Aligning Software System Level with Business Process Level through Model-Driven Architecture

Maryam Habba[1], Mounia Fredj[3]
AlQualsadi Research Team
ENSIAS, Mohammed V University in Rabat
Rabat, Morocco

Samia Benabdellah Chaouni[2]
Department of Mathematics and Computer Science
Faculty of Sciences Ain Chock, Hassan II University
Casablanca, Morocco

***Abstract*—Information systems are intended to provide organisations with a new way of sustaining themselves, by helping them manage their activities using innovative technologies. Information systems require aligned levels for maximum effectiveness. In this context, business and information technology (IT) alignment is an important issue for the success of organisations. This paper presents the first step of the proposed approach to align the software system level, modelled by a Unified Modeling Language (UML) class diagram, with the business process level, modelled by the Business Process Model and Notation (BPMN) model. A model-driven architecture approach is proposed as a means to transform a set of BPMN models into a UML class diagram. A set of transformation rules is proposed, followed by guidelines that help apply those rules.**

Keywords*—*Information system alignment; business process; software system; Business Process Model and Notation (BPMN); Unified Modeling Language (UML); class diagram

I. Introduction

The effective operation of organisations requires an approach that assesses and corrects ambiguities between its different entities. In fact, an alignment approach has become crucial for the continuity of organisations' information systems, as it provides solutions to problems associated with the diverse changes that may occur in these organisations' entities. Several previous studies have examined the subject of business/IT alignment [1]–[5]. The analysis and proposed approach described in this paper are based on the relevance of alignment in various situations. Indeed, in practice, an information system with aligned levels may undergo changes in one of these levels due to the improvement of goals or other factors. As a result, the levels will become misaligned. In another context, the levels of an organisation's information system may be modelled by different teams. Each team may then have a different perspective regarding the system, which can also result misaligned levels. Another example is the case of two organizations that merge in such a way that their levels may be of different natures. As a consequence, the resulting information system will contain levels that are not aligned. For all the mentioned situations, it seems to be a strong necessity to apply an alignment approach in order to have a successful information system.

In this context, approaches of related work based on business process and software system levels are analysed through this paper. Afterwards, an alignment approach is proposed. The first step of this approach consists of providing a series of rules to transform a set of Business Process Model and Notation (BPMN) models into a Unified Modeling Language (UML) class diagram, based on model-driven architecture (MDA).

The proposed approach contributes to the existing literature by transforming a series of source-level models that contain a significant number of BPMN elements into a UML class diagram. Moreover, the proposed approach provides a method for preserving target level information.

This paper is organised as follows: Section II presents the background of the topic; it introduces the concept of alignment and transformation through MDA. Section III provides a brief overview of related work, while the proposed approach is presented in Section 0. Section V of this paper presents a case study. Finally, a conclusion describes the future work.

II. Background

A. The Concept of Alignment

Alignment is an important topic that has been of interest for decades. Various expressions have been used to describe it in the existing literature. Chan [6] uses the terms fit and synergy. Henderson and Venkatraman [7] employ the terms fit, integration, and interrelationships. Reich and Benbasat [8] use the word linkage. Ciborra [9], defines alignment as a bridge. Smaczny [10] describes it as fusion. Luftman [11] uses the term harmony, and Nickels [12] names it congruence.

According to Ullah [5], alignment between business and IT concerns "the optimized synchronization between dynamic business objectives/processes and respective technological services provided by IT". For Luftman [11], business-IT alignment consists on the application of IT in a timely and a suitable manner, in harmony with business strategies, goals and needs. This definition of alignment considers: the way that IT is aligned with the business, and the way the business should or might be aligned with IT.

In the current work, alignment of a target level with a source level is defined as a method that ensures the continued operation of the target level, while remaining suitable to the source level.

B. Transformation and Model-Driven Architecture

Model-driven engineering considers models to have a very important role in software development. In this context, the

This work is sponsored by the Excellence Research Scholarships Program of CNRST (National Centre for Scientific and Technical Research) of Morocco (grant number 51UM52016).

Object Management Group (OMG) made their MDA initiative public [13]. Fig. 1 presents the model transformation concept [14] recommended by MDA. The model transformation takes a source model that conforms to its source metamodel and a target metamodel as input. It then uses a set of transformation rules to generate as output a target model that conforms to the target metamodel as output.

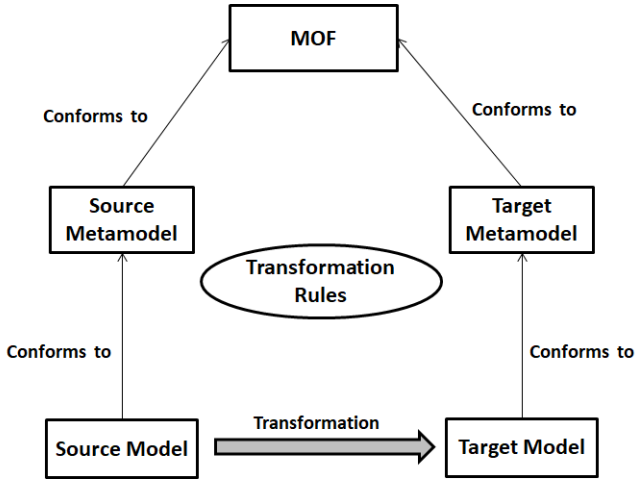

Fig. 1. Concept of Transformation in MDA [14].

III. Related Work

In the previous work [15], a pattern system was proposed as a guideline, to help organisations apply the alignment. The systematic literature review conducted by Habba et al. [16] identified various approaches of alignment of business requirement, business process and software system levels, that use different modeling languages.

We focus on UML and BPMN languages because they are standards defined by the Object Management Group (OMG). More precisely, in this paper, we focus on a business process level modelled by BPMN and a software system level modelled by a UML class diagram.

BPMN and class diagrams are subjects of interest in different approaches. Amr et al. [17] propose an MDA approach for transforming a BPMN source model into a UML class diagram, using a set of transformation rules. Brdjanin et al. [18] present an approach for the automated generation of a conceptual database model represented by a UML class diagram, from one BPMN model. Brdjanin et al. [19] take a set of business process models into account. Khlif et al. [20] describe an approach to transform a business process model into a class diagram, based on semantic and structural aspects. Rhazali et al. [21] suggest a set of transformation rules for transforming a BPMN model into a use case, state and class diagrams. Cruz et al. [22] propose an approach to obtain a data model from a business process model. Cruz et al. [23] present rules to transform a set of business process models into a data model. Kriouile et al. [24] describe an approach to transform a BPMN model into a domain class model. Bousetta et al. [25] propose an approach to building a domain class diagram, based on a BPMN model, using a set of business rules.

