourceUseCase:=
			self.InComingSF.ActivitySource.resolveone(UseCase);
			result.TargetUseCase:=
			self.SuccSF.ActivityTarget.resolveone(UseCase);
			result.Condition:=self.name;
			result.name:="Extend relation "+self.name;}

Successful Flow 2 Succ Scenario	Successful Flow	Succesful Scenario	mapping sequenceFlow::toSuccScenario():SuccScenario
			{if (self.Pool.lanes = null) then {
			result.name := self.name;
			result.SourceUseCase :=
			self.ActivitySource.resolveone(UseCase);
			result.TargetUseCase
			:=self.ActivityTarget.resolveone(UseCase);
			result.Actor := self.Pool.resolveone(Actor); }endif;
			if(self Pool lanes <> null) then {
			result name := self name;
			result.SourceUseCase:= self.ActivitySource
			.resolveone(UseCase);
			result.TargetUseCase :=self.ActivityTarget
			.resolveone(UseCase);
			result.Actor := self.LaneSource. resolveone(Actor),
			}endif,}

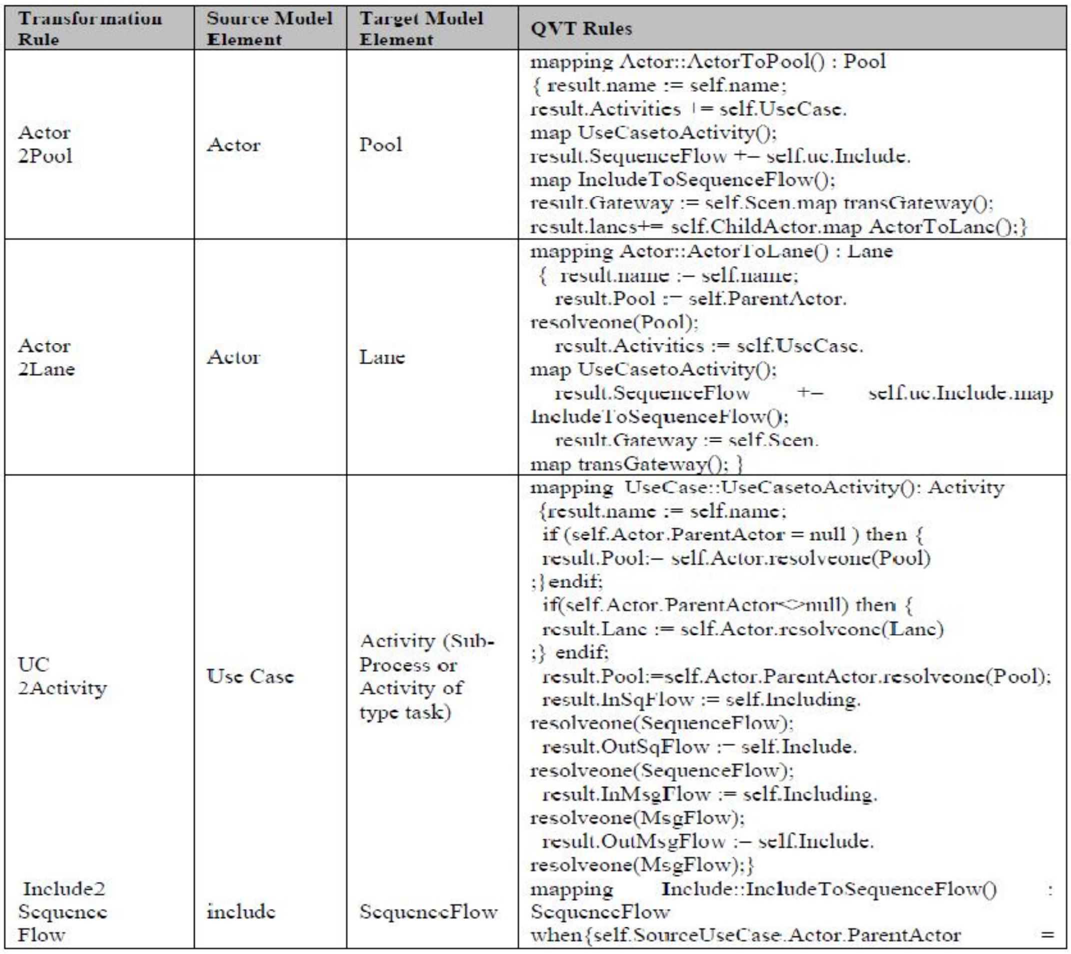
Alternative Flow to Alternative Scenario	Alternative Flow that terminate Sup-Process correctly	Alternative Scenario	mapping SequenceFlow::toAltScenario():AltScenario { if (self.Pool.lanes = null) then { result.name := self.name; result.SourceUseCase := self.ActivitySource.resolveone(UseCase); result.TargetUseCase :=self.ActivityTarget.resolveone(UseCase); result.Actor := self.Pool.resolveone(Actor); }endif; if(self.Pool.lanes <> null) then { result.name := self.name; result.SourceUseCase:= self.ActivitySource .resolveone(UseCase); result.TargetUseCase :=self.ActivityTarget .resolveone(UseCase); result.Actor := self.LaneSource. resolveone(Actor);} endif;}
ErrorFlow 2Error Scenario	Error Flow that terminate Sup-Process with errors	Error Scenario	mapping SequenceFlow::toErrScenario():ErrScenario { if (self.Pool.lanes = null) then { result.name := self.name; result.SourceUseCase := self.ActivitySource.resolveone(UseCase); result.TargetUseCase :=self.ActivityTarget.resolveone(UseCase); result.Actor := self.Pool. resolveone(Actor);}endif; if(self.Pool.lanes <> null) then { result.name := self.name; result.SourceUseCase:= self.ActivitySource .resolveone(UseCase); result.TargetUseCase :=self.ActivityTarget .resolveone(UseCase); result.Actor := self.LaneSource. resolveone(Actor); } endif;}
SubProcessLoop2 FactTypeLoop	SubProcess Loop	FactType Loop	mapping SubproLoop::toFactLoop():FLoop { if (self.Pool.lanes = null) then { result.Include += self.SequenceFlow.map SFlowtoInclude(); result.Include += self.MsgFlow.map MsgFlowtoInclude(); result.Scen += self.Getway.map ToScen(); result.Actor:= self.Pool.resolveone(Actor);}endif; if(self.Pool.lanes <> null) then { result.Include += self.SequenceFlow.map SFlowtoInclude(); result.Include += self.MsgFlow.map MsgFlowtoInclude(); result.Scen += self.Getway.map ToScen(); result.Actor := self.LaneSource.resolveone(Actor);}endif; }

3.3.2. From Use Case to BPMN

We have transformed Use Case Diagram that is conformed to the Use Cases Meta-model and its TD-SBVR that is conformed to SBVR Meta-model into BPMN Diagram which is conformed to BPMN Meta-model.

