Transformation of the CIM Model into A PIM Model According to The MDA Approach for Application Interoperability: Case of the "COVID-19 Patient Management" Business Process

https://doi.org/10.3991/ijoe.v17i05.21419

Meryem Fakhouri Amr (✉), Nezha Benmoussa, Khalifa Mansouri, Mohammed Qbadou
University Hassan II, Casablanca, Morocco
meryemfakhr@gmail.com

Abstract—Model Driven Engineering (MDE) plays a very important role in improving the development of complex systems. It focuses more on modeling than on classical programming. In this sense, model transformation is at the heart of the Model Driven Architecture (MDA) approach which advocates the use of models throughout the software development cycle on two levels. The first being the transformation of Computing Independent Model (CIM) into Platform Independent Model (PIM) and the second concerning the transformation of PIM into Platform Specific Model (PSM).

The latter has been dealt with in the majority of research works while the transformation from CIM to PIM which represents the highest level is rarely discussed in research topics.

It is for this reason and in the spirit of improving the process of transforming the CIM model into PIM according to the MDA approach, that we have developed this research work in order to propose a new method and new transformation rules for optimization of the business process "COVID-19 patient management". Our contribution consists of the semi-automatic transformation of the CIM model presented by the BPMN (Business Process Model and Notation) source model into a PIM target model presented by a class diagram by using a set of transformation and correspondence rules that we performed manually using the language ATL (Atlas Transformation Language). This automatic transformation of the two source and target models is provided by the Eclipse Modeling Framework (EMF) which executes the transformation rules described manually in ATL and generates the PIM target model as an XMI (XML Metadata Interchange) file representing the target class diagram.

Keywords—CIM, PIM, BPMN, ATL, Class Diagram, XMI, COVID-19, MDA, MDE

1 Introduction

COVID-19 is a global crisis that has particularly disrupted our lives and science since December 2019 [1]. It is a great challenge facing, so far, both individuals and scientific researchers. Since the advent of this virus, they have focused their research and efforts on studies, analyzes and tests to propose solutions or new procedures in all fields. For our part, we have thought about implementing a model based on the MDA approach whose principle is the separation of business logic from that of implementation on a given platform, thus guaranteeing interoperability of applications. It proposes a four-level architecture with the following elements at each level: meta-metamodel, metamodel, model and information. [2]

MDA promotes the use of models in the different software development cycles since models are more durable than codes. It is an approach that recommends the development of analysis models first, then from design to code by successive transformations and enrichments. [3]

On this basis, we proposed a semi-automatic transformation of the CIM model into a PIM model. We used BPMN notation to present the CIM model and the class diagram to present the PIM model. The main input of this hybrid transformation is the BPMN source model that generates the output class diagram. First, it is manual by defining a set of transformation rules and automatic by using the EMF Framework (Eclipse Modeling Framework) which will execute these rules to obtain the target model which is the "COVID-19 patient management" class diagram. For more consistency, we have based on MDE Model Driven Engineering to cover the different phases of our model development.

2 Techniques Used

The software industry and academic research have been the origin of IDM from 2001 until now. Currently, the IDM is the subject of various studies in various fields because it allows to generate productive models according to code-centered approaches. [4].

2.1 Model Driven Engineering MDE

Model Driven Engineering (MDE) is a software engineering approach that involves defining a theoretical framework for generating code based on a set of successive model transformations. [5]

The MDE makes it possible to cover the different phases of the development cycle by adopting an approach based on the transformation of models [6]. This approach ensures the consistency of the system during the different phases of the life cycle. [7]

2.2 Fundamental concepts of MDE

MDE aims to cover the entire application life cycle from design to maintenance or development of the application. To do this, it brings together a set of pre-existing techniques, including reverse engineering, code generation or even transformations, around a paradigm: everything is model. [4]

The MDE is based on an architecture that relies on the definition of a four-level meta model.

The following figure represents the different levels of abstraction and their denomination. The first M0 level does not belong to the modeling world but to the real world. Level M1 corresponds to the first level of abstraction: models. The M2 and M3 levels respectively denote the set of metamodels and the meta-metamodel. [8]

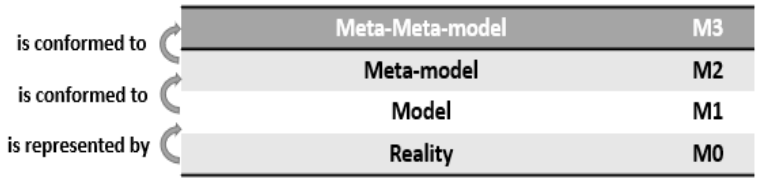

Fig. 1. The four levels of modeling in model-driven engineering

3 Model Driven Architecture

3.1 MDA principle

MDA is a software development approach offered by the OMG (Object Management Group). It is considered a variant of MDE for software engineering. Its main goal is to improve software design practices towards a model-centric approach. [9]

Model-Driven Architecture or MDA is based on the principle of separating business logic from implementation logic. Its goal is to build platform independent models and transform them into platform-dependent models. [9]

MDA is based on a typology of models and a set of transformation relations to move from one model to another. This architecture defines four main types of models as follows:

- CIM: (Computation Independent Model): It is also called a domain model or business model. It describes the situation in which the system will be used. This model represents the highest level of abstraction and describes the system requirements and the environment in which it will operate. [10]
- PIM: (Platform Independent Model): PIM describes the system regardless of the target platform on which it will run [11]. It should describe two aspects of a system: structure and behavior. The structure aspect focuses on the static structure

of the system using classes, objects, attribute of relations, while the behavior aspect focuses on the dynamic behavior of the system by showing the interactions between objects. [10]

- PDM: (Platform Description Model): PDM is the model that describes the execution platform. It provides a set of technical concepts representing the different parts of the platform. [10]
- PSM (Platform Specific Model): The PSM model is the more refined model of the MDA, it is used to generate the code that will be executed on a specific platform. The notion of PSM is always associated with the notion of code. The PSM is not intended to be sustainable. Its main function is to facilitate code generation. [12]

4 Related Works

The transformation models under the MDA approach have been studied extensively in different fields. Several research works have addressed the problem of transforming from PIM to PSM and from PSM to code, but very few deals with the transformation from CIM to PIM.