In organisations, models of both levels usually exist. The aim, therefore, is to align them. By analysing existing approaches, we notice that:

- Existing approaches propose transformation from the source level into the target level. However, an approach-based transformation is not always sufficient to apply alignment when business process and software system models exist. In fact, this approach causes a loss of information. Fig. 2 presents the result of applying one of the existing approaches when models exist in an organisation. M1 represents the business process level model and M2 is the software system level model. The existing approaches generate a new UML class diagram (M2') that is different from model M2. Therefore, information associated with the existing UML class diagram will be lost.

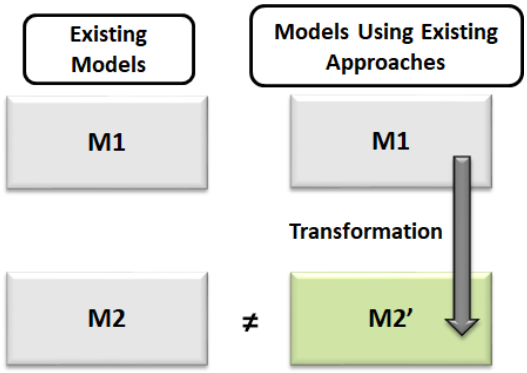

Fig. 2. Application of an Existing Approach.

- The majority of approaches take one model at the source level into consideration. Only two approaches ([19] and [23]) have achieved transformation using a set of BPMN models as a source. However, operations are not considered in the metamodel of the target model. (Table I, columns 3 and 6).
- The existing approaches do not consider all BPMN elements, such as all types of tasks and all types of data, in the source model (Table I, columns 4 and 5).

We synthesize the existing approaches in Table I, according to the criteria below:

- Preserving information: This column indicates if the proposed approach can be executed when the models of the two levels exist in the organisation.
- Considering a set of BPMN models: This column indicates if the proposed approach considers a set of BPMN models in business process level.
- Considering all types of tasks: This column indicates if the approach considers all types of tasks in the source model or not.
- Considering all types of data: This column indicates if the approach considers all types of data in the source model or not.

- Considering operations: This column indicates if the approach considers all types of data in the target model or not.

In Table I, Y shows that the criterion is considered.

TABLE I. SYNTHESIS OF APPROACH

Ref.	Preserving information	Considering a set of BPMN models	Considering all types of tasks	Considering all types of data	Considering operations
[17]	-	-	-	-	Y
[18]	-	-	-	-	-
[19]	-	Y	-	-	-
[20]	-	-	-	-	Y
[21]	-	-	-	-	Y
[22]	-	-	-	-	-
[23]	-	Y	-	-	-
[24]	-	-	-	-	Y
[25]	-	-	-	-	-

This analysis of existing approaches reveals the need for an alignment approach that preserves target level information, considers a large number of BPMN and UML elements, and uses a set of BPMN models as a source.

IV. PROPOSED APPROACH

A. Overview of the Proposed Approach

The aim of the proposed approach is to reduce the gap between the business process level and the software system level of an organisation, without losing information. The business process level is modelled using a set of BPMN models. It may be composed of a set of collaboration diagrams and expanded sub-processes, and it contains a high number of metamodel elements. The software system level is modelled by a UML class diagram. Fig. 3 illustrates a representation of the proposed approach. We assume that the organisation has two levels, composed of a set of BPMN models and one existing UML class diagram. The organisation needs to align the software system level with the business process level. The proposed alignment approach encompasses two steps:

1) Step 1: Transformation. This step consists of the application of rules to transform a set of BPMN models into a generated UML class diagram. It considers the important elements of the BPMN metamodel and the UML class diagram metamodel, including all types of tasks and all types of data.

2) Step 2: Composition. This step consists of creating a fusion between the UML class diagram generated in step 1 and the existing UML class diagram. The result is a final UML class diagram that will represent the software system level, which is aligned with the business process level.

By applying the two steps, the target level will be complete, as it contains the information related to the existing class diagram as well as the information related to the generated class diagram.

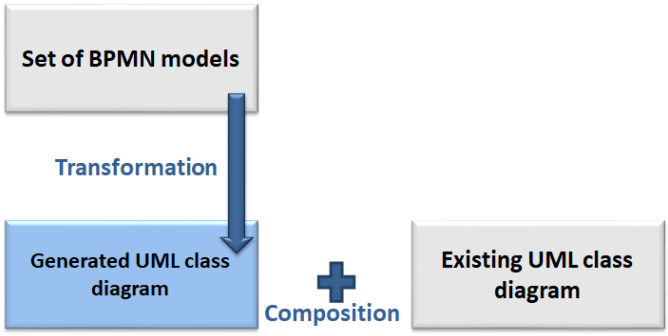

Fig. 3. Representation of the Approach.

In this paper, the first step of the approach is detailed: Transformation. This step uses a set of BPMN models as a source and applies transformation rules to derive a UML class diagram.

B. Example of a Set of BPMN Models

In this section, a set of BPMN models are presented. They represent the business process level of an organisation. Fig. 4 and Fig. 6 represent collaboration diagrams. Fig. 5, Fig. 7 and Fig. 8 depict expanded sub-processes, represented in collaboration diagrams as collapsed sub-processes.

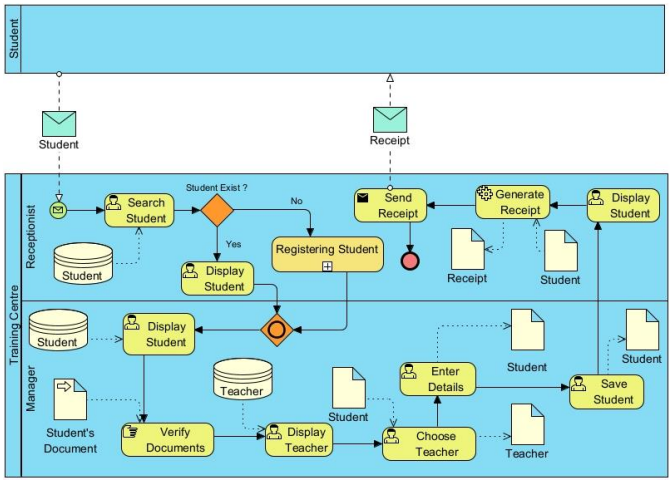

Fig. 4. Collaboration Diagram for Assigning a Professor to a Student.