As shown in the table 3, the different elements of UC-UML and its TD-SBVR are transformed to BPMN model. Thus, the source model element “Actor” is transformed into the target model element “Pool” with the same name, the “Secondary Actor” is transformed into a “Lane” with the same name, the “generalization” relation is transformed to “Lane within Pool”, all Actor’s “Use Cases” are transformed into “Activities”, the “include relation” is transformed into a “SequenceFlows”, the “extend relation” is transformed into “gateway”, the fact type “Successful Scenario” is mapped with “SuccesfulFlow”, “AlternativeScenario” is mapped with “AlternativeFlow”, and “Error Scenario” is mapped with “ErrorFlow”.

Table 3. Use-Case to BPMN Transformation QVT Rule



Transformation Rule	Source Model Element	Target Model Element	QVT Rules

Actor 2Pool	Actor	Pool	mapping Actor::ActorToPool() : Pool
			{ result.name := self.name;
			result.Activities != self.UseCase.
			map UseCasetoActivity();
			result.SequenceFlow +– self.uc.Include.
			map IncludeToSequenceFlow();
			result.Gateway := self.Scen.map transGateway();
			result.lanes+= self.ChildActor.map ActorToLane();}

Actor 2Lane	Actor	Lane	mapping Actor::ActorToLane() : Lane
			{ result.name :– self.name;
			result.Pool :– self.ParentActor.
			resolveone(Pool);
			result.Activities := self.UseCase.
			map UseCasetoActivity();
			result.SequenceFlow +– self.uc.Include.map
			IncludeToSequenceFlow();
			result.Gateway := self.Scen.
			map transGateway(); }

UC 2Activity	Use Case	Activity (Sub-Process or Activity of type task)	mapping UseCase::UseCasetoActivity(): Activity
			{result.name := self.name;
			if (self.Actor.ParentActor = null) then {
			result.Pool:– self.Actor.resolveone(Pool)
			;}endif;
			if(self.Actor.ParentActor<>null) then {
			result.Lane := self.Actor.resolveone(Lane)
			;} endif;
			result.Pool:=self.Actor.ParentActor.resolveone(Pool);
			result.InSqFlow := self.Including.
			resolveone(SequenceFlow);
			result.OutSqFlow :– self.Include.
			resolveone(SequenceFlow);
			result.InMsgFlow := self.Including.
			resolveone(MsgFlow);
			result.OutMsgFlow :– self.Include.
			resolveone(MsgFlow);}

Include2 Sequence Flow	include	SequenceFlow	mapping Include::IncludeToSequenceFlow() : SequenceFlow when{self.SourceUseCase.Actor.ParentActor =

			self.TargetUseCase.Actor.ParentActor or self.SourceUseCase.Actor.ParentActor = null and self.SourceUseCase.Actor.ParentActor = null }{result.name := self.name; result.ActivitySource := self.SourceUseCase. resolveone(Activity); result.ActivityTarget := self.TargetUseCase. resolveone(Activity); result.Pool:= self.SourceUseCase.Actor.ParentActor. resolveone(Pool); result.LaneSource := self.SourceUseCase.Actor. resolveone(Lane); result.LaneTarget := self.TargetUseCase.Actor. resolveone(Lane);}
Include2 Message Flow	Include	MessageFlow	mapping Include::IncludeToMsgFlow(): MsgFlow when{self.SourceUseCase.Actor.ParentActor <> self.TargetUseCase.Actor.ParentActor}{ result.name := self.name; result.ActivitySource := self.SourceUseCase. resolveone(Activity); result.ActivityTarget := self.TargetUseCase. resolveone(Activity);}
Scenario2 Gateway	Scenario	Gateway	mapping Scen::transGateway(): Gateway { result.name:= self.name; result.InComingSF := self.SuccScenario. SourceUseCase.Include. resolveone(SequenceFlow); result.Pool:=self.SuccScenario.Actor. ParentActor.resolveone(Pool); result.Lane := self.SuccScenario. Actor.resolveone(Lane); result.SuccSF :=self.SuccScenario. map toSqFlow(); result.AltSF := self.AltScenario. map toSqFlow(); result.ErrSF := self.ErrScenario. map toSqFlow();}

3.4. CIM to PIM Transformation

This transformation is used to transform Use Cases Model that is conformed to Use Cases Meta-model and TD-SBVR that's conformed to SBVR Meta-model existing in the CIM level into System Sequence Model that's conformed to System Sequence Meta-model (Figure 6) existing in the PIM level. In which the system is considered as a whole.

The QVT transformation rules developed to obtain from the source model elements the target elements of SSD are classified in Table 4. A “Principal Actor” is transformed into “Actor”, and the fact types in the TD-SBVR are transformed to interactions between Actor and System: the fact type “Alt-Scenario” is mapped with interaction fragment “Alt”, the “ErrScenario” is mapped with the interaction fragment “Break” and the “SuccScenario” to “message from Actor to System”, or to “System response to Actor”, and taking into account the “internal Message.

