Available online at www.sciencedirect.com

ScienceDirect

Procedia Computer Science 83 (2016) 1096 – 1101

Procedia Computer Science

The 6th International Symposium on Frontiers in Ambient and Mobile Systems (FAMS 2016)

Model Transformation with ATL into MDA from CIM to PIM Structured through MVC

Yassine Rhazali[a,]*, Youssef Hadi[a], Abdelaziz Mouloudi[a]

[a]*MISC Laboratory, Faculty of Science Kenitra, Ibn Tofail University, Kenitra, Morocco*

Abstract

Models transformation is the most important key in Model Driven Architecture (MDA). The first transformation in MDA is Computing Independent Model (CIM) to Platform Independent Model (PIM) transformation, the second is PIM to Platform Specific Model (PSM) transformation. This latter transformation deals the two less abstract levels, PIM and PSM, for that most researchers focused on this kind of transformation. Nevertheless, the top level of abstraction, CIM, and its transformation to PIM is rarely discussed in research topics. Our goal in this paper is to represent an approach that allows mastering transformation from CIM to PIM in accordance with MDA. Our method based on creating a good CIM level, using construction rules, to facilitate the transformation to PIM level. Next, our transformation rules, implemented with Atlas Transformation Language (ATL), ensure a semi-automatic transformation from CIM to PIM. Our approach conforms to MDA, because it allows considering the business dimension in the CIM level, and it allows modelling this latter level by using Business Process Model and Notation (BPMN), this latter is the OMG standard for modelling the business process. However, we founded on UML to model PIM level, because UML is advocated by MDA in PIM. Our proposal results a set of classes, organized according to the Model View Controller (MVC).

Peer-review under responsibility of the Conference Program Chairs

Keywords: Model transformation; MDA; CIM; PIM; PSM; ATL; Business model; BPMN; MVC.

1. Introduction

The first element to do when building a new application is to specify the enterprise business processes CIM model. The goal of business process models in MDA[1] is to be the first sustainable models. Once the modeled tasks,

* Corresponding author. Tel.: +212-666-525-892.
E-mail address: dr.yassine.rhazali@gmail.com

Peer-review under responsibility of the Conference Program Chairs
doi:10.1016/j.procs.2016.04.229

they are expected to provide a contractual basis, because validated by the application client. Through traceability links with the other models, a link can be created from business processes to the Analysis and design.

Analysis and design are the existing modeling stages from a long time; the design is the step of structuring the software to modules and sub-modules. In PIM level, we view only the abstract design, one that is done without knowledge of implementation techniques. The objective of analysis and design models, PIM models, is to be sustainable and to provide the link between business models, CIM models, and code models (PSM models).The principal difference between a code model and a design model resides in the fact that the code model, PSM model, is related to an execution platform.

MDA emphasizes the importance of model transformations, however, does not propose any methodological transformation process. Most searchers based on the transformation from PIM to PSM, because there are several links between these two levels. Nevertheless, the transformation from CIM to PIM is rarely discussed in research topics because they are two dissimilar levels. For that, in this research we propose an approach that allows mastering transformation from CIM business models to PIM design model structured by MVC (model view controller) design pattern.

The remainder of this paper is presented as follows: In section 2, we represent the related works concerning transformation from the CIM level to the PIM level. In section 3 we show our proposal by describing the construction rules of the CIM models and the transformation rules that allow moving from CIM level to PIM level. An illustration of our approach in a case study is represented at section 4. Finally, we conclude by specifying the future work in section 5.

2. Related Work

In this section, we describe related works concerning transformation approaches from CIM level to PIM level into MDA.

Kardoš et al.[2] represent an analytical method to move from CIM level to PIM level. The authors found on the Data Flow Diagram (DFD)[3] to represent business process in CIM models. Nevertheless, PIM models founded on four UML diagrams includes use case diagram, activity diagram, domain model, and sequence diagram.

Rodríguez et al.[4] propose a transformation approach from secure business process to use cases. This approach presents business process in CIM level through secure business process model which founded in BPMN notation[7]. Nevertheless, a set of transformation rules, checklists, and refinement rules allows transforming secure business process model to use case model that represent PIM level. In[6] Rodríguez et al, founded on the same previous methodology but UML 2 activity diagram used with BPMN notation to model the secure business process, and PIM level enlarged by the class diagram model.

A transformation methodology from CIM to PIM was proposed by De Castro et al.[8].The authors use BPMN for modeling the business process, and value model for identifying services in CIM level. However, PIM models are defined by two extensions of UML activity diagram and two extensions of UML use case diagram.