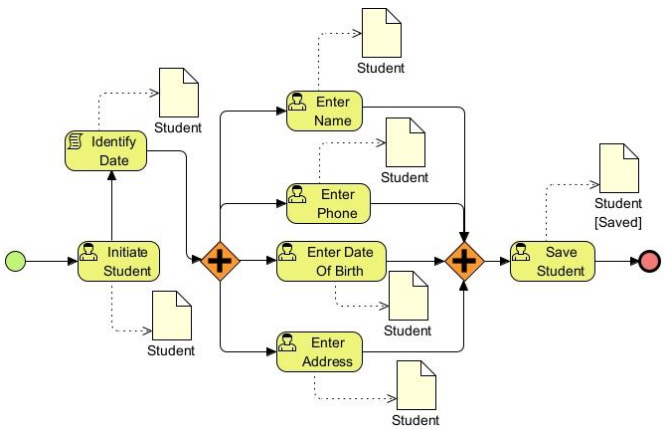

Fig. 5. Expanded Sub-process "Registering Student".

Fig. 4 illustrates the collaboration diagram for assigning a professor to a student. It is composed of two pools: "Student", as a black box and "Training Centre", which contains two lanes "Receptionist" and "Manager". The diagram begins when a student arrives at the training centre for enrolment. The first activity is performed by the receptionist. It consists of searching for a student to see if they are registered or not. Two cases are possible: if the student is registered, the student's file will be displayed. If not, the receptionist will proceed to the registration phase. Next, the manager will display the student file, verify documents, display teacher, choose teacher, enter details and then save the student file. Then, the receptionist will display the student file, generate a receipt and finally send the receipt. The second collaboration diagram (Fig. 6) contains two pools: "Supplier", represented as a black box, and "Training Centre" which contains two lanes ("Receptionist" and "Teacher"). The teacher displays a student file, and then prepares a note that will be displayed by the receptionist to prepare a quote request that will be sent to a supplier. Finally, the receptionist will receive a quote.

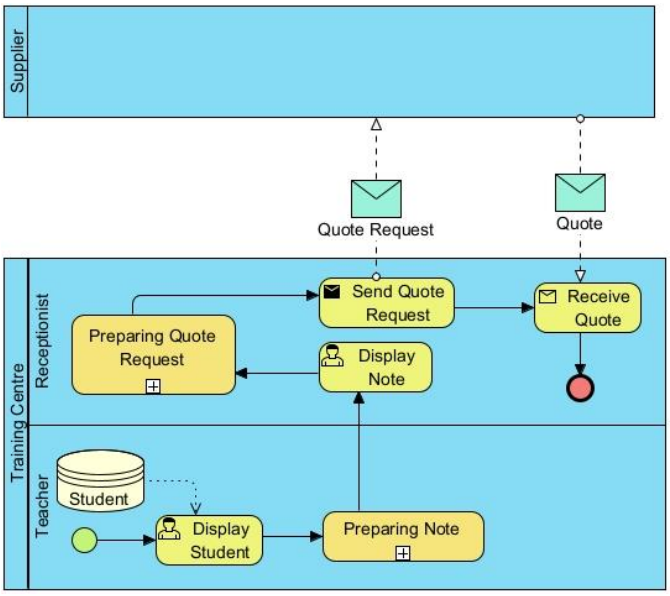

Fig. 6. Collaboration Diagram for Requesting a Quote.

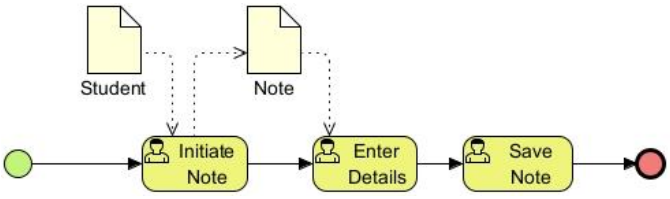

Fig. 7. Expanded Sub-process "Preparing Note".

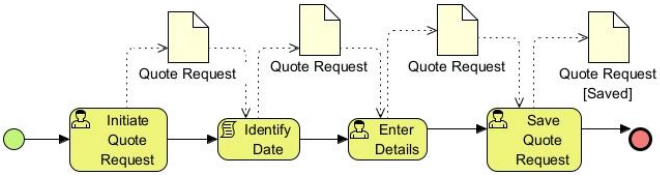

Fig. 8. Expanded Sub-process "Preparing Quote Request".

C. Transformation Rules

In order to transform a set of BPMN models into a UML class diagram, a set of rules that take different elements of BPMN into consideration are proposed.

1) Data (data object, data input, data output and data store) related to send, receive, user, service, script or business rule tasks.

TR1: This rule transforms data that is related to send, receive, user, service, script or business rule tasks into a class with the same name, containing an attribute (id).

2) Message

TR2: This rule transforms a message into a class with the same name, containing an attribute (id).

3) User task, service task, script task, business rule task

a) A task with input and output data

TR3.1: Let TusvscbrIO be a user, service, script or business rule task with data DIT (data object, data input or data store) as input, and data DOT (data object, data output or data store) as output.

TR3.1 transforms data DIT and DOT into classes, according to TR1. Then, it transforms the task TusvscbrIO into an operation that will belong to the resulting class of DOT. The name of the operation is obtained by removing the spaces between words, making the first letter of the first word in the name of the task lowercase and the first letter of all other words uppercase. This name change is called "reduced form of the task name", RTusvscbrIO. The DIT and DOT classes will be linked by an association (if the names of data DIT and data DOT are different). Fig. 9 presents an illustration of rule TR3.1. To identify multiplicities, the following guidelines are applied:

- If DIT is a singular data object, singular data input or data store, then the multiplicity on the side of the class corresponding to DIT is of value 1.
- If DIT is a collection data object or collection data input, then the value of the side of the class that corresponds to DIT is 1. .*.
- If DOT is a singular data object, singular data output or data store, then the multiplicity on the side of the class corresponding to DOT is of value 0..1.
- If DOT is a collection data object or collection data output, then the value of the side of the class that corresponds to DOT is 0..*.

Fig. 10 presents an example of rule TR3.1. The user task "Initiate Note" has the data object "Student" in singular form as input, and the data object "Note" in singular form as output. Thus, in the class diagram, two classes DIT and DOT that contains an attribute (id) will be created. The class "Note" will contains the operation "initiateNote". An association will be generated between the classes with multiplicities 1 on the side of the class "Student" and 0..1 on the side of the class "Note".