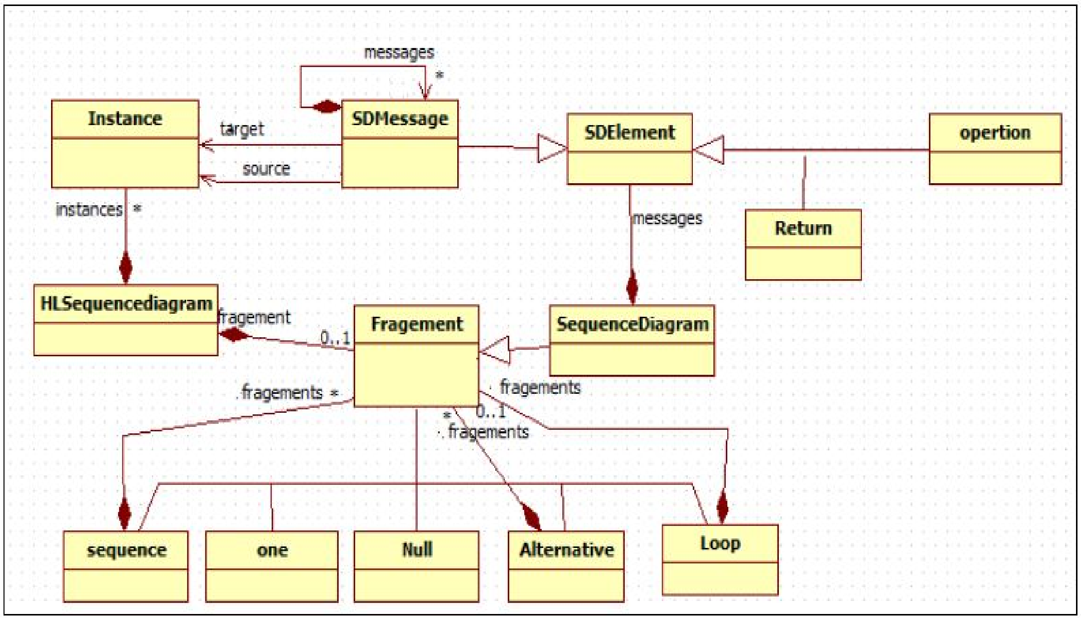


Figure 6: Principal Fragment of the SSD Meta-Model

Table 4. Use-Case to SSD Transformation QVT Rules

Transformation Rule	Source Model Element	Target Model Element	QVT Rules
UseCase2SSD	Use Case	SSD	mapping uc::UCtoDSS(): DSS{ log("Mapping UC to DSS diagram") result.Actor += self.Actors.map toActor(); result.Sequences += self.Actors.Scen.map toSequence();}
PrincipalActor 2Actor	Actor that directly operates on the System	Actor	mapping Actor::toActor(): ACTOR {result.name:= self.name;}
SystemEvent 2SystemMessage OR System Response 2SystemMessage	SystemEvent (Message sent to System) OR SystemRespon se (Response Message from the System to Actor)	Message sent From Actor to System OR Message sent From System to Actor	mapping SuccScenario :: tosucc(): Succ {result.name := self.name; result.Messages+= self.SourceUseCase.map toMsg(); result.Messages+= self.TargetUseCase .map toMsg();}
Fact Type2Alt	FactType	Interaction Fragment 'Alt'	mapping Scen::toAlt():Alt {result.Break:= self.ErrScenario.map toBreak(); result.Opt := self.AltScenario.map toOpt(); result.Succ := self.SuccScenario.map tosucc();result.Loop := self.Loop.map toLoop();}

4. Evaluation

4.1. Case Study

In this section we present an example to illustrate our approach. We consider the case of the business process of e-library books. This example models the interaction between customers and the system. Any surfer on web can access to the web site and search one book, they can read it online or download it. Also, they can request a new book by filling a form. Web surfer must connect with their account or subscribe if it's their first visit of the web site.

To implement the proposed approach for the chosen case study, we start with the lower level sub-process business model. In Figure 7, we present the detailed "Choose Book Sub-Process" organized in workflow and represented using the BPMN notation.

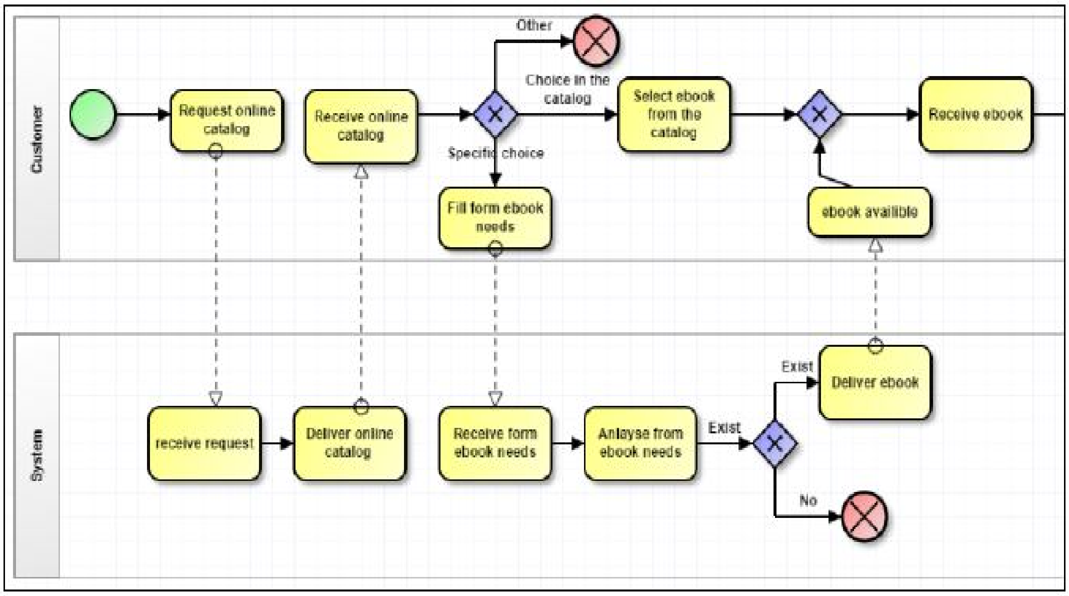


Figure 7: BPD "Choose Book" Sub-Process of the Case Study "E-Library Books"

4.1.1. BPMN to UC-UML and TD-SBVR

To obtain use cases diagram and its TD-SBVR, we use the transformation rules stipulated at Section 3.3.1. The application of these transformation rules allows to identify one actor "Customer" and then use cases: "Request online catalog", "Receive request", "Receive online catalog", "Deliver online catalog", "Select eBook from the catalog", "Fill form eBook needs", "Receive form eBook needs", "Analyze from eBook needs", "Deliver eBook" and "Receive eBook". Figure 9 illustrates the use cases model and the figure 10 depicts the textual description model formalized on SBVR (TD-SBVR model) of the case study.

