A Model Transformation in Model Driven Architecture from Business Model to Web Model

Yassine Rhazali, Youssef Hadi, Idriss Chana, Mohammed Lahmer, and Abdallah Rhattoy

***Abstract*— The models transformation is the fundamental key in Model Driven Architecture approach. In Model Driven Architecture there are two transformations kinds: the CIM to PIM transformation and the PIM to PSM transformation. The researchers focused on the transformation from PIM level to PSM level, because there are several points in common between these two levels. However, the transformation from CIM level to PIM level is rarely discussed in research subjects because they are two completely different levels. This paper proposes a model transformation method to master the transformation from CIM to PIM web-based in accordance with the Model Driven Architecture approach. Indeed, the main value of this method is to propose a transformation from business-oriented models to web-oriented models. This method is founded on building a good CIM level via well-defined rules, allowing us to win precious models that facilitate the task of the transformation from CIM to PIM web-based. Then, we define a set of well- selected rules to move from CIM to PIM web-based so as to ensure a semi-automatic transformation. This method conforms to MDA recommendations by considering the business dimension in the CIM level, and by modeling this latter level using UML activity diagram which is one of OMG standards for modeling business process. Nevertheless, we used UML models into the PIM level, because UML is recommended by MDA in this level.**

***Index Terms*— MDA, computer model, CIM, PIM, business process modeling, model transformation, computer modeling, business model, web model, web modeling**

I. INTRODUCTION

Transformations between different levels of MDA [1] begin by CIM-to-PIM transformation, which allows building PIM models from CIM models by adding in CIM the technical information related to system information design. Then, transformation from PIM models toward PSM models allows adding in PIM a set of technical information related to a target platform. In practice, automatic transformation begins from PIM to PSM. However, our ultimate aim is to make the CIM a productive level, and a basis for building PIM through an automatic transformation. The goal is that business models do not remain only simple documents of communication between business experts and software designers.

In this paper, we present a solution for automating transformation from CIM level business-based to the PIM level web-based. However, we use UML 2 activity diagram to attain concentrated CIM models, which simplifies the transformation to PIM level. Then, we define a set of selected-rules for automating the transformation from CIM to PIM.

Our approach is based on UML 2 [2] activity diagram ,which represents one of standards of business model, for define the CIM level. Then rich business models of well-concentrated information help us to obtain easily models of PIM level. First model of PIM level is use case diagram model allows modeling functionalities of the system. Then, the state diagram model defines the states of information system. Next, package diagram model allows modeling system classes and their relationships independently of a programming language, and allows organizing all classes inside packages. However, these design models contain the necessary and sufficient information for building web model.

Rest of this paper is presented as follows. In section II we analyze the related works concerning transformation from CIM to PIM. Section III presents our proposal and describes constructing rules of CIM models and transformation rules allow shifting from CIM models to PIM models. In Section IV we illustrate our method in a case study demonstrating the construction of the CIM level and its transformation to PIM level. In section V we present model transformation process of our practical case study. In section we presents model transformation process of our practical case study Finally, Section VI, we conclude by specifying outcome of our work and determining future works.

II. RELATED WORK

In this section, we present related works concerning transformation from CIM level to PIM level in MDA [27], describing advantages and disadvantages of each method.

A transformation from CIM to PIM oriented service was presented by Castro et al. [6]. Authors represent CIM level by using BPMN [3] for modeling business process, and by using value model [5] for identifying services from the beginning in the business perspective. Authors use ATL

Manuscript received February 20, 2017; revised November 30, 2017.

Yassine Rhazali is with Department of Computer, Information and Communication Systems Engineering Research Group, Moulay Ismail University, Higher School of Technology, B.P. 3103, 50000, Toulal, Meknes, Morocco. (corresponding author to provide phone: 212-608550620; e-mail: dr.yassine.rhazali@gmail.com).

Youssef hadi is with the MISC Laboratory, Faculty of science, Ibn Tofail University, Kenitra, Morocco. (e-mail: hadiyoussef@gmail.com).

Idriss Chana, Mohammed Lahmer and Abdallah Rhattoy are with Department of Computer, Information and Communication Systems Engineering Research Group, Moulay Ismail University, Higher School of Technology, B.P. 3103, 50000, Toulal, Meknes, Morocco. (e-mail: idrisschana@gmail.com, mohammed.lahmer@gmail.com, rhattoy@gmail.com).

language, for moving towards PIM level; however, this level is presented by two extensions of use case model and two extensions of the activity diagram. This approach has the advantage of identifying services and business process at CIM level in order to guide a semi-automatic transformation to PIM level. But authors study is only limited to use case diagram and activity diagram in PIM level and does not present the structural view (generally represented by the class diagram) that defines the ultimate objective of this level. Nevertheless, the use of activity diagram in the PIM level represent a great inconvenience because this diagram is considered one of standards for modeling business processes.

A transformation method from CIM to PIM, based on security requirements to represent business perspective in CIM is presented by Rodríguez et al. [7]. The authors is based on BPMN notation for modeling business processes into CIM level; then, they use QVT [12] language to transform CIM in order to obtain class diagrams and use case in PIM. This approach shows very interesting ideas for transforming CIM to PIM into security-oriented field. However, this method based only on secure information systems.

Hahn et al. [9] is based on engineering services driven by models. The authors represent CIM level through BPMN, then, they use ATL language to move to PIM level represented in this method by SoaML models. This approach apply SoaML, the new OMG standard to model services [26], but this method does not represent the ultimate aim of PIM level that is presented in use of one or more structural diagrams (as class diagram).

Zhang et al. [10] describe a method in that CIM and PIM are respectively represented by features and components. Responsibilities in this approach are considered as connectors between features and components to facilitate transformation between CIM and PIM. Grammel and Kastenholz [11] rely on a DSL connection for manage traceability in general. Both methods offer solutions for transforming CIM to PIM, while they do not define models used in CIM and PIM.

An approach according MDA, its goal is transforming use case diagram to activity diagram is represented by Gutiérrez et al. [13]. The authors founded on QVT to transform existing use cases to activity diagram. While this method allows to transform CIM to PIM via clear rules, the authors specify in CIM level the functional requirements represented by the use case.

Mazon et al. [14] present an objective-oriented method by defining a UML profile to establish the CIM level, founded on i* modeling framework. The authors based on QVT for moving to PIM, which focuses on conceptual modeling of data warehouse. However, this method deals the transformation from CIM to PIM only in the field of data warehousing.

Kherraf et al. [8] define a method for transforming CIM to PIM. The authors use business process model and use case diagram as an initial step in the modeling of business processes, then a detailed activity diagram which defines the system requirements represents the last step in the CIM level. The system requirements are transformed as components of the system. These are shown in the component diagram as a first stage in the PIM. Finally, a set of business archetypes helps to transform the components of system to class diagram. This method proffers interesting ideas allow to facilitate the transformation from CIM to PIM. Nevertheless, this approach uses diagram of use case in the CIM level even if it represents the system functionalities.

Kardos et al. [15] propose an analytical method for transforming CIM to PIM in MDE (model driven engineering) [27]. The authors use data flow diagram in the CIM level; then they base on the use case diagram for introduce the information system view. This method defines a model of activity diagram as well as a model of sequence diagram and a class diagram model. The benefit of this approach is the use of several UML diagrams that present different views of the information system in PIM, but this approach does not represent a real business view since it based on DFD in CIM level, and does not clearly describe transformation rules from CIM to PIM.

After this overview on related works about transformation from CIM level to PIM level, we can divide the works into five types. We note approaches [8], [13] that use model of system requirements (as use case diagram) early in the CIM level, to facilitate transformation to PIM level. However, other works [6], [9] even if they base on business processes in CIM level, do not establish the structural view (usually through the class diagram) in PIM level. Nonetheless, there are methods which aim transformation in a particular area [7], [14]. Then, there are researches like in [15] that establish the structural view in the PIM level and are not oriented to a particular field, but the authors do not specify rules of transformation. Finally, there are approaches [10], [11] which describe precisely the transformation rules, but do not have the models used in the CIM and PIM.

III. Proposed Method of Transformation from CIM to PIM

A business model describes abstractly how the business is working. According to our modeling needs we can, more or less, make different models to describe the same reality. However, we can model business process to improve communication with customers or partners, for controlling business process, or to establish an information system. In this paper, our object is to design business process models as a first stage in the process of information system development. In our case, we have two alternatives: either create business models in the form of documents that will be transformed manually via analysts and software designers, or to create business models that will be transformed in an automatic way. The second choice was regarded in our method for designing effective business models that contain rich information to facilitate the transformation. However, we respect MDA approach in the conception and transformation of business models.

Our study considers business dimension into the presentation of CIM level, through the use of real business model, to preserve the knowledge of the business throughout the transformation towards PIM level. This enables us the

realization of quality information system.

UML 2 activity diagram is one of standards of business modeling. In our method, we use effectively UML 2 activity diagram to present two extensions of business processes models for achieving a concentrated CIM level that help us to obtain readily models of PIM level.

MDA recommends the use of UML in the PIM level. The use case presents functional view. Then, the state diagram interprets the dynamic view, the class and package diagram shows the static view of the information system, finally, modeling diagram show the web view (cf. Fig. 1).