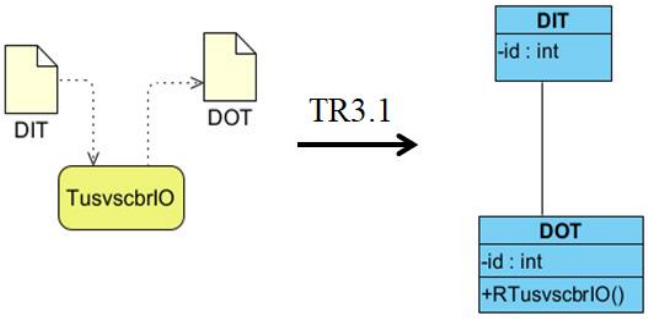

Fig. 9. Illustration of Rule TR3.1.

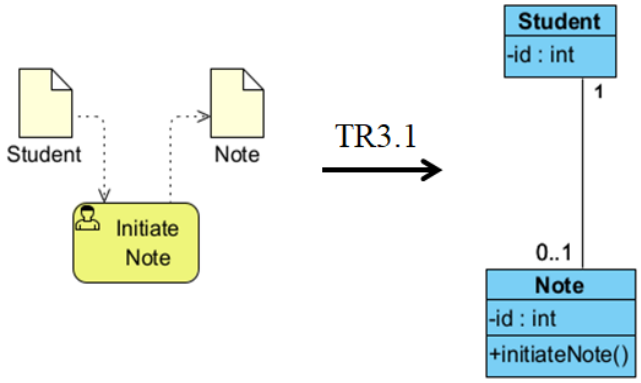

Fig. 10. Example of Rule TR3.1.

b) A task with output data only

TR3.2: Let TusvscbrO be a user, service, script or business rule task with data JDOT (data object, data output or data store) as output, and no input data.

Rule TR3.2 transforms data JDOT into a class, according to TR1. Then, it transforms the task TusvscbrO into an operation that will belong to the resulting JDOT class. The name of the operation will be the reduced form of the task name, RTusvscbrO. Fig. 11 presents an illustration of rule TR3.2 while Fig. 12 presents an example.

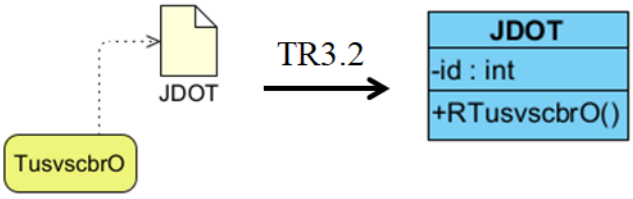

Fig. 11. Illustration of Rule TR3.2.

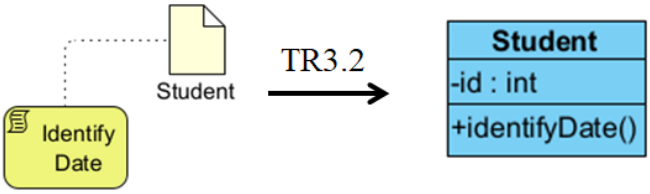

Fig. 12. Example of Rule TR3.2.

c) A task with input data only

TR3.3: Let TusvscbrI be a user, service, script or business rule task linked to data JDIT (data object, data input or data store) as input.

Rule TR3.3 transforms data JDIT into a class, according to TR1. Then, it transforms task TusvscbrI into an operation that will belong to the resulting JDIT class. The name of the operation will be the reduced form of the task name, RTusvscbrI. Fig. 13 presents an illustration of rule TR3.3 while Fig. 14 presents an example.

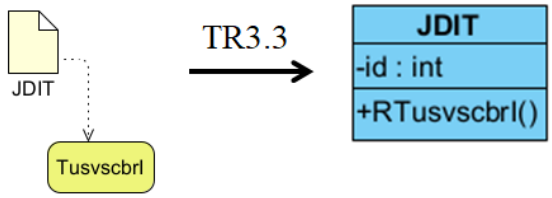

Fig. 13. Illustration of Rule TR3.3.

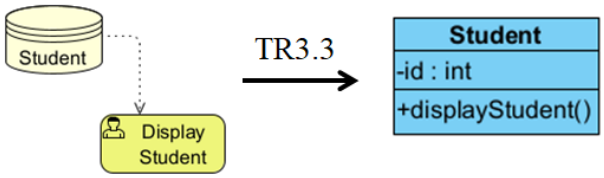

Fig. 14. Example of Rule TR3.3.

d) A task without input or output data

TR3.4: Let Tusvscbr be a user, service, script or business rule task that is not linked to any data.

Rule TR3.4 transforms task Tusvscbr into a class, named by using the singular form of the direct object of the task name. It's designated by SOTusvscbr. The class will contain an attribute id and an operation. The name of the operation will be the reduced form of the task name, RTusvscbr.

Fig. 15 presents an illustration of rule TR3.4 while Fig. 16 presents an example. The task Tusvscbr, named "Display Note", is not related to any data. "Note" is a direct object of task Tusvscbr and it is in a singular form. For this reason, the task named "Display Note" will be transformed into a class named "Note", containing an attribute id and an operation named "displayNote".

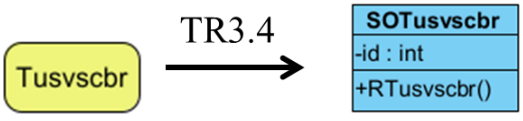

Fig. 15. Illustration of Rule TR3.4.

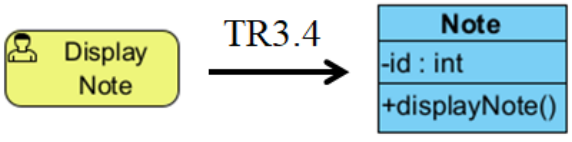

Fig. 16. Example of Rule TR3.4.

4) Send task

a) A send task with input and output data

TR4.1: Let TsdIO be a send task with data DIsdT (data object, data input or data store) as input and data DOsdT (data object, data output or data store) as output.

