Alignment of Software System Level with Business Process Level: Resolving Syntactic and Semantic Conflicts

Samia Benabdellah Chaouni[1]
Department of Mathematics and Computer Science
Faculty of Sciences Ain Chock, Hassan II University
Casablanca, Morocco

Maryam Habba[2], Mounia Fredj[3]
AlQualsadi Research Team
ENSIAS, Mohammed V University in Rabat
Rabat, Morocco

***Abstract*—Information systems help organizations manage their entities with innovative technologies. These entities are often very different in nature. In this paper, we consider a business process level based on a set of Business Process Model and Notation (BPMN) models and a software system level based on a Unified Modeling Language (UML) class diagram. The differences between these entities make them difficult to align. In addition, an organization's BPMN models may be designed by different teams, which can cause syntactic and semantic heterogeneities. We present the first step of our proposed approach for aligning a software system level with a business process level without conflict (redundancy and lost information). Syntactic and semantic rules based on ontologies and other resources for comparing BPMN models are described, as well as a process for transforming BPMN models into UML model.**

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

I. Introduction

As organizations increase in number, competition between them intensifies. In order to compete, organizations adopt innovative strategies, seek high quality human resources, follow best practices, and use the most efficient technological tools. Developing an efficient and cost-effective information system is crucial for an organization's ability to compete. For this reason, alignment is vital for organizations. Indeed, alignment can provide solutions to problems associated with the diverse changes that may occur in an organization's entities. Business/IT alignment has been the topic of several previous studies [1]–[5]. The approach proposed in this paper is relevant in various situations. An information system with aligned levels may experience changes in one of its levels due to a revision of goals or other factors. This causes the levels to become misaligned. In addition, different levels of an organization's information system may be modelled by different teams. Each team perceives the system from a different perspective, which can also result in misaligned levels. Similarly, when two organizations with levels of different natures merge, the levels of the resulting information system will likely be misaligned as well. In this case, the resulting information system will contain misaligned levels. Further, most organizations include several BPMN models at the business process level, all built independently of each other. This can cause conflicts during alignment, due to heterogeneities between the models. In fact, existing approaches considering a set of BPMN models align models syntactically. However, they only test identity. They do not detect other correspondences, such as inclusions, abbreviations and acronyms. Moreover, none of these studies takes semantic aspects into account (synonymy, homonymy or hyponymy). These heterogeneities can cause conflicts when models are aligned and may introduce problems and inconsistencies in the resulting UML model. Indeed, if alignment approaches based on identity only consider two BPMN elements to be different when they are equivalent, they will introduce a false difference and, therefore, redundancy in the UML model. Worse, considering two elements as equivalent when they are different will introduce a false equivalence, resulting in information loss in the resulting UML model. A generalisation relationship between hyponymic elements is also missing in the UML model.

In all these situations, an effective alignment approach is required to obtain a successful information system. Additionally, it is necessary to ensure that alignment is achieved without conflicts (i.e. without loss or redundant information).

The different approaches described in previous studies concerning business process and software system levels are analysed in this paper. Following that analysis, we propose an alignment approach that resolves syntactic and semantic conflicts. The first step of the proposed approach includes a set of rules for comparing BPMN models to detect equivalencies and differences as well as a transformation process for converting a set of BPMN models into a UML class diagram [6]. The second step provides a method for preserving software system level information.

This paper is organised as follows: Section II presents the background of the topic; it introduces the concept of alignment and ontology. Section III provides a brief overview of related work, while the proposed approach is presented in Section IV. Section V presents a case study. Finally, the conclusion outlines our objectives for future work.

II. Background

A. The Concept of Alignment

In the literature, various expressions have been used to describe the term alignment. Chan [7] refers to alignment by fit and synergy. Henderson and Venkatraman [8] use the terms fit, integration, and interrelationships. Reich and Benbasat [9] use the word linkage. For Ciborra [10], alignment is defined as a bridge. Smaczny [11] describes it as fusion. Luftman [12] employs the term harmony, while Nickels [13] uses congruence. Alignment between business and IT is defined by Ullah and Lai [5], as “the optimized synchronization between dynamic business objectives/processes and respective technological services provided by IT”. According to Luftman [12], business-IT alignment concerns the application of IT in a timely and a suitable manner, in harmony with business strategies, goals and needs. For Luftman [12], this definition considers the way IT is aligned with the business, and the way the business should or could be aligned with IT. In present work, we define alignment of a target level with a source level as a method that ensures the continued operation of the target level, while remaining suitable to the source level.

B. Ontology

Ontology is defined as an “explicit specification of a conceptualization” [14]. Domain ontologies are ontologies built on a particular knowledge domain. There are many domain ontologies such as MENELAS (in the medical domain) [15] and TOVE (in the business management domain) [16]. The domain ontology is a semantically rich model (it can express equivalence, inverse, disjunction, symmetry, transitivity, etc.), and is defined as an exhaustive list of concepts (ontology class) and relations between these concepts describing a particular domain (e.g. Medicine, Business, E-Government).

III. Related Work

In a previous work [17], we proposed a pattern system as a guideline, to help organizations apply the alignment. In addition, a systematic literature review was conducted [18] to present various approaches to the alignment of business requirement, business process and software system levels, that use different modelling languages.