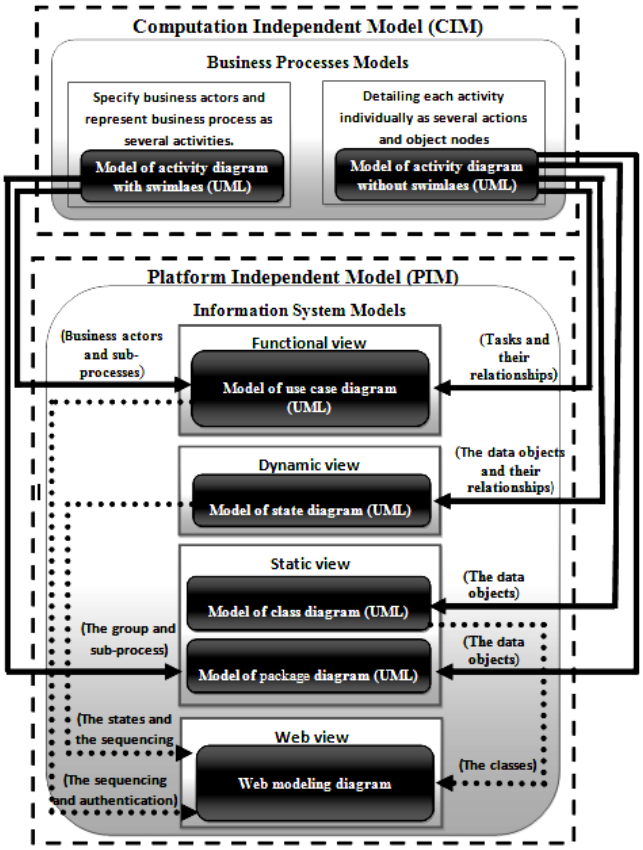

Fig. 1. Schema of the proposed approach.

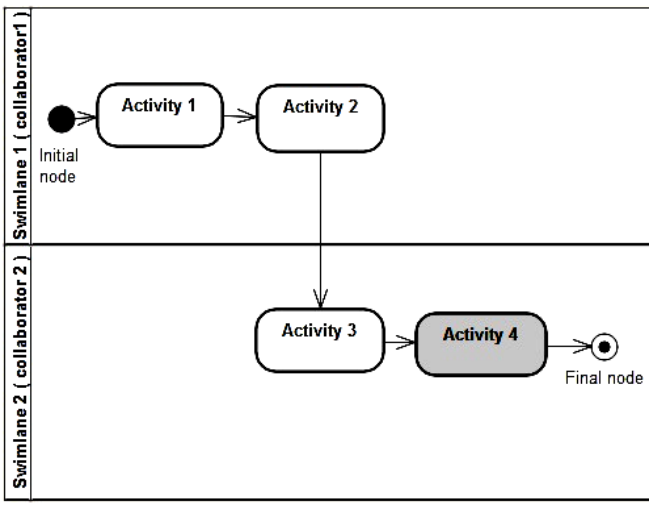

Fig. 2. Generic model of UML activity diagram (with swimlanes).

All UML design models in PIM level are obtained by an automatic or semi-automatic transformation from CIM level. However, web model is obtained thought an automatic transformation from design PIM models. Transformations are realized through concentrated rules.

A. Construction Rules of CIM Level

The rules for constructing the general model of activity diagram with swimlanes (cf. fig. 2):

- Define averages activities (not complexes activities). In fact, each activity must be comprised between 4 and 10 actions.
- If an activity consists of less than 4 actions, or represents a complementary operation to another activity, we can merge several activities into one. Provided that the activity do not exceed 10 actions.
- Avoid the maximum possible, the representation of actions.
- The model does not describe all cases and paths, but it presents just a description of the sequence of activities of the most common business processes.
- Focus on activities and their sequences.
- Coloring manual activities with another color for example we used the gray
- Use the "Region" notation to group activities that belong to the same category.
- Identify the maximum possible of the actors who interact and who collaborate in the achievement of business processes since we are talking about an enterprise process.
- Avoid in this model, the maximum possible, representation of the gateways.

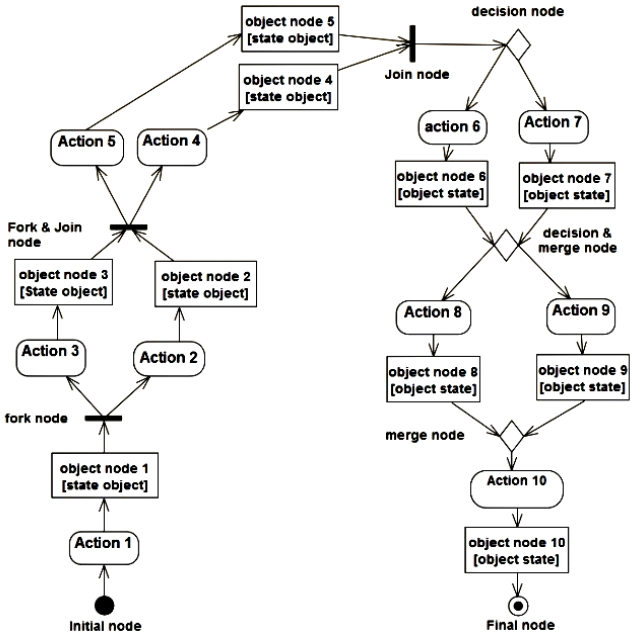

Fig. 3. Geniric model of UML activity diagram (without swimlanes).

The rules of construction of detailed model of the activity diagram without swimlanes (cf. Fig. 3):

- Detail individually each activity in a model as a several actions (this latter constitute the fundamental unit in the activity diagram).
- Do not present, into this model, manual tasks of general model of activity diagram (with swimlanes).

- Show connections in this model.
- Representing in this model of the most outstanding ways.
- Any action described in model of activity diagram (with swimlanes) will be shown here by an action.
- Add an object node, which contains object state, at the output of each action.

B. Transformation Rules from CIM to PIM

The rules of passage from models of the activity diagram (with and without swimlanes) to model of use case diagram (cf. Fig. 4):

- Every action of detailed model of activity diagram that corresponds to a functionality of the system is transformed to use case.
- The collaborator, who realizes the activity of general model of activity diagram, becomes an actor use cases that correspond to the actions of this activity.
- If there is "decision node" between two actions, the corresponding use cases connect by a relationship "*extend*".
- Do not transform the control flow returning back
- If there was just a control flow between two actions, the corresponding use cases connect by a relationship "*include*".
- Each activity of general model of activity diagram is transformed to a package which includes the use cases corresponding to the actions of this activity

The rules of passage from detailed model of the activity diagram (without swimlanes) to model of state diagram (cf. Fig. 5):

- An object node transforms to a state.
- A decision node transforms to a decision point.
- A merge node transforms to a junction point.
- A decision and merge node transforms to a junction point.
- An initial node transforms to an initial state.
- A final node transforms to a final state.
- A control flow between two actions transforms to a transition.
- A fork node transforms to a fork state.
- A joint node transforms to a joint state.
- A joint and fork node transforms to a joint and fork state.

The rules of passage from detailed model of the activity diagram to the model of class diagram (cf. Fig. 6):

- An object node transforms to class.
- A state of an object node transforms to a class method.

The rules of passage from general model of activity diagram (with swimlanes) and the model of class diagram to the model of package diagram (cf. Fig. 7):

- A region transforms to a package.
- An activity which does not belong to any region becomes a package
- A set class, which becomes the same region, will be placed in the package corresponding to the region.
- Classes coming from the same activity, that does not located in any region, will be placed in the package matching to the activity.

The rules of passage from design models, use case, state, and class diagram model, to web modeling diagram model (cf. Fig. 8):

- A class transforms to a web page name.
- If the class has several methods, each method transforms to a page web.
- Each web page have pre-condition and post-condition.
- Each state becomes a post-condition in the web page.
- If the web page has in the direct previous input a link, the pre-condition of the web page will be " name page + chosen ".
- If the web page has in the direct previous input a submit, the precondition of the web page will be "confirmed + name of realized functionality in the previous web page (use case that corresponds the previous web page)".
- The sequencing of the web pages must match the sequencing of the states in state diagram model (each transition becomes link, submit, or redirect).
- The functionality, authentication, presented in use case diagram model transforms to an authentication web page to ensure the security of the information system.
- The authentication page contains two text areas elements: the login and password. This page must have as first output a "submit" liaison to the next page if the login and password are correct, however, it must have as second output a reflexive liaison "redirect" if login and password are incorrect.
- The main page should be added as a first web page, or a second web page if the authentication precedes all functionality.
- Each web page that contains a "text area element" must have in output a" submit" liaison.
- When there are multiple outputs in a page, which contains text area elements, this latter page will have a reflexive submit, while the other outputs will be interpreted as links.
- The link name is the name of the destination web page.