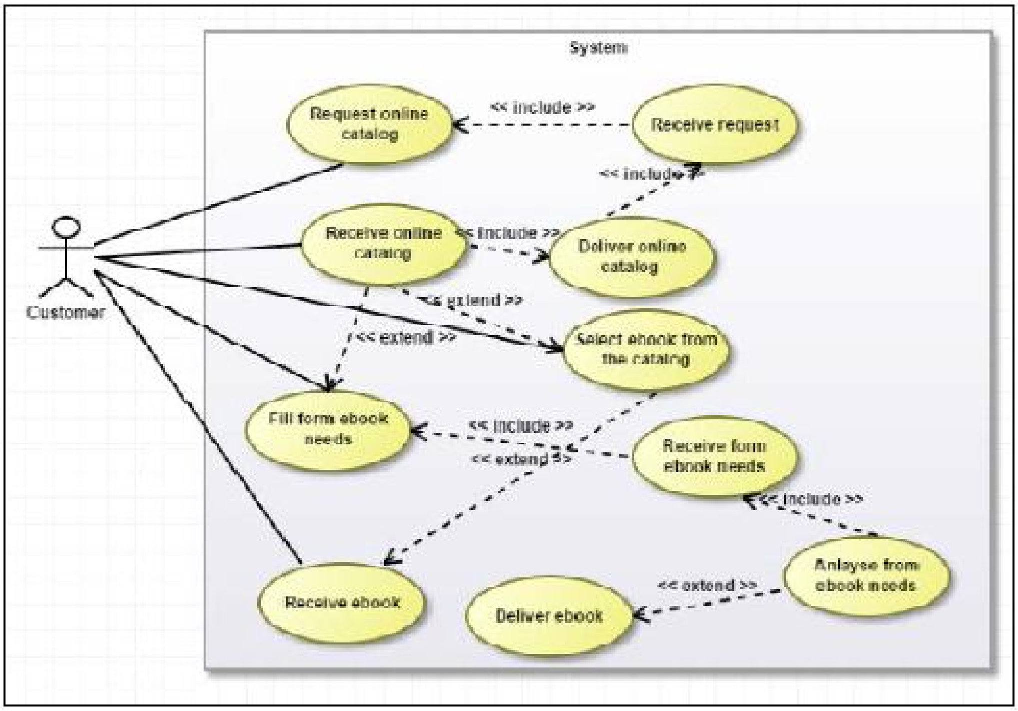


Figure 8. Use Cases Diagram of the Case Study

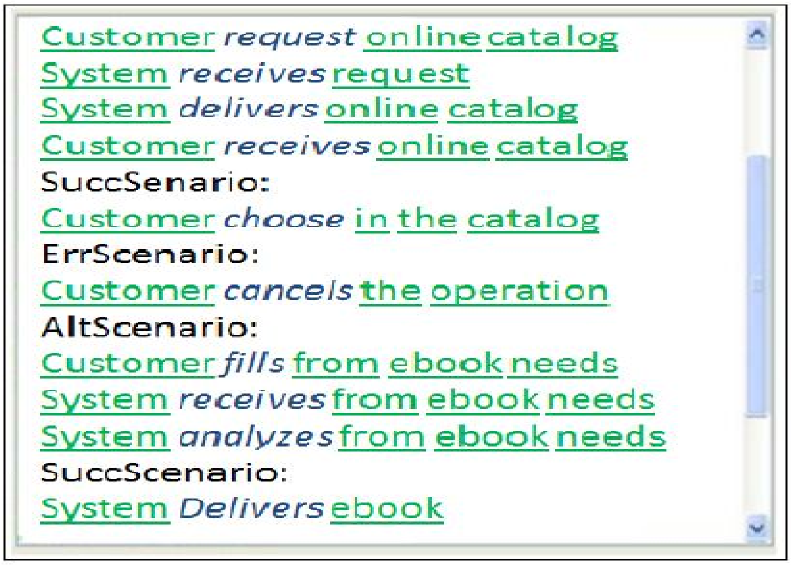

Customer request online catalog
System receives request
System delivers online catalog
Customer receives online catalog
SuccSenario:
Customer choose in the catalog
ErrScenario:
Customer cancels the operation
AltScenario:
Customer fills from ebook needs
System receives from ebook needs
System analyzes from ebook needs
SuccScenario:
System Delivers ebook