A methodology for transforming model-driven goal-oriented requirement to data warehouse is shown by mazón et al.[9]. The authors represent CIM level with UML profile using the i* modeling framework. However, some rules used to move from CIM level to s PIM level which is represented by data warehouse design.

An approach where Activity diagram model generated automatically from use case model is represented by Gutiérrez et al.[10]. The authors propose an approach for generate a transformation from system requirements model represented by use case diagram, into CIM level, to activity diagram model which interprets PIM level.

A method for establishing CIM models and transforming them to PIM models according MDA is presented by Wu et al.[11]. The author based on use case diagram, robustness diagram, and activity diagram for modeling user requirements in CIM level. Nevertheless, PIM models are represented by UML 2 class diagram and sequence diagram.

Rhazali et al.[12,13], show a transformation approaches from CIM level to PIM level. These methodologies present CIM level through business process models by using UML activity diagram and BPMN notation. The PIM models are presented by state diagram, class and package diagram. These approaches are based on a set of transformation rules that allows moving from CIM to PIM.

Rhazali et al.[14] present a methodology for transforming CIM to PIM. This method is based on activity diagram to model business process in CIM. The PIM models are presented by class diagram, state diagram, and package diagram.

3. Our Proposal

Computation Independent Model (CIM) means that this model does not contains any information about the software system. According to OMG[1], Computation Independent Model is presented by business process models that are supposed to be representative of the real world. According to OMG[7], BPMN is the standard for modeling the business process, and all the benefits of standards of business process modeling converge to BPMN. In our approach, we use BPMN notation to model business process in the CIM level. We use a set of selected rules to create business models which will contain rich information allow facilitating the transformation from CIM to PIM.

We divided PIM models according to the three classical modeling views[19] including static, dynamic, and functional view. Many researches indicate that UML is advocated by MDA in the PIM level like in[20]. Nevertheless, the intersection of modeling views and our UML models is presented as follows: The model of state diagram interprets the dynamic view, the models of class and package diagrams show the static view, and the model of the use case presents functional view. Finally, we structured all classes, of the information system, according the model view controller (MVC) design pattern. The transformation from CIM to PIM is ensured by well defined rules implemented through ATL.

3.1. Construction rules of CIM level

The rules of construction the BPMN collaboration diagram model:

- Establish a medium sub-processes, in fact, each sub-process must be contains between 4 and 10 tasks.
- If a sub-process includes less than four tasks, or represents an additional operation to another sub-process, we can regroup several sub-processes into one, provided that, the sub-process does not exceed teen tasks.
- Avoid show the tasks and represent only manual tasks.
- Show the general sequence of the business processes and does not represent all possible cases.
- Show only the sub-processes and their relations.
- Represent the maximum of the actors that collaborate in the establishment of enterprise business processes.

The rules of construction the BPMN business process diagram model:

- Detailing each sub-process as various tasks.
- Avoid the representation of manual tasks.
- Show gateways in this model.
- Show the exceptional paths.
- At the output of each task add a data object with its state.

3.2. Transformation rules from CIM to PIM

Fig. 1 represents the transformation rules from BPMN models towards use case model. Fig. 2 shows the transformation rules from BPMN models towards state model. Fig. 3 represents the transformation rules from BPMN models to class model. Fig. 4 shows the transformation rules from BPMN models towards package model. Fig. 5 represents the transformation rules from class model to class model structured through MVC.

4. Case Study

In this section, we present a case study of booking services to illustrate our transformation approach from the CIM level to the PIM level.

Transformation rules in human language	Transformation rules in ATL
R1: "gateway xor" between two "tasks" corresponds to relationship "extend" between two "use cases".	`rule R1 { from gwo : MMbpmn!Or (gwo.isTransformableGatewayOr()) to extd : MMusecase!Extend (name <-gwo.name ) }`
R2: Each "sub-process" is transformed to a "package".	`rule R2 { from sbp : MMbpmn!Subprocess (sbp.isTransformableSubprocess()) to clf : MMusecase!Classifier ( name <- sbp.name, containsActor <- sbp.belongsLane ) }`
R3: every "task" corresponds to a system functionality is transformed to a "use case".	`rule R3 { from ts : MMbpmn!Task ((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.belongsSubprocess )}`

Transformation rules in human language	Transformation rules in ATL
R4: each "collaborator" becomes an "actor"	`rule R4 { from ln : MMbpmn!Lane (ln.isTransformableLane()) to act : MMusecase!Actor ( name <- ln.name ) }`
R5: "sequence flow" between two "tasks" corresponds to relationship "include" between two "use cases". **R6:** "sequence flow" returning back is not transformable.	`rule R5R6 { from sqc : MMbpmn!SequenceFlow ((not sqc.isReturnBack()) and sqc.isTransformableSequenceFlow() ) to icld : MMusecase!Include (name <- sqc.name ) }`

Fig. 1. Transformation rules from BPMN models to use case diagram model.