In this section, we describe some related work concerning transformation approaches from CIM level to PIM level in MDA. Among these research works, [13] uses BPMN notation to model business processes at the CIM level and a set of rules to create models that will contain information to facilitate the transformation of CIM to PIM. It divides the PIM models into the three classic modeling views, static view, dynamic view, and functional view. [14] presents a semi-automation approach of the entire software process based on a set of meta-models and model transformations.

While [14] only focuses on applying a PIM analysis to design the transformation of the PIM model to automate the transition between these phases. [15] proposes a method to transform business process models into information system models. This method is based on BPMN notation to model the business process at CIM level; however, PIM models are presented by class diagram and use case diagram.

Our contribution consists of the hybrid transformation from CIM model to PIM model. In our case, the CIM model is presented by the BPMN business process model which represents the real world. BPMN notation is used to model the business process of "COVID-19 patient management" which will be transformed into a PIM model presented by a class diagram.

The transformation between these two models is done in a semi-automatic way. First, a manual transformation via the development of a set of transformation rules between the BPMN model and the class diagram respecting the logic of the modeling. Then, an automatic transformation using an ATL transformation language that describes the rules described manually before.

The peculiarity of our contribution is the hybridization of both types of manual and automatic transformation and the transformation of models using a concrete case study represented by the process of "COVID-19 patient management". We started by modeling the business process “COVID-19 patient management” using the BPMN notation recommended by the MDA approach. This BPMN model will be the source

model of our transformation. We also developed the meta model of the BPMN source model using the EMF Framework that we named the source meta model **MM1**. In the same way we created the target metamodel of the target class diagram named target metamodel **MM2**. Then, we created an instance of the BPMN source model which in our case represents the business process "COVID-19 patient management".

We recall that the most important part, in our contribution, is the development of correspondence rules defined and implemented via the ATL language which will be executed to generate the target model in the form of an XMI file.

5 Case Study: Business Process "COVID-19 Patient Management"

5.1 General context: Problem

Considering the circumstances and conditions of the COVID-19 virus that we are experiencing today, we tried to apply our case study on a concrete example in order to propose a solution that allows to detect people tested positive as well as contact cases. Our main objective is to implement a transformation of concrete models but on real cases which will guarantee better interoperability of hospital information systems [16]. The application of this business process and the adoption of this platform by Moroccan health institutions will have a commendable impact on Moroccan hospital information systems SIH. [17] This solution will allow:

- **Improving the quality of care**
  - **Improvement of communications:** Rapid patient care via teleconsultation
  - **Reduction of waiting times:** The patient is quickly served by a medical advisor
  - **Integration of the patient file:** Patient information is centralized on this platform
  - **Decision support** thanks to data traceability
  - **Digitalization of medical services**
- **Cost control**
  - **Reducing the length of stays**
  - **Reduction of administrative tasks**
  - **Decrease in personnel costs**
  - **Resource optimization**
  - **Accessibility to data and information** from all types of agents according to their profiles (patient, doctor, administrative agent, ministry, etc.)

We present our case study through a process that we have titled "COVID-19 patient management" which consists of identifying and caring for people who test positive and initiating the tracing of contact cases. This process can be presented by a

platform dedicated to this objective which will be led by doctors from health establishments.

Anyone who doubts being positive and wants to be insured can contact the health facility to be supported by a doctor or a health operator, register it on the platform by creating a patient file, and give it instructions (tests, isolation, mask, etc.).

The objective of this case study is to concretize the transformation between the models with a concrete case, the registration of a patient who can be diagnosed positive and the initiation of a patient file linked to this patient on the one hand. On the other hand, to initiate and enrich the platform with cases tested positive as well as to detect contact cases. This is how this platform can access subsequently, in the event of centralization, by the biologists to the information necessary for carrying out the sample. [18]

Through this case study, we want to present a concrete business process model in the form of a BPMN model and transform it into a class diagram model by applying ATL transformation rules according to the MDA approach. We will base ourselves during the implementation of this transformation on the MDA approach.

5.2 BPMN business process "COVID-19 patient management"

This business process represents the process we proposed for the identification of patients testing positive by COVID-19. It is initiated by the reception in consultation on site or teleconsultation of a patient presenting symptoms. A doctor or a health operator receives the patient and creates a new patient file for him if he is not already registered in the platform, otherwise he analyzes his request. Then, the health operator sends the form to the patient to keep it and depending on the symptoms he presents, he prescribes an RT-PCR (Reverse Transcriptase Polymerase Chain Reaction) test. The patient must take the RT-PCR test in a laboratory which must return the results to him within 48 hours. The figure below shows the business process that we have just described and on which we will use to present the transformation between the BPMN model and the class diagram.

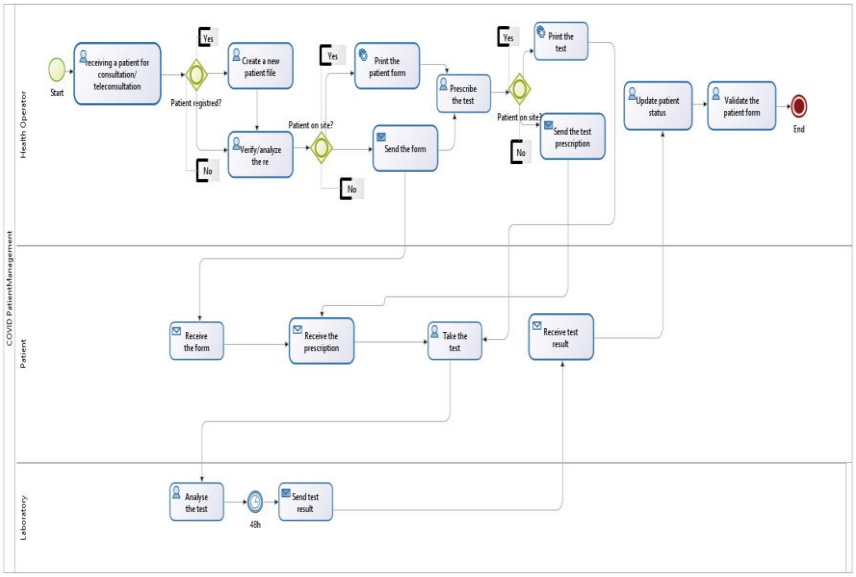

Fig. 2. Business process "COVID-19 patient management"