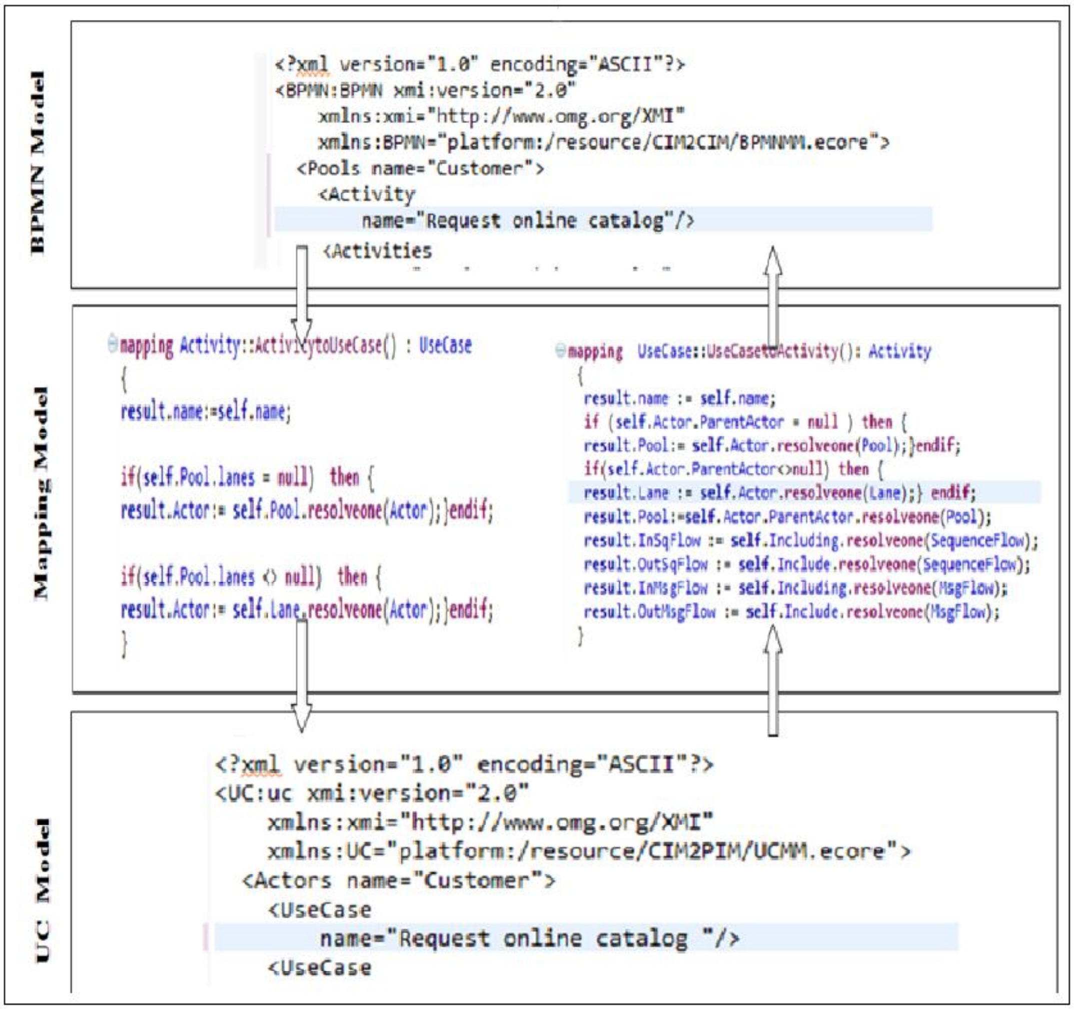
Figure 9: Extract from Textual Description of the Case Study Formalized on SBVR

4.1.2. UC-UML and TD-SBVR to BPMN

The mapping rules proposed at the section 3.3.2. allow to generate BPMN Model from UC-UML Model and its TD-SBVR. Thus, we can identify two Pools: "Customer" and "System", and the activities: "Request online catalog", "Receive request", "Receive online catalog", "Deliver online catalog", "Select eBook from the catalog', 'Fill form eBook needs', 'Receive form eBook needs', 'Analyze from eBook needs', 'Deliver eBook' and "Receive eBook". The successful flow "Choice in the catalog" of the gateway generated from the success scenario "Choice in the catalog" of the TD-SBVR, the Error Flow "Other" correspond to Error Scenario "Other'" and the Alt Flow "Specific choice" correspond to Alt Scenario "Specific choice".

4.1.3. CIM to CIM QVT Code

Figure 10 shows one example of the QVT code applied for our example. It illustrates the mapping between the Activity “Request online catalog” in the Pool “Customer” and the use case “Request online catalog” associated to the Actor “Customer” in both directions.



```
<?xml version="1.0" encoding="ASCII"?>
<BPMN:BPMN xmi:version="2.0"
    xmlns:xmi="http://www.omg.org/XMI"
    xmlns:BPMN="platform:/resource/CIM2CIM/BPMNMM.ecore">
  <Pools name="Customer">
    <Activity
        name="Request online catalog"/>
    <Activities
```

```
mapping Activity::ActivitytoUseCase() : UseCase
{
result.name:=self.name;

if(self.Pool.lanes = null)  then {
result.Actor:= self.Pool.resolveone(Actor);}endif;

if(self.Pool.lanes <> null)  then {
result.Actor:= self.Lane.resolveone(Actor);}endif;
}
```

```
mapping  UseCase::UseCasetoActivity(): Activity
{
 result.name := self.name;
 if (self.Actor.ParentActor = null ) then {
 result.Pool:= self.Actor.resolveone(Pool);}endif;
 if(self.Actor.ParentActor<>null) then {
 result.Lane := self.Actor.resolveone(Lane);} endif;
 result.Pool:=self.Actor.ParentActor.resolveone(Pool);
 result.InSqFlow := self.Including.resolveone(SequenceFlow);
 result.OutSqFlow := self.Include.resolveone(SequenceFlow);
 result.InMsgFlow := self.Including.resolveone(MsgFlow);
 result.OutMsgFlow := self.Include.resolveone(MsgFlow);
}
```

```
<?xml version="1.0" encoding="ASCII"?>
<UC:uc xmi:version="2.0"
    xmlns:xmi="http://www.omg.org/XMI"
    xmlns:UC="platform:/resource/CIM2PIM/UCMM.ecore">
  <Actors name="Customer">
    <UseCase
        name="Request online catalog "/>
    <UseCase
```

Figure 10. CIM2CIM QVT Code

4.1.4. Generating the SSD-UML from UC-UML and TD-SBVR

From the Use Case Model, by applying the mapping rules proposed in table 4 at the section 3.4., we can identify the principal actor that is the "Customer". And from the TD-SBVR we can identify the "action/response" from/to System: Loop, Opt, and Break. The Opt "Form eBook needs' correspond to Alt Successful "Form eBook needs" and the Break 'Choose Cancel' corresponds to Error Scenario "Choose cancel".

The generated SSD of the use case “choose eBook” is presented in figure 11.