Rule TR4.1 transforms data DIsdT and DOsdT into classes, according to TR1. Then, it transforms task TsdIO into an operation that will belong to the resulting DOsdT class. The name of the operation will be the reduced form of the task name, RTsdIO. The DIsdT and DOsdT classes will be linked by an association (if the names of data DIsdT and data DOsdT are different). Fig. 17 presents an illustration of rule TR4.1. To identify multiplicities, the following guidelines are applied:

- If DIsdT is a singular data object, a singular data input or data store, then the multiplicity on the side of the class corresponding to DIsdT is of value 1.
- If DIsdT is a collection data object or collection data input, then the value of the side of the class that corresponds to DIsdT is 1. .*.
- If DOsdT is a singular data object, a singular data output or data store, then the multiplicity on the side of the class corresponding to DOsdT is of value 0..1.
- If DOsdT is a collection data object or a collection data output, then the value of the side of the class that corresponds to DOsdT is 0..*.

b) A send task with output data

TR4.2: Let TsdO be a send task with data JDOsdT (data object, data output or data store) as output.

Rule TR4.2 transforms data JDOsdT into a class, according to TR1. Then, it transforms task TsdO into an operation that will belong to the resulting JDOsdT class. The name of the operation will be the reduced form of the task name, RTsdO. Fig. 18 presents an illustration of the rule TR4.2.

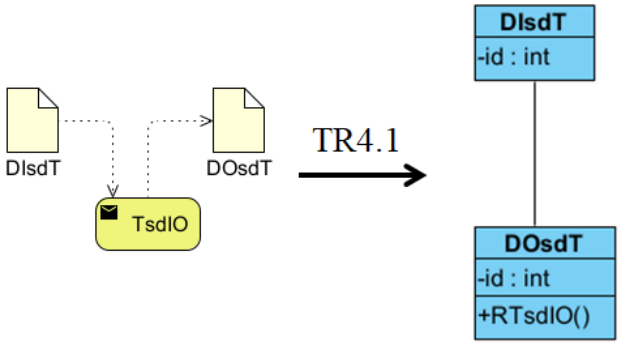

Fig. 17. Illustration of Rule TR4.1.

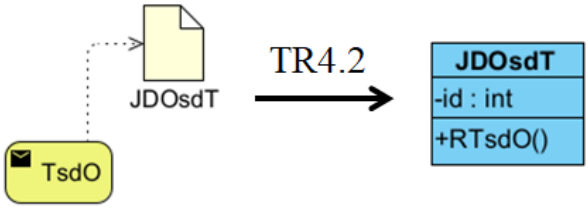

Fig. 18. Illustration of Rule TR4.2.

c) A send task with input data

TR4.3: Let TsdI be a send task with data JDIsdT (data object, data input or data store) as input.

Rule TR4.3 transforms data JDIsdT into a class, according to rule RT1. Then, it transforms task TsdI into a class, named using the singular form of the direct object (OTsdI) of the task name. It's designated by SOTsdI. The class will contain an attribute id and an operation. The name of the operation will be the reduced form of the task name, RTsdI. The JDIsdT and SOTsdI classes will be linked by an association (if the names of JDIsdT and SOTsdI are different). Fig. 19 presents an illustration of rule TR4.3. To identify multiplicities, the following guidelines are applied:

- If JDIsdT is a singular data object, a singular data input or a data store, then the multiplicity on the side of the class corresponding to JDIsdT is of value 1.
- If JDIsdT is a collection data object or collection data input, then the value of the side of the class that corresponds to JDIsdT is 1. .*.
- If OTsdI is in a singular form, then the multiplicity on the side of the class corresponding to SOTsdI is of value 0..1.
- If OTsdI is in plural, then the value of the side of the class that corresponds to SOTsdI is 0..*.

d) A send task without input or output data

TR4.4: Let Tsd be a send task without data.

Rule TR4.4 transforms task Tsd into a class, named using the singular form of the direct object of the task name. It's designated by SOTsd. The class will contain an attribute id and an operation. The name of the operation will be the reduced form of the task name, RTsd. Fig. 20 presents an illustration of rule TR4.4, while Fig. 21 presents an example.

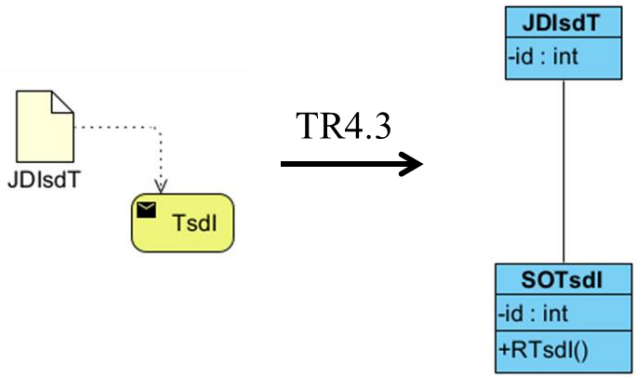

Fig. 19. Illustration of Rule TR4.3.

Fig. 20. TR4.4 Illustration.

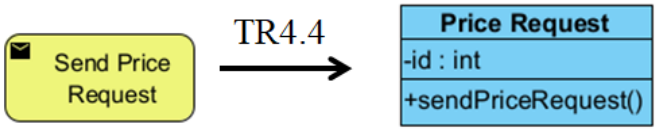

Fig. 21. Example of Rule TR4.4

5) Receive task

a) A receive task with input and output data

TR5.1: Let TrvIO be a receive task with data DIrvT (data object, data input or data store) as input and data DOrvT (data object, data output or data store) as output.

Rule TR5.1 transforms data DIrvT and DOrvT into classes, according to TR1. Then, it transforms task TrvIO into an operation that will belong to the resulting DIrvT class. The name of the operation will be the reduced form of the task name, RTrvIO. The DIrvT and DOrvT classes will be linked by an association (if the names of data DIrvT and data DOrvT are different). Fig 22 presents an illustration of the rule TR5.1.

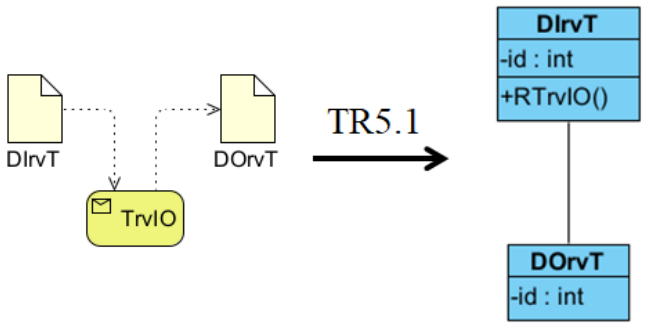

Fig. 22. Illustration of Rule TR5.1.

b) A receive task with input data

TR5.2: Let TrvI be a receive task with data JDIrvT (data object, data input or data store) as input. Rule TR5.2 transforms task TrvI into an operation in the class, corresponding to data TrvI. The name of the operation will be the reduced form of the task name, RTrvI. Fig. 23 presents an example of rule TR5.2.