5.3 Correspondence between BPMN model and class diagram

Our contribution consists in performing a semi-automatic transformation between the BPMN model and the class diagram. The BPMN model that we are going to model and transform is based on the "COVID-19 patient management" business process explained previously. It is shown in the following figure and is accompanied by its corresponding class diagram. It will be the source model of our transformation and the class diagram will be the target model.

In this figure, we present the elements that will be transformed from the BPMN model to the class diagram: each element of the BPMN model with the corresponding element in the class diagram. This correspondence is not exhaustive because the class diagram represents just the part that interests us of our business process "COVID-19 patient management" in order not to duplicate modeling and transformation.

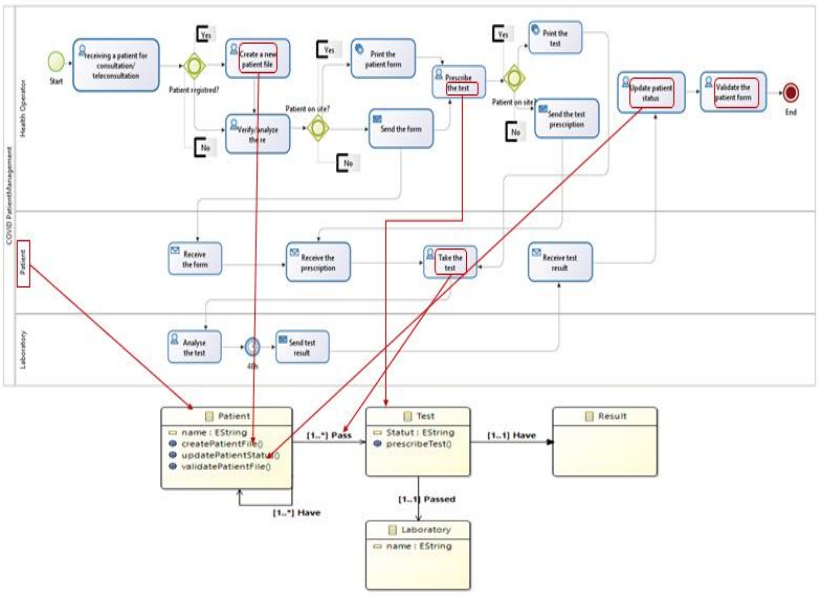

Fig. 3. Diagram of transition from BPMN model to class diagram

This figure shows the first part of the transformations we have made. We first tried to dissect the BPMN model and bring out all of these elements that compose it. Subsequently, we made a correspondence list between each component of the BPMN model and its correspondent in the class diagram. The class diagram used represents part of the overall class diagram of the "COVID-19 patient management" business process.

It is not easy to transform all the components to have a total resemblance but, in our case, we have followed a logical methodology to keep this consistency between the two models and to have consistency between the possible transformations between the two models (BPMN and class diagram). All of these transformations will be presented in the next section.

5.4 Logic for transforming the BPMN model into a class diagram

The following table shows an example of a manual transformation of the BPMN model, which is made up of a set of components each fulfilling a specific mission and role. We will not define and present the components of the BPMN model but we will present each component and the transformation it undergoes. To transform the BPMN model, we have classified its components according to universal categories: **Gateway, activity and task, flow sequence, data object** and **lane** [19]. In general, in BPMN a process is described as a graph which is a series of 4 categories of basic elements each defining a finite execution semantics. Each element can be enriched with other symbols to provide flexibility in modeling [20].

For some elements of the BPMN model, it was easy for us to transform them according to a modeling logic towards the class diagram but for others it was necessary to make a decomposition of the element and a succession of transformations to arrive at the best modeling possible. This is the case with the **"task"** element. This element is essential in a BPMN model because it is part of the elements that give meaning to the modeling of the process. A task is an action that a process participant completes in order to achieve the defined goal of the business process.

If we compare the semantics of a task with natural language, the participant of the process will be the subject of the process (Someone who does something). The activity will be the verb and the label of the activity usually contains an object: when the participant of a process performs his action on something. When tasks are labeled, the predefined structure is usually a verb + an object or an object + does + an action as follows:

<Subject> = <Verb> or <Noun> or <Term>

<Expression> = <Action> / <Process>

The rule is: <Subject> do <Expression>

=> <Subject> do <Action> / <Process>

The possible transformations that can be made are:

Transform <**Expression**> which can be <**Action**> or <**Process**> into a class:

<Expression> -> Class

These rules were adopted to make several transformations on our BPMN model, which we will present as an example in the following table:

Table 1. Proposed logic for transforming the BPMN model into a class diagram

BPMN task	Application of the rule	Explanation of the BPMN transformation to a class
Creation Patient File	**<Creation> <Patient>**	Transformation of the term Patient into a Patient class
The patient does the test in a laboratory	**<Patient> <Laboratory>**	The word Laboratory is transformed into class Laboratory
Prescribe the test	**<Prescribe> <Test>**	The word Test is transformed into a class Test
Print the test	**<Print> <Test>**	The word Test is transformed into a class Test
Validate the file	**<Validate> <File>**	The term File is transformed into a File class

This table contains an example of transforming a task from the BPMN model to the class diagram by applying the rule proposed above. Depending on the tasks we have on our BPMN model of "COVID-19 patient management", we can get several possible transformations to the class diagram.