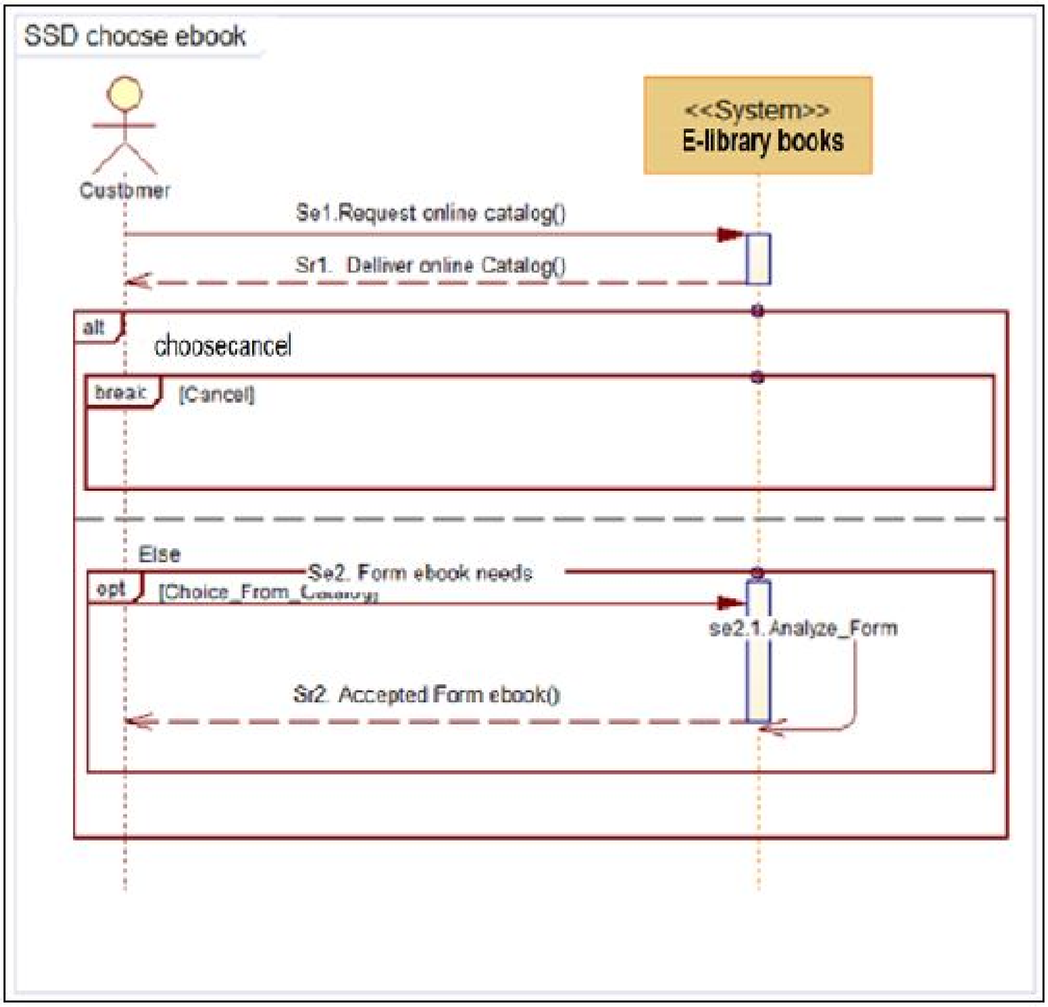
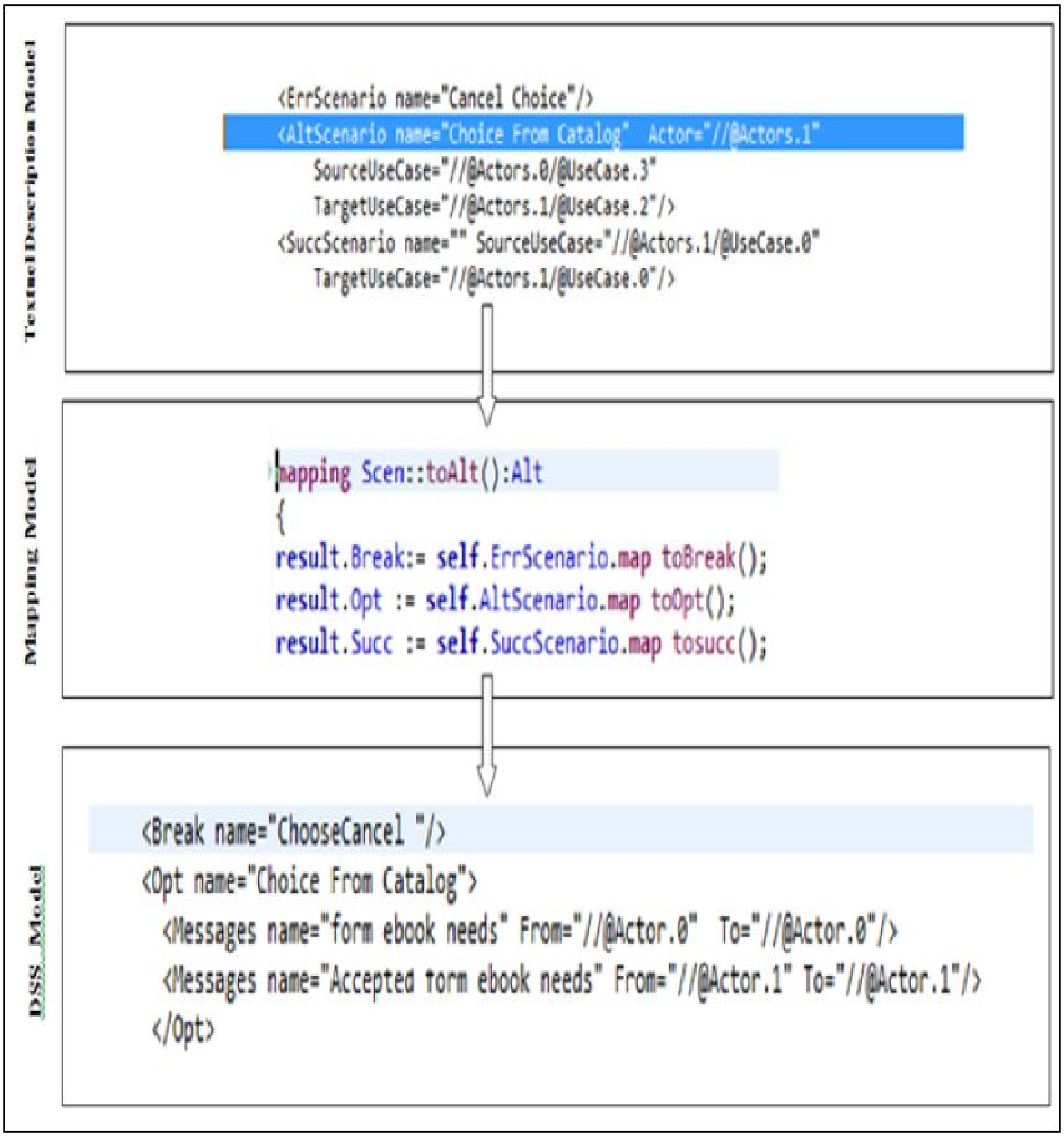