Transformation rules in human language	Transformation rules in ATL
R7: Each data object becomes a state.	`rule R7 { from dto : MMbpmn!DataObject (dto.isTransformableDataObject()) to nrstt: MMstatemachine!NormalState ( name <- dto.name +' '+ dto.state ) }`
R8: Each exclusive fork becomes a decision point.	`rule R8 { from exl : MMbpmn!ForkExclusive (exl.isTransformableForkExclusive()) to dcs : MMstatemachine!DecisionState (name <- exl.name ) }`
R9: Each exclusive join becomes a junction point.	`rule R9 { from exlj : MMbpmn!JoinExclusive (exlj.isTransformableJoinExclusive()) to jct : MMstatemachine!Jonction (name <- exlj.name )}`
R10: Each parallel fork node becomes a fork state.	`rule R10 { from frkp : MMbpmn!ForkParallel (frkp.isTransformableForkParallel()) to frks : MMstatemachine!ForkState (name <- frkp.name )}`
R11: Each parallel join becomes a joint state.	`rule R11 { from jnp : MMbpmn!JoinParallel (jnp.isTransformableJoinParallel()) to jns : MMstatemachine!JointState ( name <- jnp.name )}`

Transformation rules in human language	Transformation rules in ATL
R12: Each parallel joint and fork becomes a joint and fork state.	`rule R12 { from prll : MMbpmn!parallel(prll.isTransformableparallel()) to frkjns : MMstatemachine!ForkandJointState (name <-prll.name )}`
R13: Each exclusive fork and join becomes a junction point.	`rule R13 { from exlsv : MMbpmn!Exclusive(exlsv.isTransformableExclusive()) to jct : MMstatemachine!Jonction (name <- exlsv.name )}`
R14: Each start event is transformed to an initial state.	`rule R14 { from strtevnt : MMbpmn!Start(strtevnt.isTransformableStartEvent()) to intstt: MMstatemachine!InitialState ( name <- strtevnt.name )}`
R15: Each end event becomes a final state.	`rule R15 { from ende : MMbpmn!End (ende.isTransformableEndEvent()) to fnls : MMstatemachine!FinalState (name <- ende.name)}`
R16: Each sequence flow becomes a transition.	`rule R16 { from sqcf : MMbpmn!SequenceFlow (sqcf.isTransformableSequenceFlow()) to trst : MMstatemachine!Transition ( name <- sqcf.name, source <- sqcf.source(), target <- sqcf.target() )}`

Fig. 2. Transformation rules from BPMN model to state diagram model.

Transformation rules in human language	Transformation rules in ATL
R17: each "data object" is transformed to a "class".	`rule R17 { from dtobj : MMBPMN!DataObject (dtobj.isTransformableDataObject()) to cls : MMclass!Class ( name <- dtobj.name, operations <- dtobj.stateobject.name )}`

Transformation rules in human language	Transformation rules in ATL
R18: Each "state" of a "data object" becomes a "class method".	`rule R18 { from stt : MMBPMN!StateObject (stt.isTransformableStateObject()) to opr : MMclass!Operation ( name <- stt.name ) }`

Fig. 3. Transformation rules from BPMN model to class diagram model.

4.1. Presentation of the CIM level

The model of business process represented by BPMN collaboration diagram is illustrated in Fig. 6.

The second model in the CIM level is represented as a BPMN business process diagram model. However, the sub-process "choose room for reservation" is detailed Fig. 6.

4.2. Presentation of the PIM level

The Fig. 6 shows the state diagram model. The Fig. 7 shows the use case diagram model, the class diagram model, package diagram model and class diagram model structured by MVC.

Transformation rules in human language	Transformation rules in ATL
R19: Each "group" becomes a "package".	`rule R19 { from grp : MMbpmn!Group (grp.isTransformableGroup()) to pck : MMpackage!Package ( name <- grp.name ) }`
R20: Each "sub-process" that does not belong to any "group" transforms to a "package".	`rule R20 { from subp : MMbpmn!Subprocess ((not subp.subprocessBelongsGroup()) and subp.isTransformableSubprocess() ) to pck : MMpackage!Package ( name <- subp.name ) }`