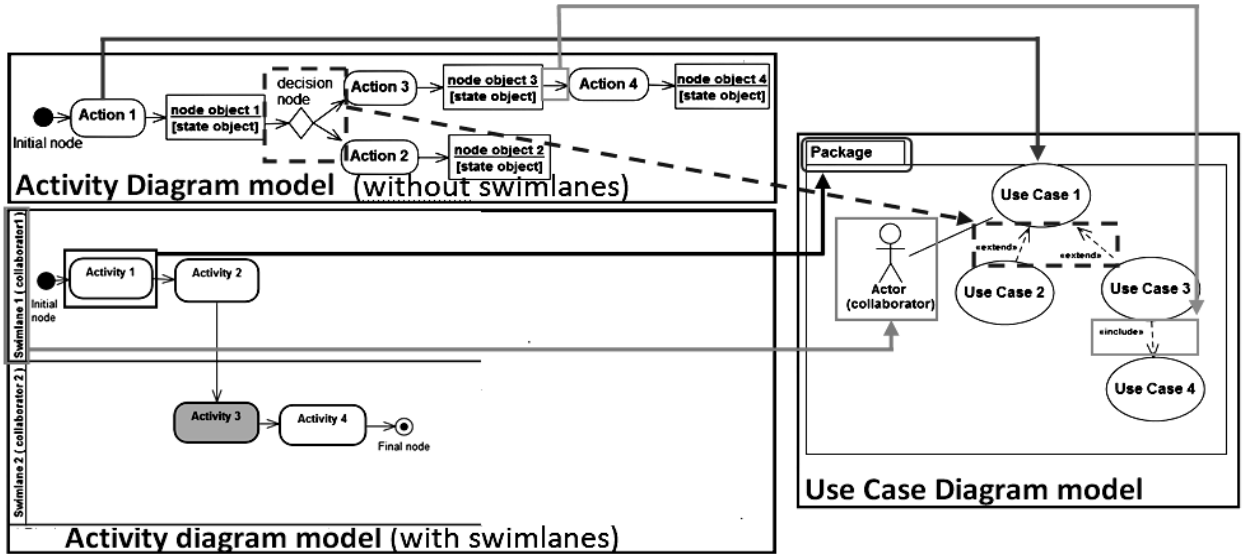

Fig. 4. Schema of passage from activity diagram models (with and without swimlanes) to use case diagram model

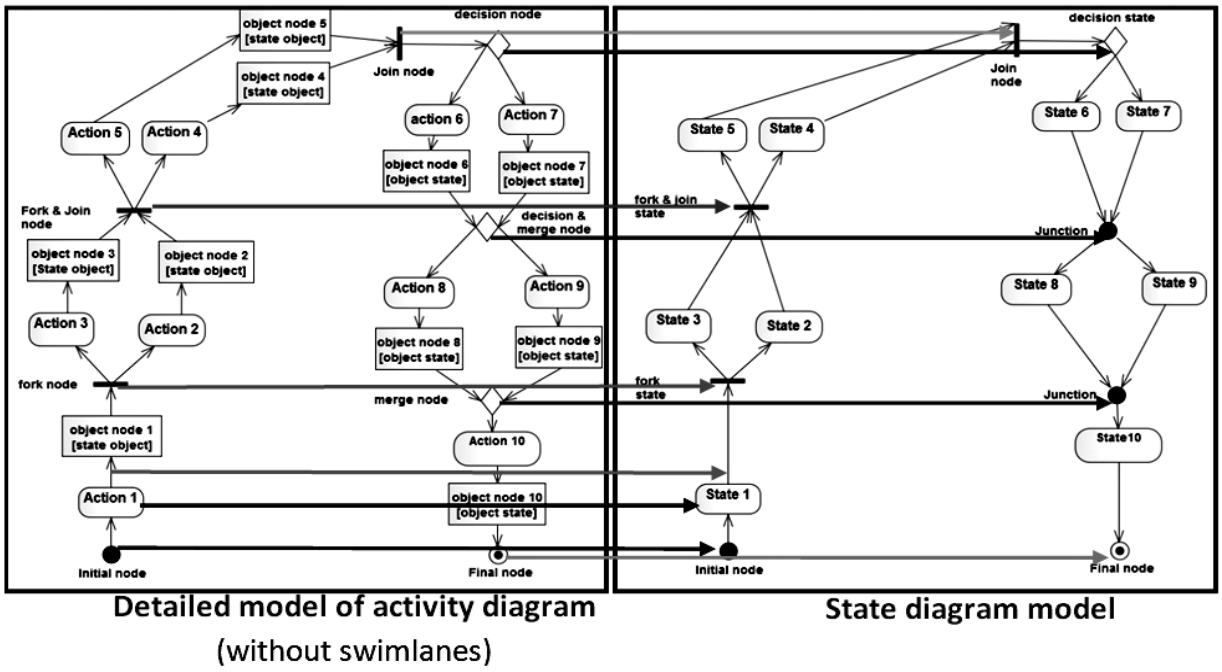

Fig. 5. Schema of passage from detailed model of activity diagram (without swimlanes) to class diagram model

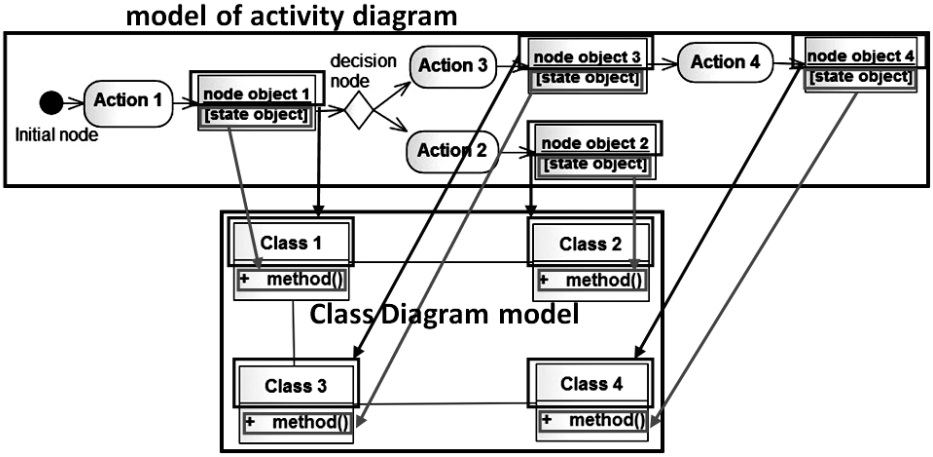

Fig. 6. Schema of passage from detailed model of activity diagram (without swimlanes) to class diagram model.

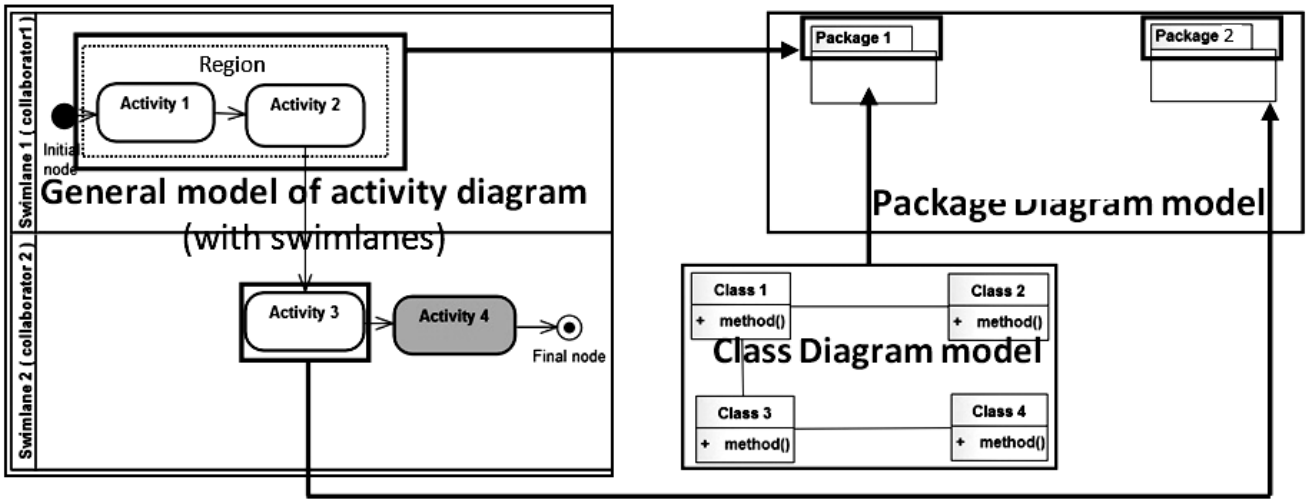

Fig. 7. Schema of passage from general model of activity diagram (with swimlanes) and class diagram model to package diagram

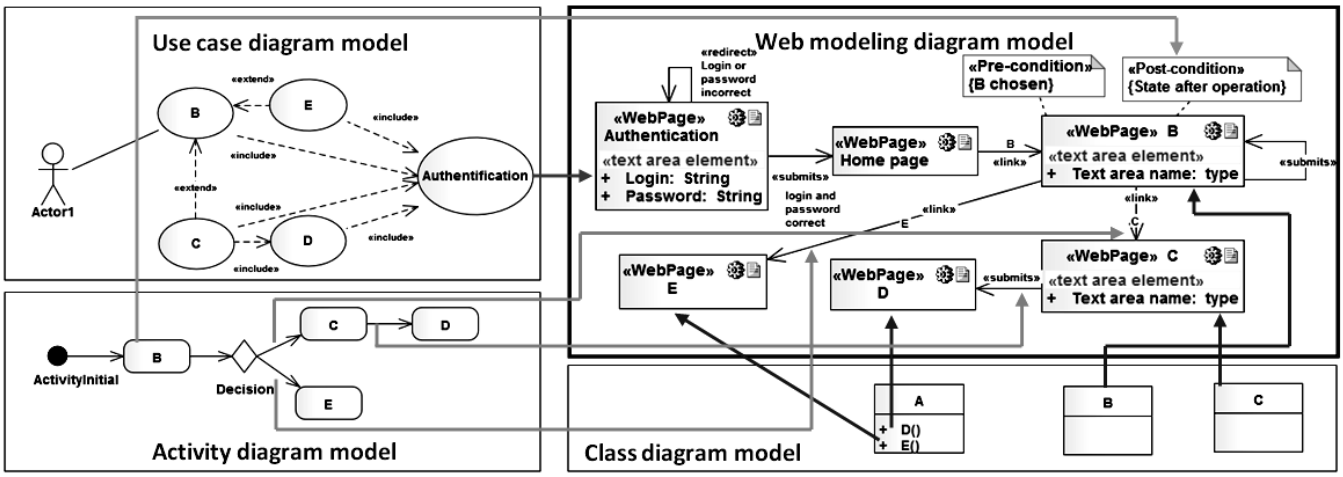

Fig. 8. Schema of passage from design models, use case, state, and class diagram model, to web modeling diagram model.