Figure 11: The SSD of the Choose Ebook Use Case

4.1.5. CIM to PIM QVT Code

We illustrate in figure 12, the mapping from the Alt Scenario “Choice from catalog” to “Opt” and the “Cancel choice” to “Break”.



```
<ErrScenario name="Cancel Choice"/>
<AltScenario name="Choice From Catalog"  Actor="//@Actors.1"
   SourceUseCase="//@Actors.0/@UseCase.3"
   TargetUseCase="//@Actors.1/@UseCase.2"/>
<SuccScenario name="" SourceUseCase="//@Actors.1/@UseCase.0"
   TargetUseCase="//@Actors.1/@UseCase.0"/>
```

```
mapping Scen::toAlt():Alt
{
result.Break:= self.ErrScenario.map toBreak();
result.Opt := self.AltScenario.map toOpt();
result.Succ := self.SuccScenario.map tosucc();
```

```
<Break name="ChooseCancel "/>
<Opt name="Choice From Catalog">
  <Messages name="form ebook needs" From="//@Actor.0"  To="//@Actor.0"/>
  <Messages name="Accepted form ebook needs" From="//@Actor.1" To="//@Actor.1"/>
 </Opt>
```

Figure 12: CIM2PIM Transformation

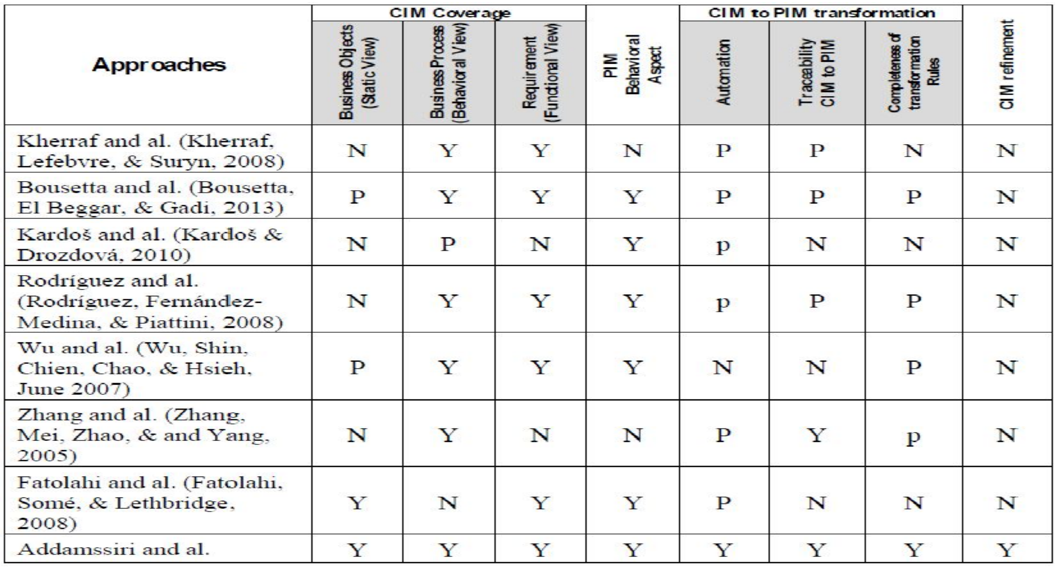
4.2. Criterion Evaluation

After evaluating our approach with a case study, we propose to evaluate it according to four evaluation criteria: "CIM Coverage", "PIM Behavioral aspect", "CIM to PIM transformation" and "CIM refinement".

In table 5, we present the evaluation of seven approaches with our proposal based on the criteria mentioned before. The comparison of those seven approaches was extracted from our previous work (Kriouile, Gadi, & Balouki, CIM to PIM Transformation: A criteria Based Evaluation, July-August 2013). Regarding to “CIM coverage”, no method of those seven methods fully covers the three aspects of the CIM: Static, Behavioral and Functional view of CIM. While regarding to "PIM Behavioral Aspect", five out of seven methods can generate behavioral model of the PIM. However, concerning the "CIM to PIM transformation" criterion, we can see that the most of the methods do not even provide guidelines to ensure traceability between CIM and PIM, the definitions of the methods are not complete, and the transformation still require a human intervention. Finally, no method of those seven methods ensures the CIM refinement.

Consequently, according to the criteria mentioned above we can consider our approach as complete. It covers the static, the dynamic and the behavioral view of the CIM level, as well as assures their refinement and can generate automatically the behavioral aspect of the PIM level.

Table 5. Criteria Based Evaluation



Approaches	CIM Coverage			PIM Behavioral Aspect	CIM to PIM transformation			CIM refinement
	Business Objects (Static View)	Business Process (Behavioral View)	Requirement (Functional View)		Automation	Traceability CIM to PIM	Completeness of transformation Rules	
Kherraf and al. (Kherraf, Lefebvre, & Suryn, 2008)	N	Y	Y	N	P	P	N	N
Bousetta and al. (Bousetta, El Beggar, & Gadi, 2013)	P	Y	Y	Y	P	P	P	N
Kardoš and al. (Kardoš & Drozdová, 2010)	N	P	N	Y	p	N	N	N
Rodríguez and al. (Rodríguez, Fernández-Medina, & Piattini, 2008)	N	Y	Y	Y	p	P	P	N
Wu and al. (Wu, Shin, Chien, Chao, & Hsieh, June 2007)	P	Y	Y	Y	N	N	P	N
Zhang and al. (Zhang, Mei, Zhao, & and Yang, 2005)	N	Y	N	N	P	Y	p	N
Fatolahi and al. (Fatolahi, Somé, & Lethbridge, 2008)	Y	N	Y	Y	P	N	N	N
Addamssiri and al.	Y	Y	Y	Y	Y	Y	Y	Y

Legend: Y: Yes; N: No; P: Partial

5. Conclusions and Future Perspectives

Being aware of the importance of the MDA high parts in bridging the gap between the business experts and the system analyses experts, we have so far presented an approach defining, firstly, the artifacts of the CIM. They cover its static, behavioral and functional aspects based on BPMN-BPD, UC-UML and TD-SBVR models. Secondly the approach also ensures the validation and the refinement of the CIM level by establishing a bidirectional transformation between UC-UML and BPMN.