Transformation rules in human language	Transformation rules in ATL
R21: Each set of "classes", become from the same "group", will be placing in the "package" that matches the "group".	`rule R21 { from dtobj : MMbpmn!DataObject (dtobj.dataBelongsGroup() and dtobj.isTransformableDataobject()) to cls :MMpackage!Class ( name <- dtobj.name, operations <- dtobj.stateobject, belongsPackage <- dtobj.belongsGroup ) }`
R22: "Classes", resulting from the same "sub-process" which belongs to no "group", will be placing in the package that corresponds to the sub-processes.	`rule R22 { from dtobj : MMbpmn!DataObject ((not dtobj.dataBelongsGroup()) and dtobj.isTransformableDataobject()) to cls :MMpackage!Class ( name <- dtobj.name, operations <- dtobj.stateobject, belongsPackage <- dtobj.belongsSubprocess ) }`

Fig. 4. Transformation rules from BPMN model to package diagram model.

Transformation rules in human language	Transformation rules in ATL
R23: Each "attribute" is transformed to a "data" into model in MVC.	`rule R23 { from att : MMclass!Attribute (att.isTransformableAttribute()) to dt : MMMVC!Data ( name <- att.name, belongsModel <- att.belongsClass ) }`
R24: Each "method" is transformed to an "operation" into controller in MVC.	`rule R24 { from mtd : MMclass!Method (mtd.isTransformableMethod()) to oprt : MMMVC!Operation ( name <- mtd.name, belongscontroller <- mtd.belongsClass ) }`
R25: Each "attribute" is transformed to "text field" into view in MVC.	`rule R25 { from mtdd : MMclass!Attribute (mtd.isTransformableAttribute()) to txtfl : MMMVC!Textfields ( name <- mtdd.name, belongsView <- mtdd.belongsClass) }`

Transformation rules in human language	Transformation rules in ATL
R26: each "method" is transformed to an "action" into view in MVC.	`rule R26 { from attr : MMclass!Attribute (mtd.isTransformableMethod()) to ctrl : MMMVC!Action ( name <- attr.name, belongsView <- att.belongsClass ) }`
R27: each "class" is transformed to a "model", "view" and "controller" in MVC.	`rule R26 { from cls : MMclass!Class (cls.isTransformableMethod()) to ctrl : MMMVC!Model ( name <- 'Model'+cls.name ) Vw : MMMVC!View ( name <- 'View'+cls.name ) Ctl : MMMVC!Controller ( 'Controller'+name <- cls.name ) }`

Fig. 5. Transformation rules from class model to class model structured through MVC.

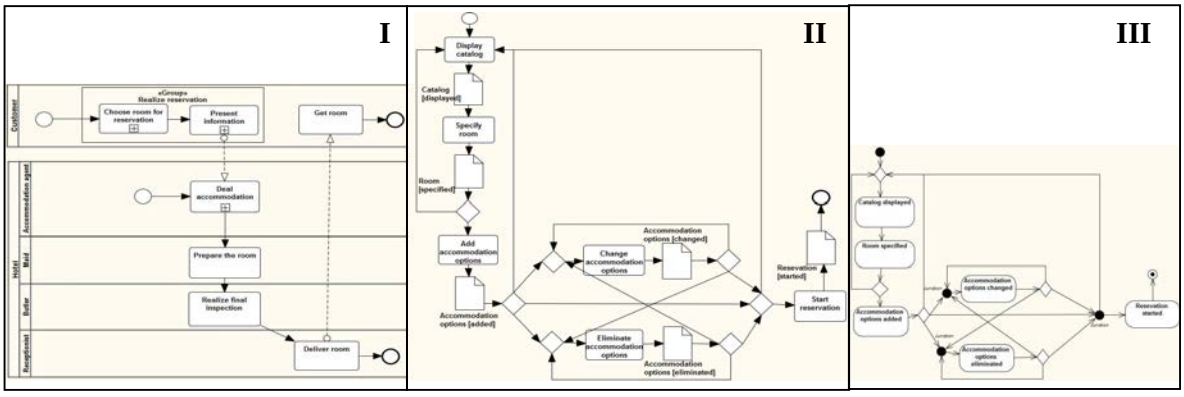

Fig. 6. I: BPMN collaboration diagram model, II: BPMN business process diagram model, III: state diagram model.