IV. Case Study

In this section, we show a case study of order products to illustrate our method of transforming CIM level to PIM level.

A customer can show items available in the catalog. He can also see detailed information concerning each product, then he decides to put a quantity of product in cart or not. In any moment the customer has the right to modify the quantity or eliminate entirely the product from the cart. Once products that match to the needs are definitely selected by the customer, the latter can start the command. Then, he shows payment information, and chooses delivery details.

An order agent treats the order, and declares the reservation of products chosen by the customer. Then, the assembly worker collects manually the reserved articles from stock.

For each product the assembly team leader inspects the quality and quantity. Then the delivery agent delivers the confirmed order, in order that customer obtains his products.

A. Presentation of the CIM Level

Fig.9 shows first model in CIM level, through business process established by our general model of activity diagram (with swimlanes). In this model we just specify the activities and their sequencing, by avoiding the identification of actions and gateways for presenting business process in general. Nevertheless, we have set the maximum of collaborators for representing a true business process, in which there is collaboration between several business actors, e.g. instead of placing a single lane "delivery service", we identified the lanes: "assembly worker", "assembly team leader" and "delivery agent".

Fig. 10 shows the second model in CIM level as a detailed model of activity diagram (without swimlanes). In this model we detail individually each activity of the previous model as a several actions. However, in this model the activity "select product for order" is analyzed. Then, we have presented all possible paths. Then we identify an object node together with its state in the output of each action.

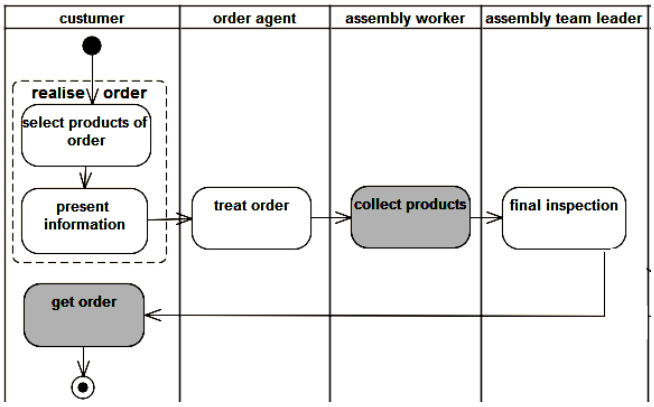

Fig. 9. Model of activity diagram (with swimlanes) of “order products”.

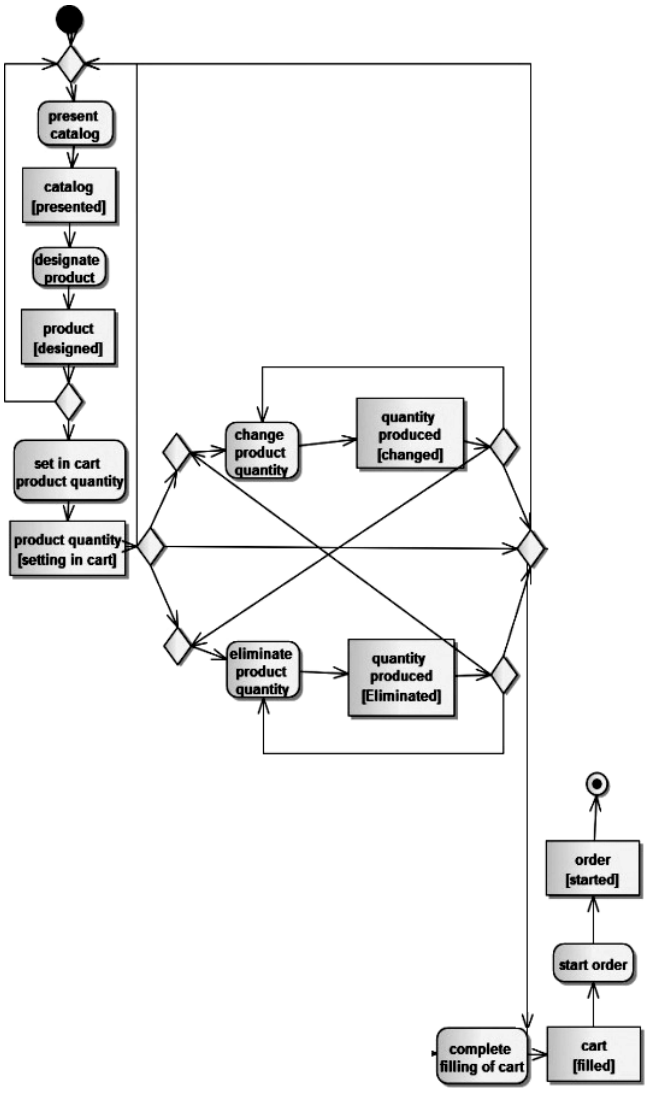

Fig. 10. Activity diagram model of “select products of order”.

B. Presentation of the PIM level

Fig. 11 presents a model of diagram use case. This model is transformed from the business models of CIM level. However, in this model the activity “select product for order” is transformed to a package. Then, the collaborator "customer" who performs the activity becomes actor. Then the actions that detail the activity are transformed to use cases. Decision nodes that lie between two actions become relationship "extend". For example, in this model there is a decision node between the two actions "designate product" and "put in cart quantity product "; so the two correspondent use cases are connected via an "extend" relationship. Control flows that lie between two actions become relationship "include." Thus, in this model there is flow control between the two actions "present catalog" and "designate product," so the two corresponding use cases are connected via an "include" relationship. However, it is not presented in this model the flows which return backward. For example, the relationship between the action "put in cart product quantity" and "present catalog" is not specified in this model, so as not to complicate the model, and so that the diagram use case would not focus only on the identification of functionality and not on the sequence.

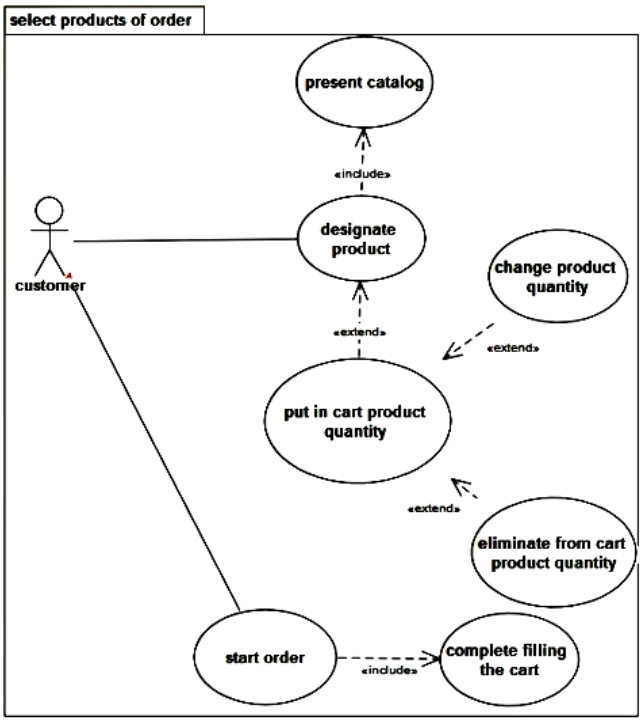

Fig. 11. Use case digram model of “select products of order”.

Fig. 12 shows state diagram model transformed from the detailed model of activity diagram of CIM. In this model the states are obtained from nodes of objects. Then, the control flow which connects two actions is transformed to a transition. E.g. the object node "catalog" with state "presented" becomes "catalog presented" in state diagram model. Then, initial state is transformed from initial node; final node becomes to a final state; node fusion transformed to junction point; decision node becomes a decision point and decision and fusion node transformed to a junction point.

Fig. 13 represents class diagram model, which is the final goal of the PIM level. This model is obtained from the detailed model of activity diagram. In this model object nodes are transformed to classes. Then functions of the class are transformed from states of an object. E.g. the object node "order" with state "started" transform to class "order" that holds the "start" method.

Fig. 14 represents model of package diagram. So the region "realize order" becomes package. Then the activities that are not in a region, such as "treat order" and "final inspection" are transformed to packages.

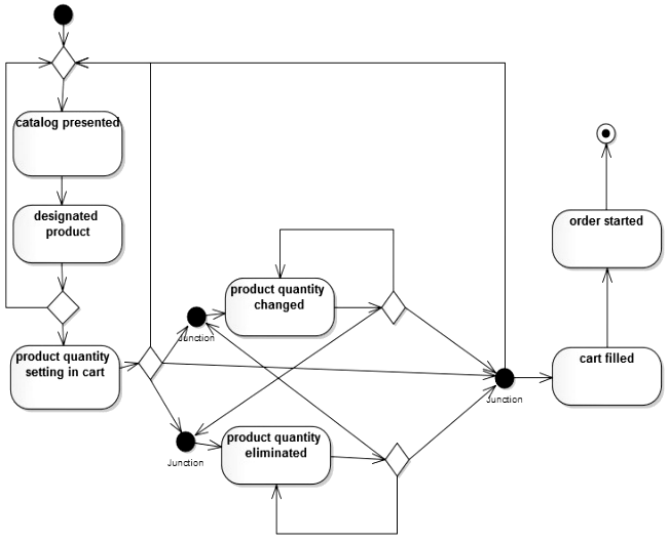

Fig. 12. State digram model of “select products of order”.