Moreover, this approach specifies the behavior PIM Model by SSD-UML Model which is automatically generated using the QVT transformation language from UC-UML Model and its TD-SBVR.

The method was evaluated trough using both a concrete case study concerning an e-library books system and criteria based evaluation.

The proposed method in this paper completes our previous works (Kriouile, Gadi, Addamssiri, & El Khadimi, 2014) and (Kriouile, Addamssiri, Gadi, & Balouki, 2014) subscribing in global method which aims at automating the whole CIM to PIM transformation. For future perspective, we intend to develop a graphical tool that allows designing the artifacts and successively running transformations defined in the present approach.

References

Agrawal, A. (2003). GReAT: a metamodel based model transformation language. 18th IEEE International Conference on Automated Software Engineering.

Braun, P., & Marschall, F. T. (2003). The Bidirectional Object Oriented Transformation. RN.

Bajwa, I. S., Lee, M. G., & Bordbar, B. (2011). SBVR Business Rules Generation from Natural Language Specification. AAAI Spring Symposium: AI for Business Agility, 2-8.

Bousetta, B., El Beggar, O., & Gadi, T. (2013). A methodology for CIM modelling and its transformation to PIM. Journal of Information Engineering and Applications, 3(2), 1-21.

Falleri, J., Huchard, M., & Nebut, C. (2006). Towards a traceability framework for model transformations in kermeta. European Conference on Model-Driven Architecture Traceability Workshop (ECMDA-TW), Bilbao,Spain.

Fatolahi, A., Somé, S. S., & Lethbridge, T. C. (2008). Towards a semi-automated model-driven method for the generation of web-based applications from use cases. 4th Model Driven Web Engineering Workshop (p. 31).

Jouault, F., & Kurtev, I. (2005). Transforming models with ATL. International Workshop on Model Transformations in Practice (MTiP).

Kherraf, S., Lefebvre, E., & Suryn, W. (2008). Transformation from CIM to PIM Using Patterns and Archetypes. 19th Australian Conference on Software Engineering, (pp. pages 338–346).

Kardoš, M., & Drozdová, M. (2010). Analytical method of CIM to PIM transformation in Model Driven Architecture (MDA). JOURNAL OF INFORMATION AND ORGANIZATIONAL SCIENCES, 34, 89-99.

Kriouile, A., Gadi, T., & Balouki, Y. (July-August 2013). CIM to PIM Transformation: A criteria Based Evaluation. IJCTA, Vol 4 (4)(ISSSN:2229-6093), 616-625.

Kriouile, A., Gadi, T., Addamssiri, N., & El Khadimi, A. (2014). Obtaining Behavioral Model of PIM from the CIM. Multimedia Computing and Systems (ICMCS), 2014 International Conference. IEEE, 949 - 954.

Kriouile, A., Addamssiri, N., Gadi, T., & Balouki, Y. (2014). Getting the Static Model of PIM from the CIM. 3rd Colloquium IEEE on Information Science and Technology (CiSt'14). Tetuan.

Larman, C. (2004). Applying UML and Patterns : An Introduction to Object-Oriented Analysis and Design and the Unified Process. Addison-Wesley Professional.

Miller, J., & Mukerji, J. (2003). MDA Guide Version 1.0.1. OMG.

OMG. (July 2001). Model Driven Architecture, ormsc/2001-07-01.

Osis, J., Asnina, E., & Grave, A. (2008). Computation independent representation of the problem domain. MDA. J. Software Eng, 2(1), 19-46.

OMG. (2011, junuary 1). QVT, Meta Object Facility (MOF) 2.0 Query / View/ Transformation Specificatio. Récupéré sur
http://www.omg.org/spec/QVT/1.1/

OMG. (September 2013). Business Process Model and Notation (BPMN), Version 2.0.1. OMG, http://www.omg.org/spec/BPMN.

OMG. (2013, November). Semantics of Business Vocabulary and Business Rules (SBVR). Récupéré sur http://www.omg.org/spec/SBVR/1.2/PDF

Robert, F., & Bernhard, R. (2007). Model-driven Development of Complex Software: A Research Roadmap. ICSE .

Rodríguez, A., Fernández-Medina, E., & Piattini, M. (2008). Towards obtaining analysis-level class and use case diagrams from business process models. Advances in Conceptual Modeling–Challenges and Opportunities. Springer Berlin Heidelberg., 103-112.

Streekmann, N., Steffens, U., Möbus, C., & Garbe, H. (2006). Model-driven integration of business information systems. Softwaretechnik-Trends, 26(4).

Sharifi, H. R., & Mohsenzadeh, M. (2012). A New Method for Generating CIM Using Business and Requirement Models. World of Computer Science and Information Technology Journal (WCSIT), 2(1), 8-12.

Wu, J. H., Shin, S. S., Chien, J. L., Chao, W. S., & Hsieh, M. C. (June 2007). An extended MDA method for user interface modeling and transformation. The 15th European Conference on Information Systems (pp. 1632-1641).

Yashwant, S., & Manu, S. (2009). Models and Transformations in MDA. Computational Intelligence, Communication Systems and Networks, 253-258.

Zhang, W., Mei, H., Zhao, H., & and Yang, J. (2005). Transformation from CIM to PIM: A Feature-Oriented Component-Based Approach. Model Driven Engineering Languages and Systems volume 3713 of Lecture Notes in Computer Science, pages 248–263. Springer Berlin / Heidelberg.