5.5 Discussion: Benefits of transforming models

The choice of model transformation was not a random choice, it is justified by several reasons. First of all, the model driven architecture allows business requirements to be retained, especially models since they are more durable than codes, by separating functional specifications from implementation specifications. Which

will allow for better interoperability between applications. In addition, this architecture makes it possible to reduce the tasks of redesigning applications and reuse architectural choices. [21].

The general principle of MDA is to build models, first of all starting with the analysis models then of design until the code while going through a set of transformations, derivations and successive enrichments. We can say that it is the most reliable way to guarantee the integrity and consistency between the phases of a project.

It is for this reason that we chose to transform models with the MDA approach to guarantee the reuse of our transformation code for future development work as well as to facilitate interoperability between hospital information systems [16] by model transformation that can be reused.

5.6 Development of the source BPMN metamodel (extension)

Our transformation is based on several principles and follows specific steps from the development of the metamodel to the generation of the target model. [22]. We must first start with the modeling of the source and target metamodels at the modeling platform. [23]. This modeling consists of the creation of two metamodels which respectively describe the BPMN model and the class diagram. Note that the basic source metamodel conforms to the BPMN source model. The most important part is the elaboration of the transformation rules. A file must contain all the possible transformations between the two source and target models. In our case, our transformation file contains transformation and mapping rules between the BPMN model and the class diagram. We have tried through this transformation to respect the modeling logic between the two source and target models.

Our case study presents the manual (rules) and automatic transformation of the BPMN model to the class diagram. We used a platform that uses these rules to perform the desired transformation.

First, we started with the development of the source and target meta models. We modeled according to the EMF platform, the source metamodel which describes the structure of the BPMN model in order to propose and customize our meta model according to our conception and perception of the BPMN model. Our contribution can be seen as an extension of the BPMN meta model of other research work. The following figure shows the extension of the meta model source of the BPMN model that we have designed and developed on the EMF platform according to our own perception.

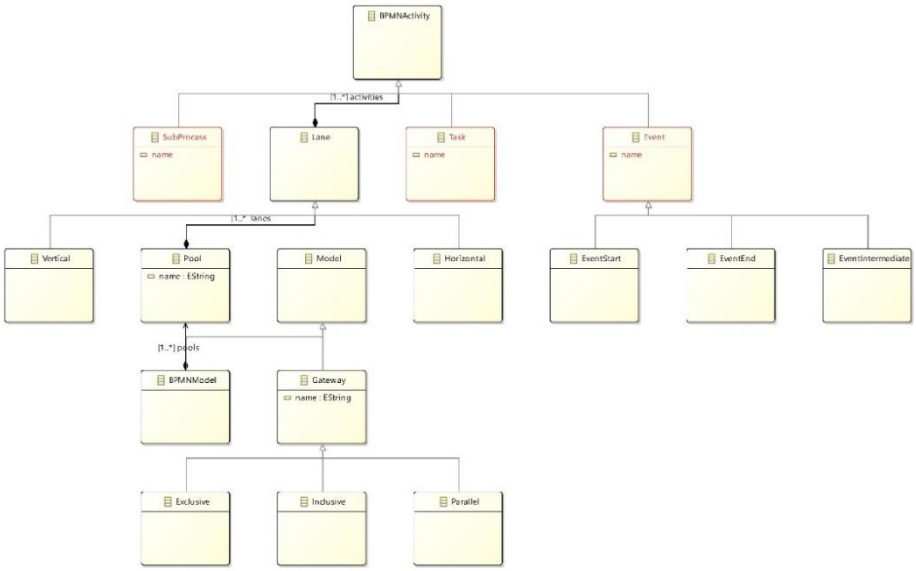

Fig. 4. Extension of the BPMN metamodel

5.7 Development of the metamodel of the target class diagram (extension)

The second target metamodel is the class diagram meta model that we customized too. It reflects our own modeling inspired by other existing metamodels with new options.

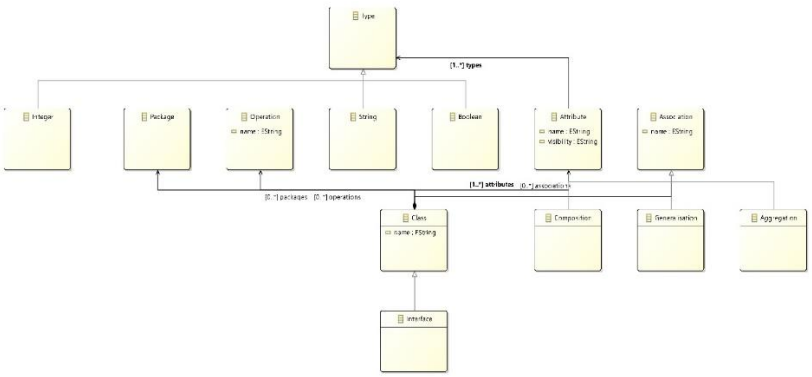

Fig. 5. Extending the metamodel of the class diagram

5.8 Development of the target model (Class diagram)

In this case, our target model is the class diagram. Thanks to a set of transformation rules which will be applied on the target metamodel and based on the source

metamodel, we will obtain our target model which is the class diagram and which presents part of our source BPMN model of " COVID-19 patient management" implementing the following entities:

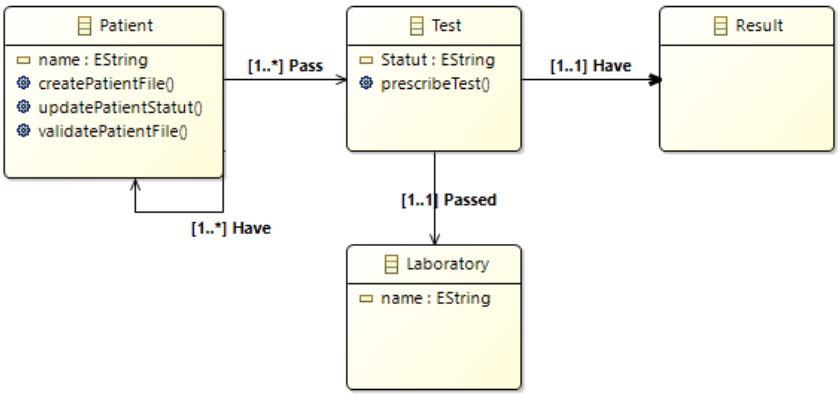

Fig. 6. Example of target class diagram

5.9 Rules for transforming the BPMN model into a class diagram

The table we propose allows us to first perform a manual transformation between the BPMN model and the class diagram and then move on to the automatic transformation via a modeling framework which will allow us to generate the models.