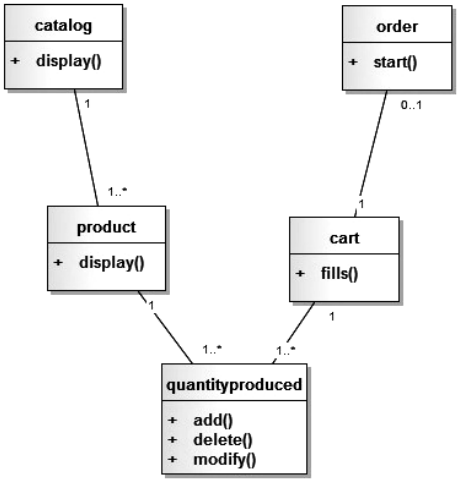

Fig. 13. Class digram model of “select products of order”.

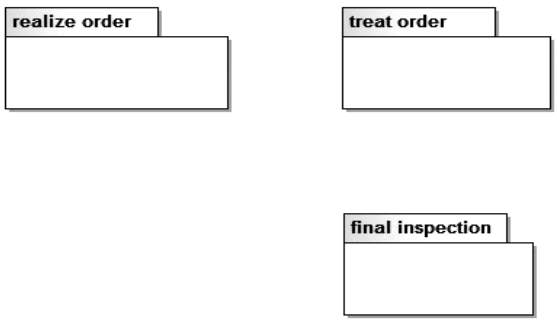

Fig. 14. Package digram model of “order products”.

Fig. 15 represents model of web modeling diagram. This last model transformed from design models: use case diagram model, state diagram model, class diagram model.

V. META-MODELS AND PRACTICAL CASE

The ATL rules of passage from models of the activity diagram to model of use case diagram (cf. Fig. 16 & Fig. 17):

Transformation rules in human language	Transformation rules in ATL
R1: every “action” corresponds to a system functionality is transformed to a “use case”.	`rule R1 { from ts : MMActivity!TAction ((not ts.isManual()) and ts.isTransformableTask()) to uc : MMusecase!Usecase ( name <- ts.name, extendIn <- ts.gatewayOut, extendOut <- ts.gatewayIn, includeIn <- ts.flowOut, includeOut <- ts.flowIn, belongsClassifier <- ts.belongsActivity ) }`
R2: each “lane” becomes an “actor”	`rule R2 { from ln : MMActivity!Lane (ln.isTransformableLane()) to act : MMusecase!Actor ( name <- ln.name ) }`
R3: “gateway xor” between two “actions” corresponds to relationship "extend" between two “use cases”.	`rule R3 { from gwo : MMActivity!Or (gwo.isTransformableGatewayOr()) to extd : MMusecase!Extend ( name <-gwo.name ) }`

Fig. 16. The ATL rules of passage from models of the activity diagram to model of use case diagram (part 1).

R4: “transition” between two “actions” corresponds to relationship "include” between two “use cases”. **R5:** “transition” returning back is not transformable.	`rule R3R4 { from sqc : MMActivity!transition ((not sqc.isReturnBack()) and sqc.isTransformabletransition() ) to icld : MMusecase!Include ( name <- sqc.name ) }`
R6: Each “activity” is transformed to a “package”.	`rule R6 { from sbp : MMactivity!Activity (sbp.isTransformable Activity()) to clf : MMusecase!Classifier ( name <- sbp.name, containsActor <- sbp.belongsLane ) }`

Fig. 17. The ATL rules of passage from models of the activity diagram to model of use case diagram (part 2).

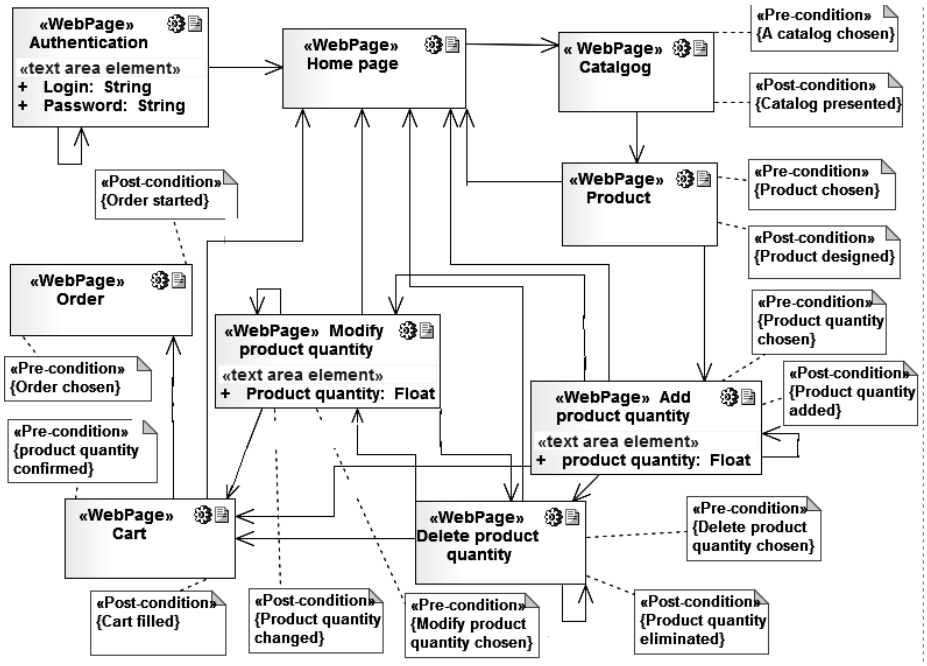

Fig. 15. Web modeling digram model of “order products”.

The ATL rules of passage from models of the activity diagram to model of state diagram (cf. Fig. 18 & Fig. 19):

Transformation rule in human language	Transformations rules In ATL language
R7: Each object node becomes a state.	`rule R7 { from dto : MMActivity!ObjectNode (dto.isTransformableObjectNode()) to nrstt: MMstatemachine!NormalState ( name <- dto.name +' '+ dto.state ) }`
R8: Each exclusive fork becomes a decision point.	`rule R8 { from exl : MMActivity!ForkExclusive (exl.isTransformableForkExclusive()) to dcs : MMstatemachine!DecisionState ( name <- exl.name )}`
R9: Each exclusive join becomes a junction point.	`rule R9 { from exlj : MMActivity!JoinExclusive (exlj.isTransformableJoinExclusive()) to jct : MMstatemachine!Jonction ( name <- exlj.name )}`
R10: Each parallel fork node becomes a fork state.	`rule R10 { from frkp : MMActivity!ForkParallel (frkp.isTransformableForkParallel()) to frks : MMstatemachine!ForkState ( name <- frkp.name )}`
R11: Each parallel join becomes a joint state.	`rule R11 { from jnp : MMActivity!JoinParallel (jnp.isTransformableJoinParallel()) to jns : MMstatemachine!JointState ( name <- jnp.name) }`
R12: Each parallel joint and fork becomes a joint and fork state.	`rule R12 { from prll : MMActivity!parallel (prll.isTransformableparallel()) to frkjns : MMstatemachine!ForkandJointState ( name <-prll.name) }`
R13: Each exclusive fork and join becomes a junction point.	`rule R13 { from exlsv : MMActivity!Exclusive (exlsv.isTransformableExclusive()) to jct : MMstatemachine!Jonction ( name <- exlsv.name)}`
R14: Each start event is transformed to an initial state.	`rule R14 { from strtevnt : MMActivity!Start (strtevnt.isTransformableStartEvent()) to intstt: MMstatemachine!InitialState ( name <- strtevnt.name)}`

Fig. 18. The ATL rules of passage from models of the activity diagram to model of state diagram (part 1).

R15: Each end event becomes a final state.	`rule R15 { from ende : MMActivity!End (ende.isTransformableEndEvent()) to fnls : MMstatemachine!FinalState ( name <- ende.name)}`
R16: Each transition becomes a transition.	`rule R16 { from sqcf : MMActivity!transition (sqcf.isTransformableSequenceFlow()) to trst : MMstatemachine!Transition ( name <- sqcf.name, source <- sqcf.source(), target <- sqcf.target() ) }`

Fig. 19. The ATL rules of passage from models of the activity diagram to model of state diagram (part 2).

The ATL rules of passage from models of the activity diagram to model of class diagram (cf. Fig. 20):

Transformation rules in human language	Transformation rules in ATL
R17: Each “state” of a “object node” becomes a “class method”.	`rule R17 { from stt : MActivity!StateObject (stt.isTransformableStateObject()) to opr : MMclass!Operation ( name <- stt.name) }`
R18: each “object node” is transformed to a “class”.	`rule objectnode2class { from dtobj : MMActivity!ObjectNode (dtobj.isTransformableObjectNode()) to cls : MMclass!Class ( name <- dtobj.name, operations <- dtobj.stateobject.name)}`

Fig. 20. The ATL rules of passage from models of the activity diagram to model of class diagram.