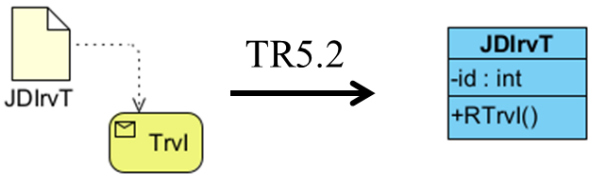

Fig. 23. Illustration of Rule T5.2.

c) A receive task with output data

TR5.3: Let TrvO be a receive task with data JDOrvT (data object, data output or data store) as output.

Rule TR5.3 transforms data JDOrvT into a class, according to rule RT1. Then, it transforms task TrvO into a class, named using the singular form of the direct object (OTrvO) of the task name. It's designated by SOTrvO. The class will contain an attribute id and an operation. The name of the operation will be the reduced form of the task name, RTrvO. Fig. 24 presents an illustration of rule TR5.3. The JDOrvT and SOTrvO classes will be linked by an association (if the names of JDOrvT and SOTrvO are different). To identify multiplicities, the following guidelines are applied:

- If OTrvO is in a singular form, then the multiplicity on the side of the class corresponding to SOTrvO is of value 1.
- If OTrvO is in a plural form, then the value of the side of the class that corresponds to SOTrvO is 1..*.
- If JDOrvT is a singular data object, singular data output or a data store, then the multiplicity on the side of the class corresponding to JDOrvT is of value 0..1.
- If JDOrvT is a collection data object or collection data output, then the value of the side of the class that corresponds to JDOrvT is 0. .*.

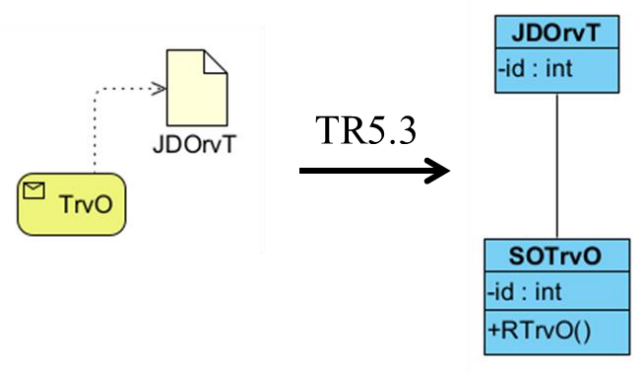

Fig. 24. Illustration of Rule TR5.3.

d) A receive task without input or output data

TR5.4: Let Trv be a receive task without data.

Rule TR5.4 transforms task Trv into a class, named using the singular form of the direct object of the task name. It's designated by SOTrv. The class will contain an attribute id and an operation. The name of the operation will be the reduced form of the task name, RTrv. Fig. 25 presents an illustration of rule TR5.4, while TR5.4 and Fig. 26 presents an example.

Fig. 25. Illustration of Rule TR5.4.

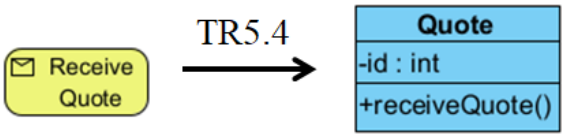

Fig. 26. Example of Rule TR5.4.

6) Pool

TR6: Rule TR6 transforms a pool into a class with the same name as the pool, containing five attributes (id, name, email, phone, address).

7) Lane within a pool

TR7: Rule TR7 transforms a lane within a pool into classes with the same names as the pool and the lane. Each class will contain five attributes (id, name, email, phone, address). Then, it adds aggregation between the class corresponding to the pool and the class corresponding to the lane (multiplicities 1 and 0..* respectively).

8) Relationship between a pool/lane and a non-manual task belonging to it

TR8: Rule TR8 transforms a relationship between a pool/lane and a non-manual task belonging to it into an association between the class corresponding to the pool/lane and the class that contains the reduced form of the task's name (multiplicities 1 and 0 respectively).

9) Relationship between a message and an element that is the source or the target of the message flow (pool, event belonging to a pool/lane or task belonging to a pool/lane).

TR9: Rule TR9 transforms a relationship between a message and an element that is the source or the target of the message flow (pool, event belonging to a pool/lane or task belonging to a pool/lane) into an association between the class corresponding to the pool/lane and the class corresponding to the message (multiplicities 1 and 0..* respectively).

D. Isolated elements

In this section, the isolated elements are presented. An isolated element belongs to the BPMN metamodel and does not have an equivalent in the UML class diagram metamodel.

1) Manual task: A manual task is performed without the intervention of any application. Therefore, it will not be visualised by the software. For this reason, this type of task is considered an isolated element, and it has no equivalent in the class diagram.

2) Data linked to a manual task: Data that is linked to a manual task will not have traceability through the system. Because the manual task has no visualisation, this data is considered an isolated element.

3) Event: Usually, an event is a fact that occurs during the process. Because the class diagram represents a static aspect, an event is considered an isolated element.

4) Gateway: The goal of a gateway is to control the convergence or divergence of flows in a process. It does not have an equivalent in the class diagram.

5) Artifact: An artifact (group or annotation) aims to provide more clarity to understand the process. It does not have an equivalent in the class diagram.

6) Sequence flow: A sequence flow can indicate the flow of activities through a process. It does not have an equivalent. In fact, the tasks linked by the sequence flows that have an equivalent in the class diagram.

7) Association: An association is a way to link the artifacts with different BPMN elements and does not have an equivalent in the class diagram.

E. Steps for a Set of Models

In order to apply the rules presented in section C, a series of steps based on BPMN notation are presented in this section, to transform a set of BPMN models into a class diagram. Fig. 27 shows the process of this transformation. It constitutes five looped sub-processes.

1) Transformation of task: this sub-process can

a) Identify non manual task.

b) Identify task type.

According to the type of task, apply rule TR3.1, TR3.2, TR3.3, TR3.4, TR4.1, TR4.2, TR4.3, TR4.4, TR5.1, TR5.2, TR5.3 or TR5.4, which all call rule TR1. When applying a rule, a check is performed to determine whether an element (class, operation or association) has already been created by another rule. If it has:

- All the instructions associated with that rule are applied, except creation of the element.
- If an association already exists, such that the multiplicities are different, the existing association is kept, and the union of multiplicities is applied for each end of the association.