This table presents transformation rules between the BPMN model and the class diagram written in human language and their correspondence in ATL language. First, we have dissected the BPMN model and the functionality of each component in order to deduce exactly what will correspond to each component in a class diagram. Then, we identified the elements in common and those which represent a certain coherence between the two models. Finally, we have listed the possible transformations between the two models with the conditions that must be met to obtain the best transformation.

The table below shows the list of possible transformations between the two BPMN models and class diagram.

Table 2. Proposed rules for transforming the BPMN model into a class diagram

BPMN to class transformation rule	ATL Transformation Rules
Each "**Lane**" which represents an actor involved in the system is transformed into a "class"	rule TransfLane {from Trlane: MM! Lane to Cls: MM1! Class (name ←Trlane.name)}
"**Gateway xor**": between tasks expressing a membership link is transformed into a "composition relation"	rule Gatexor {from Gxor: MM! Exclusive to Compos: MM! Composition (name←Gxor.name)}
"**Gateway Parallel**" between two "tasks" becomes normal "relation / association"	rule Gateparallel {from paral: MM! Parallel (paral.isTransformableParallel ()) to relation: MM1! Association (name←paral.name)}
each "S**equence Flow**" between two tasks corresponds to "relation / association" between two class	rule Seqflow {from seq: MM! Sequenceflow (seq.is TransformableSequenceflow ()) to assoc: MM1! Association (name←seq.name)}
Task: the action in the human task or performed by the system can be transformed into a "class"	rule Taskaction {from Taskact: MM! Task (taskact.isTransformableTask ()) to cl: MM1! Class (name←taskact.name)}
Task: - performed by the system -Manually performed by humans. This task is transformed into a "method"	rule Taskmethod {from Taskmeth: MM! Task (taskmetht.isTransformableTask ()) to Method: MM1! Operation (name←taskmeth.name)}
"**Data Object**" is transformed into "class"	rule Dataobject {from dobject: MM! object (dobject.isTransformableTask ()) to clss: MM1! Class (name←dobject.name)}

6 Implementation: Automatic Transformation of CIM Model into PIM Model

6.1 Implementation diagram

The following figure shows the diagram that we will follow to guarantee the transformation of the CIM model into a PIM model, by the transformation of the BPMN model into a class diagram [24]. In the following figure, a BPMN source model is transformed into a CLASS target model. The transformation is directed by a transformation program MM1ToME2.atl written in ATL. This program is a model. The source and target models as well as the transformation program conform to their respective meta-models: MM1, MM2 and ATL. These meta-models conform to the Ecore meta-meta-model.

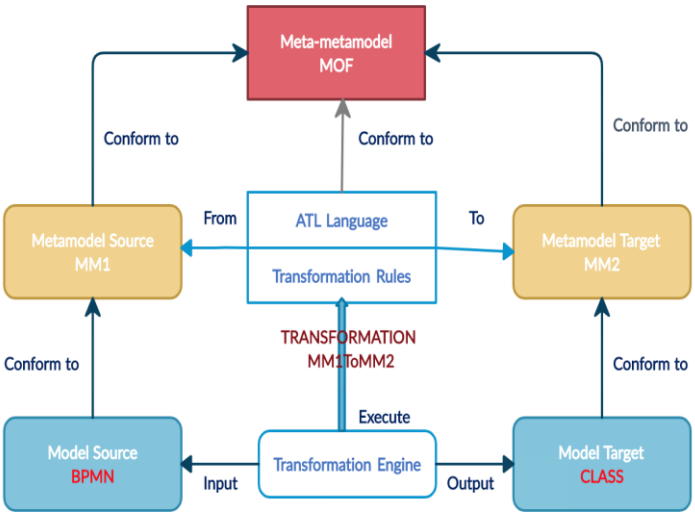

Fig. 7. Model transformation diagram

6.2 Implementation of the transformation

In this part, we describe the techniques used to implement the approach presented. Since we are implementing our case study with a model driven approach, we used a technical environment suitable for modeling, meta-modeling and transformation of models. We used the Eclipse Modeling Framework (EMF) platform which uses Ecore [25] to create the models which is the basis for the creation of an ATL project according to the following structure: Metamodels, Models and Transformations.

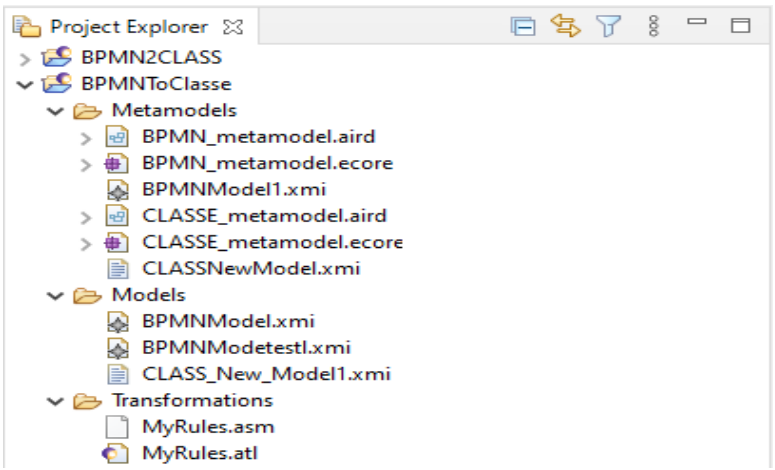

Fig. 8. Structure of an ATL project

An ATL project must follow the structure described in the figure above. Folders which will respectively contain the Ecore metamodels and the models that we want to transform and the transformations on these models. The following figure illustrates the Ecore metamodels used by our transformation module:

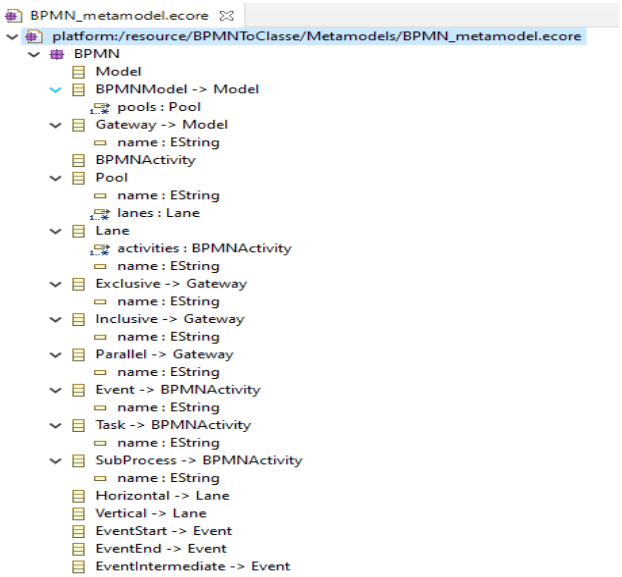

Fig. 9. ECORE meta model of the BPMN model

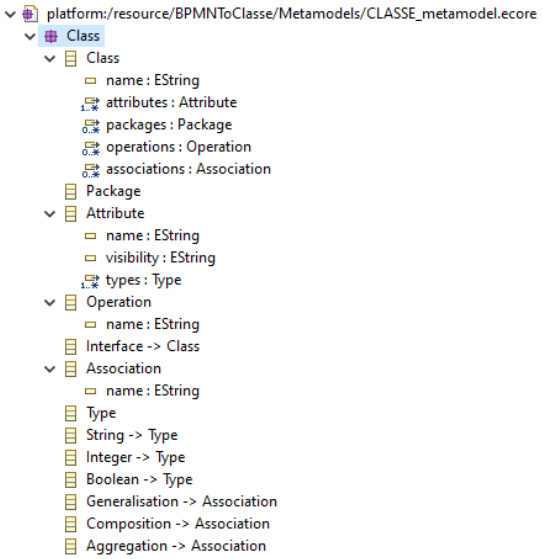

Fig. 10.ECORE Meta Model of the Class Diagram

The choice of transformation language is not random, we based ourselves on specific criteria that align with our approach. Indeed, the Framework used for the implementation of our approach integrates the language that we want to use. Thus, we used the ATL language for the transformation of models. ATL is a model transformation language in the field of MDE. It is a way to specify how to produce a number of target models from source models [26].

The following figure shows an extract of the ATL language used to ensure the model transformation and the generation of the PIM corresponding to the class diagram from the BPMN model.

MyRules.atl

```
-- @path MM=/BPMNToClasse/Metamodels/BPMN_metamodel.ecore
-- @path MM1=/BPMNToClasse/Metamodels/CLASSE_metamodel.ecore
module MyRules;
create OUT : MM1 from IN : MM;
rule BPMN2Class {
    from B : MM!BPMN
    to C : MM1!Class(Operation<-B.Task)}

rule TransfLane {
   from Trlane : MM!Lane (Trlane.isTransformableLane())
   to Cls : MM1!Class (name<- Trlane.name)}

rule Gatexor {
   from Gxor : MM !Exclusive(Gxor.isTransformableExclusive())
   to Compos : MM !Composition (name<-Gxor.name)}

rule Gateparallel{
   from Paral : MM!Parallel (Paral.isTransformableParallel())
   to Relation : MM1!Association(name<-Paral.name)}

rule Seqflow {
   from Seq : MM!Sequenceflow(Seq.isTransformableSequenceflow())
   to Assoc : MM1!Association(name<-Seq.name)}

rule Taskaction{
   from Taskact : MM!Task (Taskact.isTransformableTask())
   to Cl : MM1!Class (name<-Taskact.name)}

rule Taskmethod{
  from Taskmeth : MM!Task (Taskmeth.isTransformableTask())
  to Method : MM1!Operation(name<-Taskmeth.name)}

rule Dataobject{
  from Dobject : MM!object (Dobject.isTransformableTask())
  to Clss : MM1!Class (name<-Dobject.name)}
```

Fig. 11. Model transformation ATL code

The next step is to instantiate the metamodel of a part of the BPMN model "COVID-19 patient management", an example of this instance is shown in this figure.

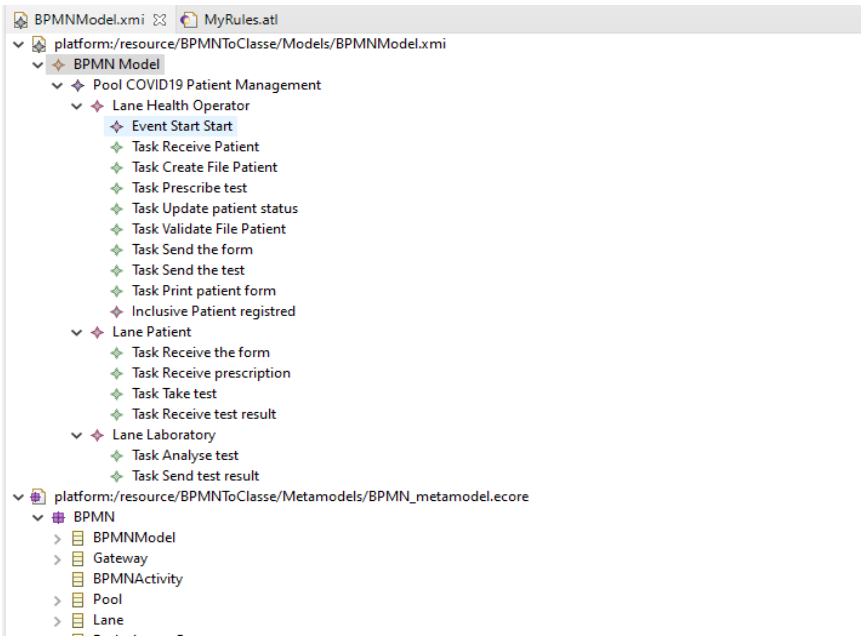

Fig. 12. Instantiation of the BPMN model "COVID-19 patient management"

In this section, we present the file generated after the execution of the transformation rules in ATL. The generation of this file is successful when the transformation rules are correct and logical and present consistency between the source and target model. The generated file presents the structure of a class diagram of a part of the "COVID-19 patient management" process and can be presented in several forms while ensuring the basic structure of the diagram. The generated file is in the form of an XMI file and it conforms to its Ecore meta model. The structure of this file described in XML can be transformed into several forms including the class diagram.