The ATL rules of passage from models of the activity diagram to model of package diagram (cf. Fig. 21):

Transformation rules in human language	Transformation rules in ATL

R20: Each "region" becomes a "package".	`rule R20 {`
	`from`
	`grp : MMActivity!Region`
	`(grp.isTransformableRegion())`
	`to`
	`pck : MMpackage!Package ( name <- grp.name ) }`

R21: Each "Activity" that does not belong to any "group" transforms to a "package".	`rule R2 {`
	`from`
	`subp : MMActivity!Activity ((not`
	`subp.activityBelongsRegion()) and`
	`subp.isTransformableAcivity() )`
	`to`
	`pck : MMpackage!Package ( name <- subp.name ) }`

R21: Each set of "classes", become from the same "region", will be placing in the "package" that matches the "region".	`rule R21 {`
	`from`
	`dtobj : MMActivity!ObjectNode`
	`(dtobj.ObjectBelongsGroup() and`
	`dtobj.isTransformableObjectnode())`
	`to`
	`cls :MMpackage!Class ( name <- dtobj.name,`
	`operations <- dtobj.stateobject,`
	`belongsPackage <- dtobj.belongsRegion ) }`

R22: "Classes, resulting from the same "Activity" which belongs to no "region", will be placing in the package that corresponds to the Activity.	`rule R22 {`
	`from`
	`dtobj : MMActivity!ObjectNode ((not`
	`dtobj.dataBelongsRegion()) and`
	`dtobj.isTransformableObjectNode())`
	`to`
	`cls :MMpackage!Class ( name <- dtobj.name,`
	`operations <- dtobj.stateobject,`
	`belongsPackage <- dtobj.belongsActivity ) }`

Fig. 21. The ATL rules of passage from models of the activity diagram to model of package diagram.

The (Fig. 22) presents model transformation process of our practical case study. We generate system information model, conforming to the PIM meta-model (UML meta-model), from business model which conforms to the CIM meta-model (Activity diagram meta-model). The transformation is described by transformation rule model that is conforming to ATL meta-model. This last model is based on CIM and PIM meta-model. However, the meta-models CIM, PIM and ATL are conformed to Ecore meta-meta-model that is defined by the Eclipse Modeling Framework (EMF).

In (Fig. 23) we structured our practical case study in three folders. Folder of meta-model contains CIM and PIM meta-models through two extensions (.ecore and .ecore_diagram). Business and system information models are putted in model folder with .xmi extension. The rules are grouped into folder of transformation rules, through extensions (.asm and .atl).

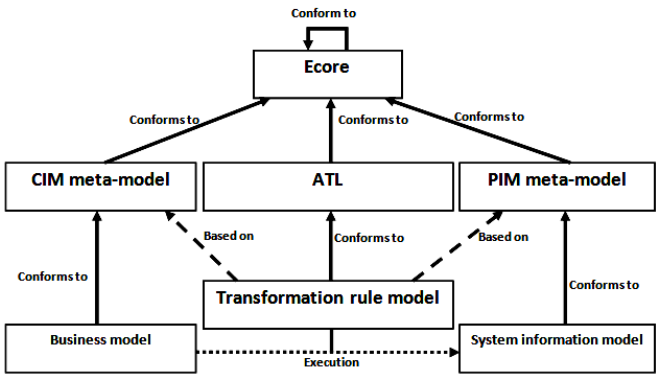

Fig. 22. Model transformation process of practical case study

In this section, we validate our idea on the practical level. Thus, we illustrated in (Fig. 24) a part of activity diagram model which represents the first model in the CIM level.

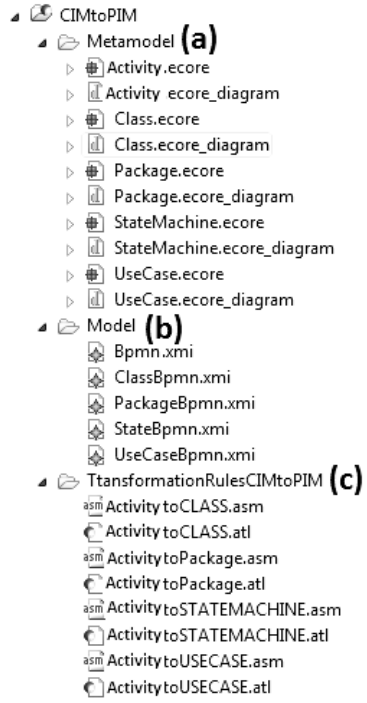

Fig. 23. Structure of practical case: (a) folder of meta-models (b) folder of models (c) folder of transformation rules

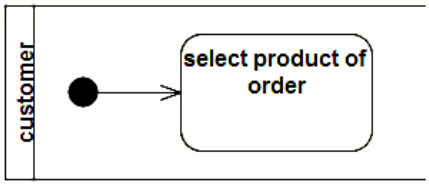

Fig. 24. A portion of the activity diagram model for practical case

The activity diagram model is developed into activity diagram detailled model which represent the second model in the CIM level (Fig. 25). Then, we transformed the two previous models to use case diagram model (Fig. 26) that represents, in this case, the PIM model. We have executed a transformation to the all PIM models discussed in this paper, but in this section we shown only the transformation to the use case diagram model for not overburden the practical case.

We created CIM model as XMI model (in tree structure) in eclipse from models (Fig. 24, Fig. 25).

Through the model transformation language ATL, we automatically transformed CIM model to use case diagram model (Fig. 26). This last model is presented as PIM model.

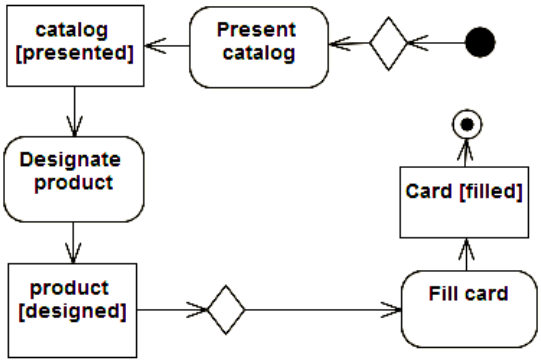

Fig. 25. A portion of development of the activity "select product of order" in model of the activity diagram

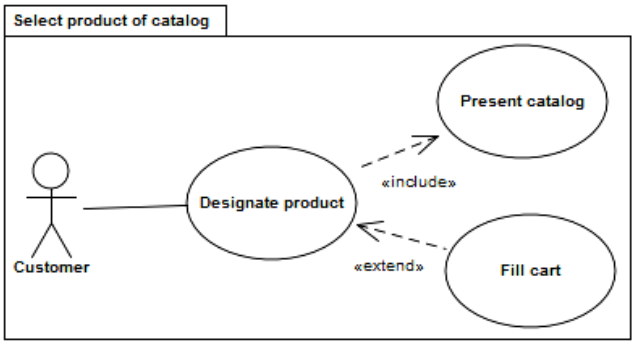

Fig. 26. Use case diagram model of patical case "select product of catalog"

The (Fig. 28) shows the CIM meta-model, which is designed as Activity diagram meta-model and the (Fig. 29 and Fig. 30) show the PIM meta-model, which is presented by use case meta-model, state meta-model, class meta-model and package meta-model.

VI. Analysis And Discussion Based on Analytical Survey

A. Evaluation Criteria

From the conceptual framework [25], we deduce the evaluation criteria of CIM to PIM transformation approaches. According the (Fig. 27), the evaluation criteria are CIM business modeling, PIM completeness, transformation elements, and Assessment methodology. Indeed, we deduce the "business modeling of CIM" from the "CIM taxonomy", the "PIM completeness" is derived from the "PIM taxonomy", however, "transformation elements" comes from "transformation approach", and the "Assessment methodology" is not related to any conceptual framework element, reciprocally, the "static model" not derived from any evaluation criteria.

- Evaluation criterion for CIM :

The CIM level must present the business process through one or more business models. However, according [8], [23] BPMN is a standard for modeling business process.

- Evaluation criterion for PIM :

The PIM level is considered complete if it contains one or more models for each modeling view [23]. The key model is class diagram, since it contains IS structure and it is easily transformable towards PSM.

- Evaluation criterion for transformation :

Model transformation based on source metamodel, target metamodel, and transformation rules. These last rules may be described by human language which has less value. However, algorithm and programming language are more expressive, but model transformation languages considered the most effective in model transformation.

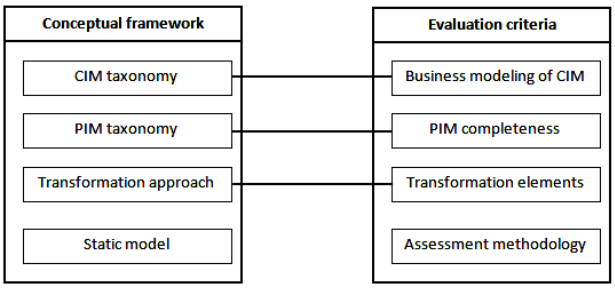

Fig. 27. Evaluation criteria

- Assessment methodology :

It is the method which approves CIM to PIM transformation approach. This method may be a theoretical case study, or a practical transformation through a standard tool as eclipse or a personal tool developed for supporting transformation.

B. Analysis

We based on criteria evaluation, concluded from the conceptual framework, for comparing the CIM to PIM transformation approaches (Fig. 31).