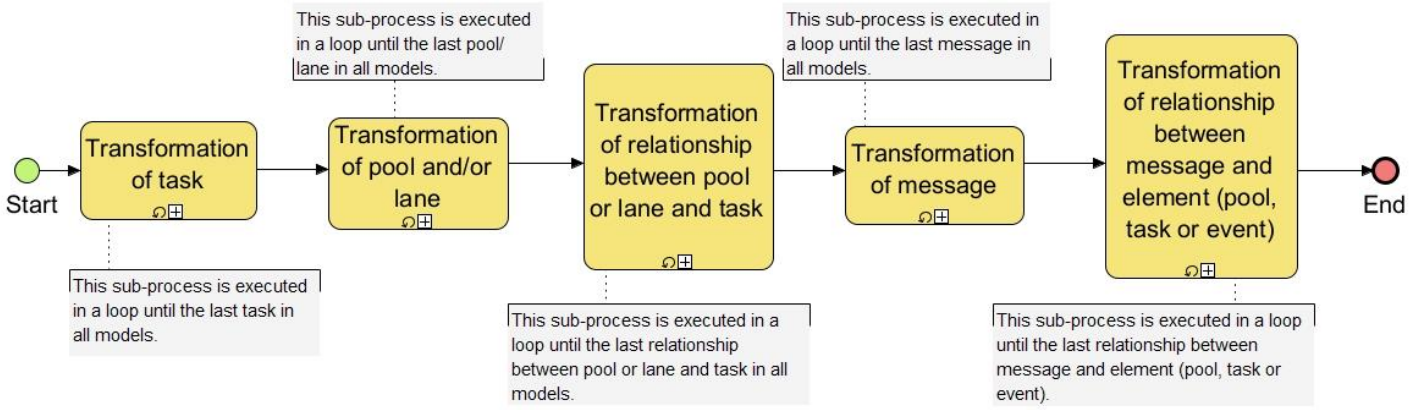

Fig. 27. The Transformation Process.

2) Transformation of pool and/or lane: for each pool that exists in the different models this sub-process can.

a) Identify the pool

b) According to the type of pool (with or without lanes), apply the rule TR6 or TR7. When applying a rule, a check is performed to determine whether an element (class, attribute or aggregation) has already been created by another rule. If this is the case, all instructions *associated with that rule* are applied, except creation of the element.

3) Transformation of relationship between pool or lane and task: for each relationship between a pool or a lane and a task this sub-process can.

a) Identify the relationship between the pool or lane and task.

b) According to the type of relationship, apply the rule TR8. When applying a rule, a check is performed to determine whether an association has already been created by another rule. If an association already exists such that the multiplicities are different, the existing association is kept, and the union of multiplicities is applied for each end of the association.

4) Transformation of message: for each message this sub-process can.

a) Identify a message.

b) Apply TR2.

5) Transformation of relationship between message and element (pool, task or event): for each relationship between a message and an element (pool, task or event) this sub-process can.

a) Identify the relationship between a message and an element (pool, task or event).

b) Apply TR9. When applying the rule, a check is performed to determine whether an association has already been created by another rule. If an association already exists such that the multiplicities are different, the existing association is kept, and the union of the multiplicities is applied for each end of the association.

V. Case Study

In order to illustrate the application of the proposed transformation rules, the set of BPMN models represented in section B are transformed into a UML class diagram. The example describes two collaboration diagrams and three expanded sub-processes. The first context is the training centre that receives students who want to have a teacher as a mentor. In the second context, the training centre communicates with suppliers to obtain a particular quote, related to student needs. Fig. 28 presents the class diagram obtained by the application of the transformation rules, according to the global process of transformation presented in Section EIVE.

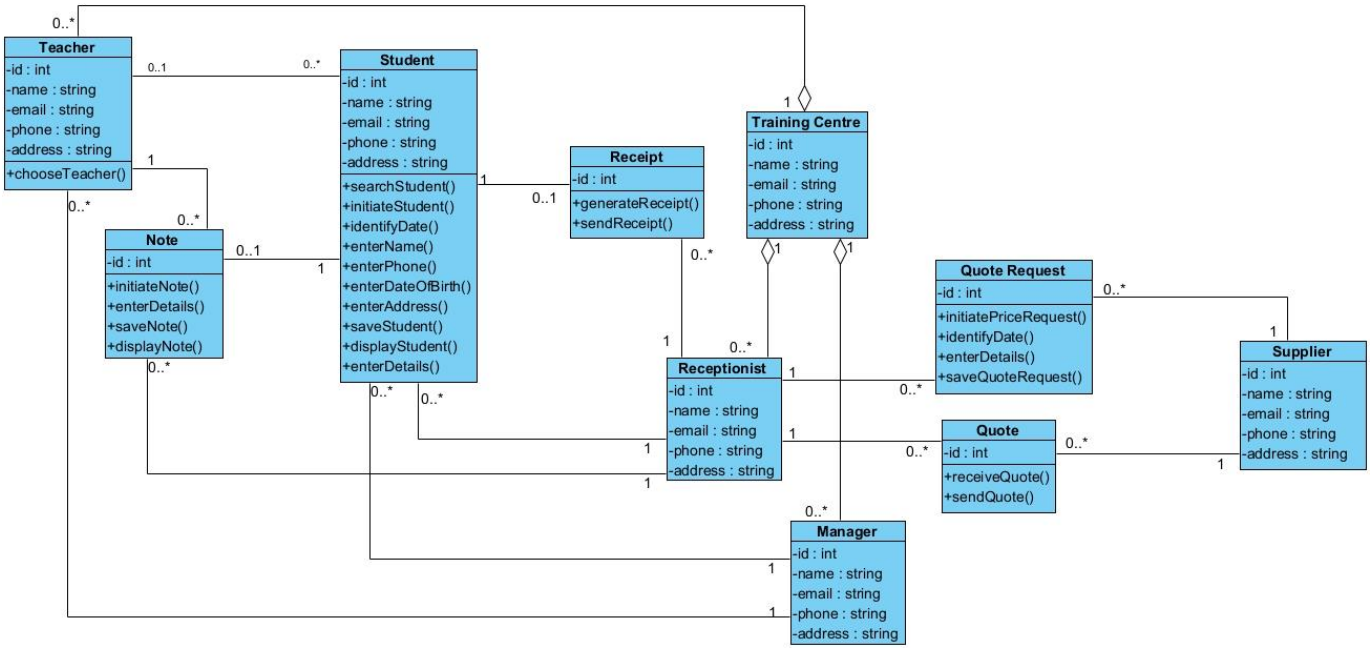

Fig. 28. Obtained Class Diagram.