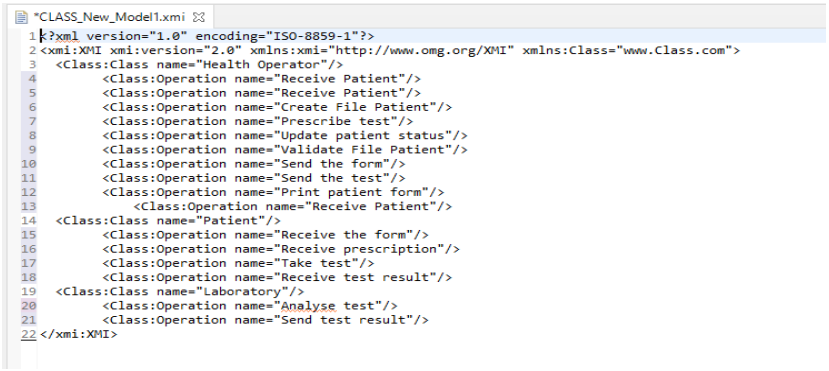

Fig. 13. File generated after the execution of the transformation rules in ATL

7 Conclusion

Through this article, we have succeeded in implementing a transformation from the CIM model to the PIM model according to the MDA approach. This transformation is ensured by a set of correspondence rules defined in ATL language. Our transformation started with the development of the source and target metamodels going through the development of transformation rules in ATL until the execution of these rules to obtain the target model in the form of an XMI file. In our case the source model of the transformation is represented by the BPMN notation and the target model is presented by the class diagram. In this article, we have tried to achieve real transformation based on the real business process of "COVID-19 patient management". As a perspective, we plan to achieve a transformation from the BPMN model to other target models including the use case diagram, state diagram and activity diagram.

8 Refecences

[1] J.Kachaoui, J.Larioui, A.Belangour, "Towards an Ontology Proposal Model in Data Lake for Real-time COVID-19 Cases Prevention," International Journal of Online and Biomedical Engineering (iJOE), Vol. 16, No. 9, 2020, https://doi.org/10.3991/ijoe.v16i09. 15325

[2] F. Bacha, Une approche MDA pour l'intégration de la personnalisation du contenu dans la conception et la génération des applications interactives.Autre [cs.OH], Université de Valenciennes et du Hainaut-Cambresis, 2013. Français. NNT: 2013VALE0016. tel-00874766. https://doi.org/10.3934/dcdss.2009.2.559

[3] A.Elmounadi, N.Berbiche, N.Sefiani, "Architecture dirigée par les modèles : Méthodes et outils de transformation de modèles," Laboratoire d'Analyse des Systèmes, du Traitement de l'Information et du Management Intégré,École Supérieure de Technologie de Salé – Maroc, Novembre 2013. https://doi.org/10.21474/ijar01/11742

[4] G.Barbier, Contribution de l'ingénierie dirigée par les modèles à la conception de modèles grande culture, Autre. Université Blaise Pascal - Clermont-Ferrand II, 2013. Français. ⟨NNT: 2013CLF22357⟩. ⟨tel-00914318⟩.

[5] P.A.Caron, Ingénierie dirigée par les modèles pour la construction de dispositifs pédagogiques sur des plateformes de formation, Autre [cs.OH]. Université des Sciences et Technologie de Lille - Lille I, 2007. Français. tel-00156376. https://doi.org/10.3406/rfp.1986.1504

[6] O.Betari, S.Filali, A.Azzaoui, M.A.Boubnad, "Applying a Model Driven Architecture Approach: Transforming CIM to PIM Using UML," International Journal of Online and Biomedical Engineering (iJOE), Vol. 14, No. 9, 2018, https://doi.org/10.3991/ijoe.v14i09.9137

[7] S.Azaiez, Approche dirigée par les modèles pour le développement de systèmes multi-agents. Génie logiciel [cs.SE], Université de Savoie, 2007. Français. tel-00519195.

[8] S.Baina, H.Panetto, K.Benali, "Apport de l'approche MDA pour une interopérabilité sémantique : Interopérabilité des systèmes d'information d'entreprise," Revue des Sciences et Technologies de l'Information - Série ISI : Ingénierie des Systèmes d'Information, Lavoisier, 2006, 11/13, pp.11-29. hal-00086522. https://doi.org/10.3166/isi.11.3.11-29

[9] C. Hardebolle, Composition de modèles pour la modélisation multi-paradigme du comportement des systèmes, Université Paris-Sud XI Orsay, Novembre 2008.

[10] B.Bousetta, O.El beggar, T.Gadi, "A methodology for CIM modelling and its transformation to PIM," Journal of Information Engineering and Applications, ISSN 2224-5782 (print) ISSN 2225-0506 (online), Vol.3, No.2, 2013.