The actual method presents an improved approach from old approaches [16], [17], [18], [19], [20], [21], [22], [23], [24], [25]. These approaches give several methodologies for transforming CIM to PIM but they are based on BPMN to create CIM level for this we are limited when we transform CIM level into the PIM level. For example with BPMN we cannot present object state in CIM models etc...

We find that the approach [6] validates most of our criteria. This approach allows building the CIM level on the basis on value model that is not a business modeling standard. Then the PIM level is presented just by the models of the use case and the activity diagram, which makes the transformation to PSM very difficult. Indeed, in the PIM level, there are not classes on which we base to move toward code models in PSM. However, this method does not provide clear rules to transform the CIM level to the PIM level.

Our approach is the unique method based on construction rules for structuring CIM in order to facilitate the transformation toward the PIM. However, the builder of CIM level must produce models intended to be transformed to PIM, by using optionally several refinements on the base models and by respecting our construction rules of the CIM.

Approaches of related work do not provide clear and structured transformation rules. In most approaches, we do not find any description of the rules; the reader must deduce the rules from the case study. In the rest approaches there are subsections which contain just rules hints. Our approach describes clear transformation rules with graphic presentation.

VII. Conclusions and Future Work

One of the major challenges in software development process is the establishment of a methodology that allows shifting from models that present the running of the business to models which describe the analysis and design of software.

Founded on MDA, our methodology provides a solution to the difficulty of transformation from business models, represented in CIM level, to the analysis and design models, modeled in PIM level. This method achieves a set of well organized and useful classes in the process of software development.

The ongoing work is intended to implement, construction rules and transformation rules, in a tool through the ATL language. However, in our future work, we anticipate to transform the models obtained in the PIM level to models of PSM level, indeed, our ultimate objective is to provide the source code from the business models by means of automatic transformations and test it [28].

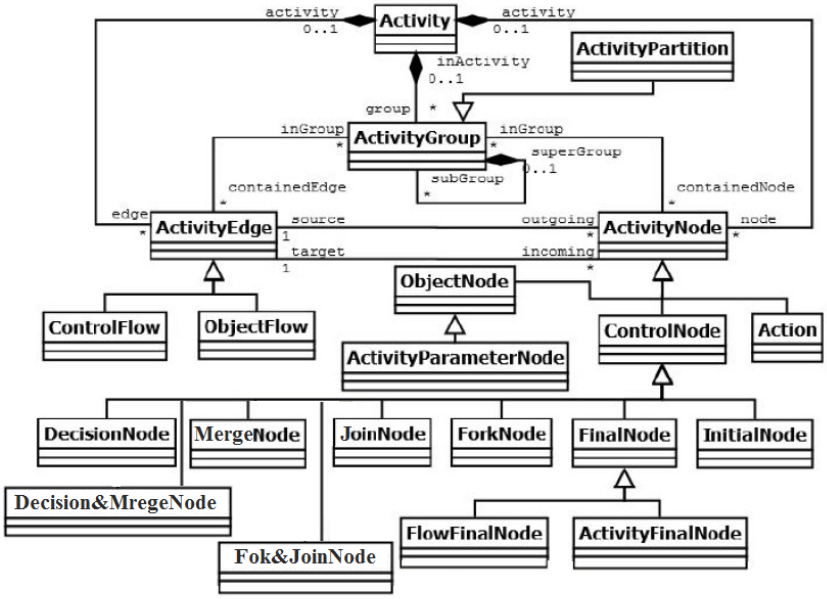

Fig. 28. CIM meta-model: activity diagram meta-model.

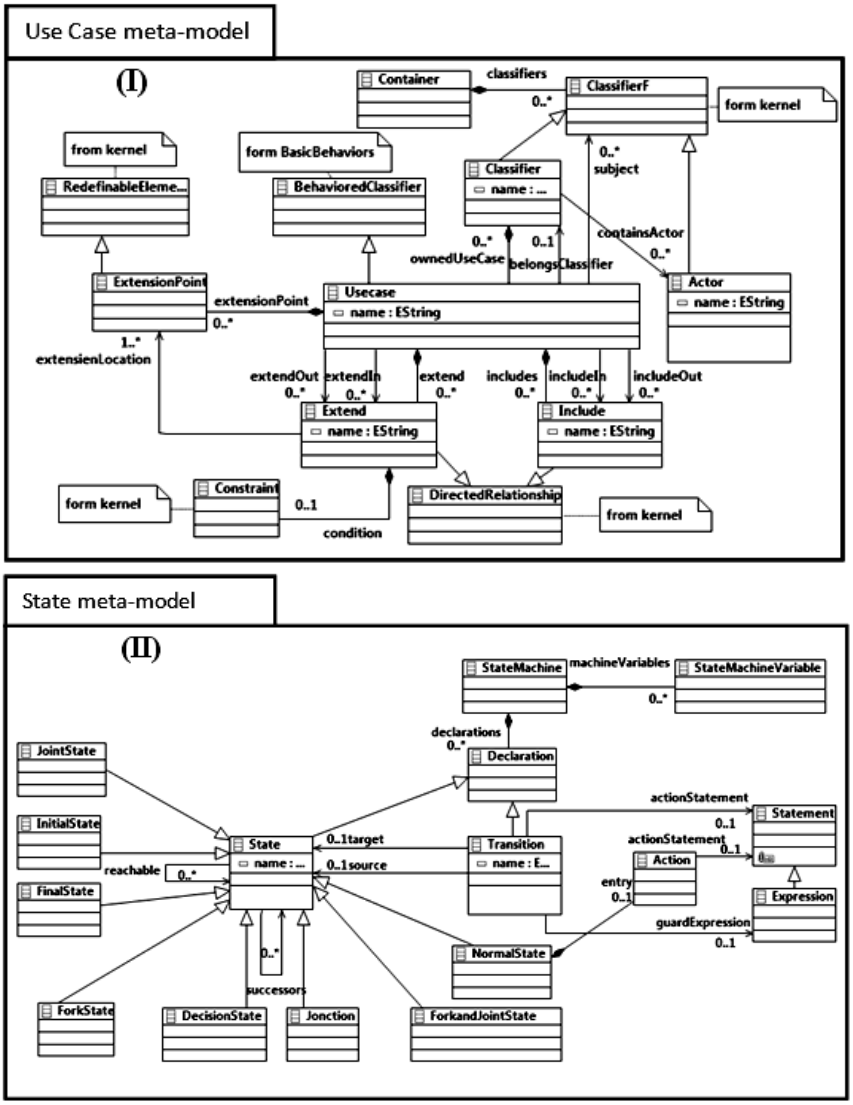

Fig. 29. PIM meta-model: (I) use case meta-model, (II) state meta-model.

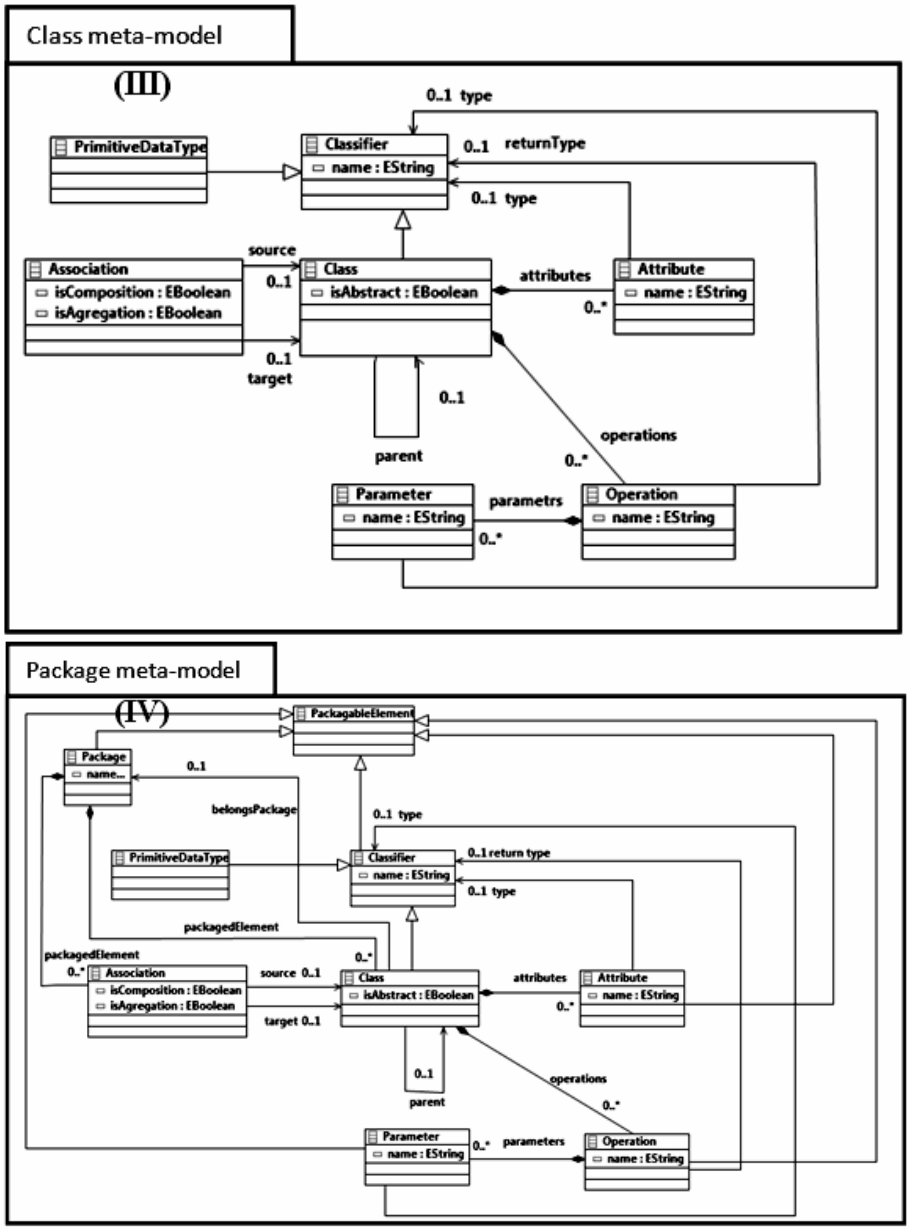

Fig. 30. PIM meta-model: (III) class meta-model, (IV) package meta-model.