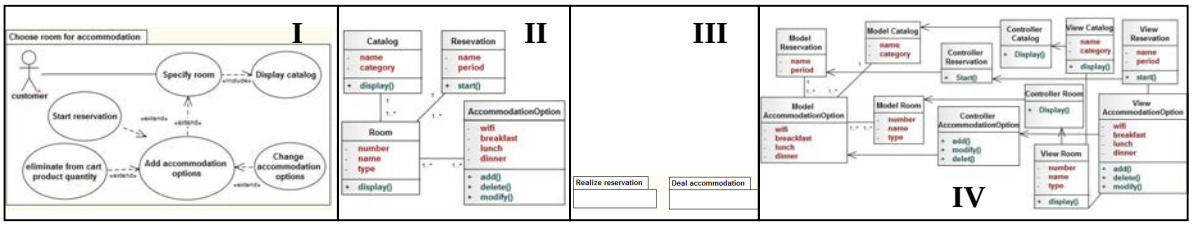

Fig. 7. I: use case diagram model, II class diagram model, III: package diagram model, IV: class diagram model structured by MVC

5. Conclusion and Future Work

One of the major challenges in the software development process is the definition of an approach that allows moving from models that describe the working of the business to models which present the analysis and design of software.

Based on MDA, our approach provides an efficient solution to the problem of transformation of business models represented in CIM level to analysis and design models, modeled in PIM level. In this approach, we benefited from our experience in old methods[5,15,16,17,18,21] to provide, through semi-automatic transformation implemented by ATL, a set of classes structured according to MVC.

The future work intended to improve the rules of construction of the CIM level and the rules of transformation to the PIM, In order to implement these transformations into a tool via the ATL language.

References

1. OMG-MDA. *MDA Guide revision 2.0*. OMG; 2015.
2. 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.
3. Qing Li,Yu-Liu Chen, Modeling and Analysis of Enterprise and Information Systems. Springer Berlin Heidelberg, 2009.
4. Rodríguez, A., Fernández-Medina E., Piattini, M., *Towards CIM to PIM transformation: from Secure Business Processes defined in BPMN to Use-Cases*, *Business Process Management*, Proceedings of the 5th International Conference on Business Process Management (Page: 408 Year of Publication: 2007 ISBN: 3-540-75182-3 978-3-540-75182-3).
5. 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, 2016.
6. 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, 2010, pp. 945–971.
7. OMG, Business Process Model and Notation (BPMN)-Version 2.0 (OMG, 2011).
8. 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 Technology, 2011, pp. 87–105.
9. 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.
10. 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 .
11. Wu, J. H., Shin, S. S., Chien, J. L., Chao, W. S., Hsieh, M. C., *An Extended MDA Method for User Interface Modeling and Transformation*, Proceedings of the 15th European Conference on Information Systems (Page: 1632 Year of Publication: 2007).
12. Y. Rhazali, Y. Hadi, and A. Mouloudi, "Disciplined Approach for Transformation CIM to PIM in MDA," presented at 3rd International Conference on Model-Driven Engineering and Software Development, Angers, France, 2015, pp. 312 - 320.
13. Y. Rhazali, Y. Hadi, and A. Mouloudi, " Transformation Approach CIM to PIM: From Business Processes Models to State Machine and Package Models," presented at the 1st International Conference on Open Source Software Computing, Amman, Jordan, 2015, pp. 1 - 6.
14. Y. Rhazali, Y. Hadi, and A. Mouloudi, "Methodology for Transforming CIM to PIM through UML: From Business View to Information System View," presented at the IEEE Third World Conference on Complex Systems, Marakech, Morocco, 2015.
15. Y. Rhazali, Y. Hadi, A. Mouloudi, Transformation Method CIM to PIM: From Business Processes Models Defined in BPMN to Use Case and Class Models Defined in UML, *International Journal of Computer, Electrical, Automation, Control and Information Engineering*, Vol. 8, n. 8, pp. 1453–1457, 2014.
16. Y. Rhazali, Y. Hadi, and A. Mouloudi, A Methodology of Model Transformation in MDA: from CIM to PIM, International Review on Computers and Software, Vol. 10, No. 12, pp. 1186-1201, 2015.
17. 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, 2016.
18. Y. Rhazali, Y. Hadi, and A. Mouloudi, Disciplined Approach for Transformation CIM to PIM in MDA, Proceedings of the 4rd International Conference on Model-Driven Engineering and Software Development (Page: 266 Year of Publication: 2016 ISBN: 978-989-758-168-7).
19. P. Roques, UML in Practice: The Art of Modeling Software Systems Demonstrated through Worked Examples and Solutions (Wiley, 2004).
20. A. G. Kleppe, J. Warmer, W. Bast, MDA Explained: The Model Driven Architecture: Practice and Promise (Addison-Wesley, 2003).
21. Y. Rhazali, Y. Hadi, and A. Mouloudi, Model Transformation from CIM to PIM in MDA: from Business Models defined in DFD to Design Models defined in UML, la revue des technologies de l'information, No. 9, 2016.