[11] A.Azzaoui, O.Rabhi, A.Mani, "A Model Driven Architecture Approach to Generate Multidimensional Schemas of Data Warehouses," International Journal of Online and Biomedical Engineering (iJOE), Vol. 15, No. 12, 2019, https://doi.org/10.3991/ijoe.v15i12.10720

[12] S.Garredu, Approche de méta-modélisation et transformations de modèles dans le contexte de la modélisation et simulation à évènements discrets: application au formalisme DEVS. Autre [cs.OH], Université Pascal Paoli, 2013. Français. NNT: 2013CORT0003. tel-01074207. https://doi.org/10.3166/tsi.25.1225-1260

[13] Y.Rhazali, Y.Hadi, A.Mouloudi, "Model Transformation with ATL into MDA from CIM to PIM Structured through MVC," Procedia Computer Science, Volume 83, pp. Pages 1096-1101, ISSN 1877-0509, https://doi.org/10.1016/j.procs.2016.04.229

[14] B.Bousetta, O.El Beggar, T.Gadi, "Automating Software Development Process: Analysis-PIMs to Design-PIM Model Transformation," International Journal of Software Engineering and Its Applications, Vol.7, No.5 (2013), pp.167-196, https://doi.org/10.14257/ijseia.2013.7.5.17

[15] Y.Ghazali, Y.Hadi, 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 (2016), pp. 125-142, https://doi.org/10.14257/ijseia.2016.10.4.13

[16] M.F.Amr, B.Riyami, K.Mansouri, M.Qbadou, "Interoperability of Hospital Information Systems through the modeling of the Web Service "Request for medical care"," 2020 IEEE International conference of Moroccan Geomatics (Morgeo), Casablanca, Morocco, 2020, pp. 1-6, https://doi.org/10.1109/morgeo49228.2020.9121893

[17] K.Bourquard, A.Coat, "A development process for interoperability and security frameworks for the health information technology," Pratiques et Organisation des Soins, 2009/4 Vol. 40, pages 283 à 296, ISSN 1952-9201. https://doi.org/10.3917/pos.404.0283

[18] A.Anwar, S.Ebersold, B.Coulette, M.Nassar, A.Kriouile, "Vers une approche à base de règles pour la composition de modèles. Application au profil VUML," L'objet: logiciel, bases de données, réseaux, pp. ISSN-L (papier): 1262-1137 ISSN (électronique): 1958-5756, Article pp.73-103 du Vol.13 n 4 (2007). https://doi.org/10.3166/obj.13.4.73-103

[19] A.Suchenia, P.Lopata, P.Wisniewski, B.Stachura-Terlecka, "Towards UML representation for BPMN and DMN models," MATEC Web Conf. Volume 252, 2019. III International Conference of Computational Methods in Engineering Science (CMES'18), p. https://doi.org/10.1051/matecconf/201925202007

[20] O.Macek, K.Richta, "“The BPM to UML activity diagram transformation using XSLT"," DATESO Computer Science (2009).

[21] A.Rodríguez, I.García-Rodríguez de Guzmán, E.Fernández-Medina, M.Piattini, "Semi-formal transformation of secure business processes into analysis class and use case models: An MDA approach," Information and Software Technology, p. Volume 52, Issue 9, 2010, Pages 945-971, ISSN 0950-5849. https://doi.org/10.1016/j.infsof.2010.03.015

[22] C.Hug, Méthode, modèles et outil pour la méta-modélisation des processus d'ingénierie de systèmes d'information, Autre [cs.OH]. Université Joseph-Fourier - Grenoble I, 2009. Français, tel-00430246v1

[23] D.Cetinkaya, A.Verbraeck, M.D.Seck, "MDD4MS: a model driven development framework for modeling and simulation," Society for Modeling & Simulation International, 2011, ISBN 9781617829505.

[24] S.Hammoudi, D.Lopes, "From Mapping Specification to Model Transformation in MDA: Conceptualization and Prototyping," In Proceedings of the Joint Workshop on Web Services and Model-Driven Enterprise Information Systems - Volume 1: WSMDEIS, (ICEIS 2005) ISBN 972-8865-27-9, pages 132-143, https://doi.org/10.5220/0002573001320143

[25] J. Bach, Un îlot formel pour les transformations de modèles qualifiables, Français. tel-01081055, Langage de programmation [cs.PL]. Université de Lorraine, 2014.

[26] F.Jouault, Contribution à l'étude des langages de transformation de modèles, 26 septembre 2006, Thèse de doctorat: Université de Nantes, Ecole des Mines de Nantes.

9 Authors

Meryem Fakhouri Amr is a PhD student at the laboratory SSDIA (Signals, Distributed Systems and Artificial Intelligence) of the Normal School of Technical Education ENSET of Mohammedia, University Hassan II of Casablanca, Morocco. She is currently working as a Systems Administration Engineer. His main interests are the interoperability of information systems, the transformation of MDA models, Web services, business intelligence, ontologies. Email: meryemfakhr@gmail.com

Nezha Benmoussa is a teacher of computer science, project management and research professor at the ENSET Mohammedia, University Hassan II of Casablanca, Morocco. Holder of a Master in Engineering of Information Systems and Training in 2012 at the Faculty of Sciences Ben M'sik of Casablanca. PhD in 2019 at the Faculty of Technical Sciences of Mohammedia. His research is focused on SIAD Decision Support Information Systems.

Khalifa Mansouri is now a teacher of computer science and researcher at the Hassan II University of Casablanca, ENSET Institute. He had a Diploma of ENSET Mohammedia in 1991, CEA in 1992 and first PhD in 1994 in Mohammed V University of Rabat, HDR in 2011 and second PhD in Hassan II University of Casablanca, Morocco. His research is focused on Real-time modeling systems, information systems, e-Learning systems and modeling, simulation and optimization of industrial systems.

Mohamed Qbadou is a teacher of computer science and researcher at the University Hassan II Casablanca, ENSET Institute. His research is focused on semantic Web, Distributed computing system, Knowledge database system, educational information system and e-learning, Assistive robotics, Natural Language processing. Diploma ENSET Mohammedia in 1992, CEA in 1993 and Ph.D. in 1998 at the Mohammed V University of Rabat, HDR in 2017 at the Hassan II University of Casablanca, Morocco.

Article submitted 2021-01-22. Resubmitted 2021-02-27. Final acceptance 2021-02-27. Final version published as submitted by the authors.