Studied Papers	Business modeling of CIM		Completeness of PIM						Transformation				Assessment methodology
			Functional view		*Static view*		*Dynamic view*		*Source meta-model*	*Target meta-model*	*Transformation rules*		
	Attained	Representation	Attained	Representation	Attained	Representation	Attained	Representation			Category	Representation	
Kherraf et al. [8]	Yes	UML activity diagram			Yes	UML Component & class diagram					Human language		Case study
Zhang et al. [10]											Algorithm		Case study
Kardoš et al. [15]	Yes	DFD	Yes	UML Use Case & Sequence Diagram	Yes	Domain diagram	Yes	UML Activity diagram			Human language		Case study
Rodríguez et al. [7]	Yes	BPMN	Yes	UML Use Case diagram							Model transformation language	QVT	Case study
Castro et al. [6]	Yes	BPMN	Yes	UML Use Case diagram			Yes	UML Activity diagram	Yes	Yes	Model transformation language	ATL	Case study & practice in standard tool (eclipse)
Hahn et al. [9]	Yes	BPMN									Model transformation language	ATL	
mazón et al. [14]											Model transformation language	QVT	Case study
Gutiérrez et al. [13]							Yes	UML Activity diagram	Yes	Yes	Model transformation language	QVT	Case study
Rhazali et al. [16]	Yes	BPMN & UML activity diagram	Yes	UML Use Case diagram	Yes	UML class diagram	Yes	UML state diagram			Human language		Case study
Rhazali et al. [23]	Yes	UML activity diagram			Yes	UML class & package diagram	Yes	UML state diagram			Human language		Case study
Our proposal	Yes	UML activity diagram	Yes	UML Use Case diagram	Yes	UML class & package diagram	Yes	UML state diagram	Yes	Yes	Model transformation language	ATL	Case study & practice in standard tool (eclipse)

Fig. 31. Comparison of studied papers through Evaluation criteria.

REFERENCES

[1] OMG-MDA, "MDA Guide Version 1.0.1," 1 juin 2003.

[2] OMG, UML Superstructure 2.0. OMG Adopted Specification ptc/03-08-02, 2003. <http://www.uml.org/>.

[3] OMG, "Business Process Modelling Notation", Version 2.0, 2011.

[4] C. Schmidt, "COVER FEATURE Model Driven Engineering", 2006.

[5] J. Gordijn, and J. M. Akkermans, "Value based requirements engineering: exploring innovative e-commerce idea," Requirements Engineering Journal 8 (2), 2003, pp. 114–134.

[6] V. D. Castro, E. Marcos, and J. M. Vara, "Applying CIM-to-PIM model transformations for the service-oriented development of information systems," presented at 2nd Information and Software Technolog , pp. 87–105. 2011.

[7] A. Rodríguez, I. García-Rodríguez de Guzmán, E. Fernández Medina, and M. Piattini, "Semi-formal transformation of secure business processes into analysis class and use case models: an MDA approach," presented at 9th Information and Software Technology 52, pp. 945–971. 2010.

[8] S. Kherraf, E. Lefebvre, and W. Suryn, "Transformation from CIM to PIM using patterns and archetypes," presented at 19th Australian Conference on Software Engineering, pp. 338-346. 2008.

[9] C. Hahn, P. Dmytro, and K. Fischer, "A model-driven approach to close the gap between business requirements and agent-based execution," presented at Proceedings of the 4th Workshop on Agent-based Technologies and applications for enterprise interoperability, Toronto, Canada, pp. 13–24. 2010.

[10] W. Zhang, H. Mei, H. Zhao, and J. Yang, "Transformation from CIM to PIM: a feature-oriented component-based approach," presented at MoDELS 2005, Montego Bay, Jamaica, 2005.

[11] B. Grammel, and S. Kastenholz, "A generic traceability framework for facet-based traceability data extraction in model-driven software development," presented at the 6th ECMFA Traceability Workshop held in conjunction ECMFA 2010, Paris, France, 2010, pp. 7–14.

[12] OMG, "MOF 2.0 Query/View/Transformation (QVT)", V1.0. OMG Document – formal/08-04-03, 2008.

[13] J. J. Gutiérrez, C. Nebut, M. J. Escalona, M. Mejías, and I. M. Ramos, "Visualization of use cases through automatically generated activity diagrams," presented at 11th international conference on Model Driven Engineering Languages and Systems, France, 2008, pp. 83-96 .

[14] J. Mazón, J. Pardillo, and J. Trujillo, "A model-driven goal-oriented requirement engineering approach for data warehouses," presented at the Conference on Advances in Conceptual Modeling: Foundations and Applications, Auckland, New Zealand, 2007, pp. 255–264.

[15] M. Kardoš, and M. Drozdová, "Analytical method of CIM to PIM transformation in model driven architecture (MDA)," : JIOS, VOL. 34, NO. 1, 2010, pp. 89-99.

[16] Y. Rhazali, Y. Hadi and A. Mouloudi, "Disciplined approach for transformation CIM to PIM in MDA," In Proceedings of the 2015 3rd International Conference on Model-Driven Engineering and Software Development (MODELSWARD), Angers, France, pp312-320, 2015. DOI: 10.5220/0005245903120320 .

[17] Y. Rhazali, Y. Hadi and A. Mouloudi, "A methodology for transforming CIM to PIM through UML: From business view to information system view," In Proceedings of the 2015 Third World Conference on Complex Systems (WCCS), Marrakech, Morocco, pp1-6, 2015. doi: 10.1109/ICoCS.2015.7483318 .

[18] Y. Rhazali, Y. Hadi and A. Mouloudi, "Transformation approach CIM to PIM: from business processes models to state machine and package models," In Proceedings of the 2015 International Conference on Open Source Software Computing (OSSCOM), Amman, Jordan, pp1-6, 2015. doi: 10.1109/OSSCOM.2015.7372686 .

[19] Y. Rhazali, Y. Hadi and A. Mouloudi, "Model Transformation with ATL into MDA from CIM to PIM Structured through MVC, " Procedia Computer Science, Vol. 83, pp1096-1101, 2016. doi:10.1016/j.procs.2016.04.229 .

[20] Y. Rhazali, Y. Hadi and A. Mouloudi, "CIM to PIM Transformation in MDA: from Service-Oriented Business Models to Web-Based Design Models, " International Journal of Software Engineering and Its Applications, Vol. 10, no. 4, pp125-142, 2016. DOI: 10.14257/ijseia.2016.10.4.13 .

[21] Y. Rhazali, Y. Hadi and A. Mouloudi, "A Methodology of Model Transformation in MDA: from CIM to PIM," International Review on Computers and Software (IRECOS), vol. 10, no. 12, pp1186-1201, 2016. DOI: 10.15866/irecos.v10i12.8088 .

[22] Y. Rhazali, Y. Hadi and A. Mouloudi, "A New Methodology CIM to PIM Transformation Resulting from an Analytical Survey," In Proceedings of the 2016 4th International Conference on Model-Driven Engineering and Software Development, Rome, Italy. pp266-273, 2016. DOI: 10.5220/0005690102660273 .

[23] Y. Rhazali, Y. Hadi, and A. Mouloudi, "A Based-Rule Method to Transform CIM to PIM into MDA," International Journal of Cloud Applications and Computing, vol. 6, No. 2, pp11-24, 2016. DOI: 10.4018/IJCAC.2016040102 .

[24] Y. Rhazali, Y. Hadi and A. Mouloudi, "A model transformation in MDA from CIM to PIM represented by web models through SoaML and IFML," In Proceedings of the 2016 4th IEEE International Colloquium on Information Science and Technology (CiSt), Tangier, 2016, pp116-121.doi: 10.1109/CIST.2016.7805027 .

[25] Y. Rhazali, Y. Hadi and A. Mouloudi, "*Transformation des modèles depuis CIM vers PIM dans MDA*". Saarbrücken, Germany, Noor Publishing, 2016**.** ISBN-10: 3330851376. Available: https://www.amazon.com/dp/3330851376 .

[26] H.K. Kim, "Modeling of Distributed Systems with SOA & MDA," IAENG International Journal of Computer Science, Vol. 35, no. 4. pp509-515, 2008.

[27] M. Y. Haouam, and D. Meslati, "Towards Automated Traceability Maintenance in Model Driven Engineering," IAENG International Journal of Computer Science, Vol. 43, no. 2, pp147-155, 2016.

[28] H. Sharma, R. Kuvedu-Libla, and A. K. Ramani, "Towards Model Driven Testing of Human Machine Interface Framework for In-vehicle Infotainment Platforms," IAENG International Journal of Computer Science, Vol. 35, no. 2, pp209-216, 2008.