VI. Conclusion

In this paper, a method for aligning software system level using UML class diagram with business process level using BPMN notation is proposed. This proposal enables to contribute to the alignment process of an organization, by considering a set of models at the source level that contains a large number of BPMN metamodel elements. Furthermore, the method aims to preserve information, filling a crucial need for organisations' long-term success. The first phase was described here, detailing a series of rules for transforming a set of BPMN models into a UML class diagram. Moreover, a guideline is presented to help organisations apply rules properly. A set of isolated elements is also presented to explain the BPMN elements that are not considered by the transformation rules. The application of the proposed rules is demonstrated in a case study. In future work, we aim to use the ATLAS Transformation Language to automate the proposed transformation rules.

Acknowledgment

We would like to thank the Excellence Research Scholarships Program of CNRST (National Centre for Scientific and Technical Research) of Morocco for supporting this research. This work is under research grant number 51UM52016.

References

[1] T.Wasiuk, F.P.C.Lim, "Factors Influencing Business IT Alignment". International Journal of Smart Business and Technology, vol.9, no.1, pp.1-12, Mar. 2021.

[2] H. Darii, J. Laval, V. Botta-Genoulaz, and V. Goepp, "Measurement of the business/IT alignment of information systems." In ILS 2020-8th International Conference on Information Systems, Logistics and Supply Chain, pp. 228-235. 2020.

[3] P. Gajardo and L. P. Ariel, "The business-it alignment in the digital age." In The 13th Mediterranean Conference on Information Systems (ITAIS & MCIS), Naples, Italy. 2019.

[4] M. Zhang, H. Chen, and A. Luo, "A systematic review of business-IT alignment research with enterprise architecture," IEEE Access, vol. 6, pp. 18933–18944, 2018.

[5] A. Ullah and R. Lai, "A systematic review of business and information technology alignment," ACM Trans. Manag. Inf. Syst., vol. 4, no. 1, pp. 1–30, 2013.

[6] Y. E. Chan, "Business Strategy, information system strategy, and strategic fit: Measurement and performance impacts," p. 362, 1992.

[7] J. C. Henderson and H. Venkatraman, "Strategic alignment: Leveraging information technology for transforming organizations," IBM Syst. J., vol. 38, no. 2.3, pp. 472–484, 1999.

[8] B. H. Reich and I. Benbasat, "Measuring the linkage between business and information technology objectives," MIS Q. Manag. Inf. Syst., vol. 20, no. 1, pp. 55–77, 1996, doi: 10.2307/249542.

[9] C. U. Ciborra, "De profundis? Deconstructing the concept of strategic alignment," Scand. J. Inf. Syst., vol. 9, no. 1, p. 2, 1997.

[10] T. Smaczny, "Is an alignment between business and information technology the appropriate paradigm to manage IT in today's organisations?," Manag. Decis., 2001.

[11] J. Luftman, "Assessing business-IT allignment maturity," in Strategies for information technology governance, Igi Global, pp. 99–128, 2004.

[12] D. W. Nickels, "Business and IT Alignment: What We Know That We Still Don't Know," Proc. 7th Annu. Conf. South. Assoc. Inf. Syst., pp. 79–84, 2004.

[13] R. Soley, "Model driven architecture," OMG white Pap., vol. 308, no. 308, p. 5, 2000.

[14] J. Bézivin and O. Gerbé, "Towards a precise definition of the OMG/MDA framework," in Proceedings 16th Annual International Conference on Automated Software Engineering (ASE 2001), pp. 273–280, 2001.

[15] M. Habba, M. Fredj, and S. B. Chaouni, "Towards an operational alignment approach for organizations," ACM Int. Conf. Proceeding Ser., pp. 29–34, 2017, doi: 10.1145/3149572.3149602.

[16] M. Habba, M. Fredj, and S. Benabdellah Chaouni, "Alignment between Business Requirement, Business Process, and Software System: A Systematic Literature Review," J. Eng., vol. 2019, 2019.

[17] M. F. Amr, N. Benmoussa, K. Mansouri, and M. Qbadou, "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," iJOE, vol. 17, no. 05, p. 49, 2021.

[18] D. Brdjanin, G. Banjac, and S. Maric, "Automated synthesis of initial conceptual database model based on collaborative business process model," in International Conference on ICT Innovations, pp. 145–156, 2014.

[19] D. Brdjanin, A. Vukotic, G. Banjac, D. Banjac, and S. Maric, "Automatic Derivation of Conceptual Database Model from a Set of Business Process Models," in 2020 International Conference on INnovations in Intelligent SysTems and Applications (INISTA), pp. 1–8, 2020.

[20] W. Khlif, N. Elleuch, E. Alotabi, and H. Ben-Abdallah, "Designing BP-IS Aligned Models: An MDA-based Transformation Methodology," in Proceedings of the 13th International Conference on Evaluation of Novel Approaches to Software Engineering, pp. 258– 266, 2018.

[21] Y. Rhazali, Y. Hadi, and A. Mouloudi, "A methodology of model transformation in MDA: From CIM to PIM," Int. Rev. Comput. Softw., vol. 10, no. 12, pp. 1186–1201, 2015, doi: 10.15866/irecos.v10i12.8088.

[22] E. F. Cruz, R. J. Machado, and M. Y. Santos, "From business process modeling to data model: A systematic approach," Proc. - 2012 8th Int. Conf. Qual. Inf. Commun. Technol. QUATIC 2012, pp. 205–210, 2012, doi: 10.1109/QUATIC.2012.31.

[23] E. F. Cruz, R. J. Machado, and M. Y. Santos, "Deriving a Data Model from a Set of Interrelated Business Process Models.," in ICEIS (2), pp. 49–59, 2015.

[24] A. Kriouile, N. Addamssiri, T. Gadi, and Y. Balouki, "Getting the static model of PIM from the CIM," in 2014 Third IEEE International Colloquium in Information Science and Technology (CIST), pp. 168–173, 2014.

[25] B. Bousetta, O. El Beggar, and T. Gadi, "A methodology for CIM modelling and its transformation to PIM," J. Inf. Eng. Appl., vol. 3, no. 2, pp. 1–22, 2013.