In this study, we focus on UML and BPMN languages, because they are standards defined by the Object Management Group (OMG). More precisely, 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. [19] propose an MDA approach for transforming a BPMN source model into a UML class diagram, using a set of transformation rules. Brdjanin et al. [20] present an approach for the automated generation of a conceptual database model represented by a UML class diagram from a single BPMN model. Brdjanin et al. [21] take a set of business process models into account. Khlif et al. [22] describe an approach to transform a business process model into a class diagram, based on aspects descriptions. Rhazali et al. [23] suggest a set of rules for transforming a BPMN model into a use case, state and class diagrams. Cruz et al. [24] propose an approach to obtain a data model from a business process model. Cruz et al. [25] present rules to transform a set of business process models into a data model. Kriouile et al. [26] describe an approach to transform a BPMN model into a domain class model. Bousetta et al. [27] propose an approach for building a domain class diagram based on a BPMN model, using a set of business rules.

In organizations, models of both levels usually exist. An analysis of existing approaches makes it clear that all 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, these approaches can cause a loss of information. Fig. 1 presents the result of applying one of the existing approaches. M1 represents the business process level model, while M2 represents the software system level model.

Let X, Y and Z be three models belonging to business process level (M1). X contains elements A, F, K and I. Y contains elements B and J, while Z contains elements G and H. Model M2 contains elements D and E. The existing approaches can generate a new UML class diagram (M2’), containing T(A), T(F), T(K), T(I), T(B), T(G) and T(H), which represent the results of the transformation of elements A, F, K, I, B, G and H respectively.

We note that M2’ differs from model M2. Therefore, information D and E associated with the existing UML class diagram, will be lost after alignment (TABLE I, column 2). None of the existing approaches consider all BPMN elements frequently used in organizations (complete metamodel), which provide a detailed description of the models and which belong to the latest BPMN 2.0.2 specification [28], such as all task types or all data types (TABLE I, column 3). The majority of approaches take a single model at the source level into consideration. Only two existing approaches ([21] and [25]) have achieved transformation using a set of BPMN models as a source (TABLE I, column 4).

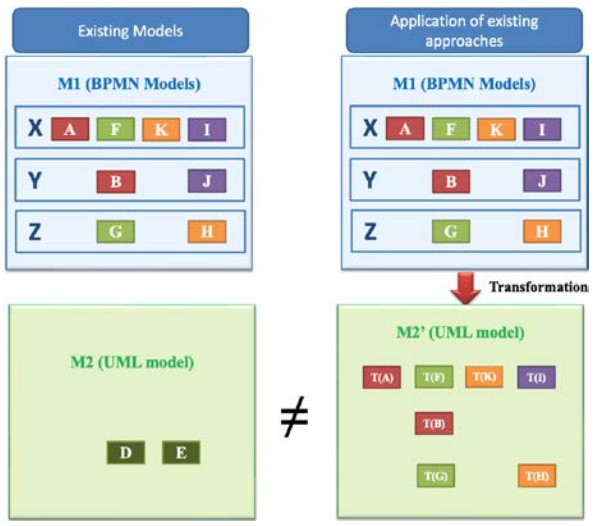

Fig. 1. Application of Existing Transformation Approaches.

TABLE I. SYNTHESIS OF APPROACHES

Ref.	Preserving information	Complete BPMN MM	Set of BPMN models	Syntactic aspect				Semantic aspect		
				Id	Inc	Ab	Ac	S	Hp	Hm
[19]	-	-	-	-	-	-	-	-	-	-
[20]	-	-	-	-	-	-	-	-	-	-
[21]	-	-	x	x	-	-	-	-	-	-
[22]	-	-	-	-	-	-	-	-	-	-
[23]	-	-	-	-	-	-	-	-	-	-
[24]	-	-	-	-	-	-	-	-	-	-
[25]	-	-	x	x	-	-	-	-	-	-
[26]	-	-	-	-	-	-	-	-	-	-
[27]	-	-	-	-	-	-	-	-	-	-

Because models X, Y and Z are usually created independently, they include many types of heterogeneity. All approaches considering a set of BPMN models align models elements syntactically. However, they only test identity: the string equality of elements. They do not detect other correspondences, such as inclusions (e.g. element A, “Medical Office” in X and element B, “Office” in Y), abbreviations (e.g. “Qty” and “Quantity”) and acronyms (e.g. "UOM" and "Unit Of Measure") (TABLE I, column 5). Moreover, none of these studies take semantic aspects into account (TABLE I, column 6). They do not detect synonymy (e.g. element F, "Doctor" in X and element G, "Medical Practitioner" in Z), homonymy (e.g. element I, "Invoice" (of Patient) in X, and element J "Invoice" (of Supplier) in Y), or hyponymy (a semantic relationship between terms where the meaning of one is included in another, more general term) (e.g. element K, “Patient” in X and element H, “Diabetic” in Z).

As shown in Fig. 1, these heterogeneities can cause conflicts when models are aligned and may introduce problems and inconsistencies in the resulting UML model. Indeed, if alignment approaches based on identity only consider two BPMN elements (belonging to X and Y) to be different when they are equivalent, they will introduce a false difference and therefore redundancy in the UML model. For example, we can find both T(A) “Medical Office” and T(B) “Office” in the UML model. Worse, considering two elements (belonging to X and Y) as equivalent when they are different will introduce a false equivalence, resulting in information loss in the resulting UML model. For example, only T(I) “Invoice” (of Patient) can be found in the UML model, and not “Invoice” (of Supplier). A generalisation relationship between hyponymic elements is also missing in the UML model. For example, the generalisation relationship between element T(K) “Patient” and element T(H) “Diabetic”.

We synthesize the existing approaches in TABLE I according to the following criteria: preserving information, considering complete BPMN metamodel (MM), considering a set of BPMN models, considering syntactic aspects (we note syntactic comparison, the comparison strings of model elements’ letters. It indicates if the approach detects identity (Id), inclusions (Inc), abbreviations (Ab) and acronyms (Ac)), and considering semantic aspects (we note semantic comparison, the comparison of the meaning associated with the model’s elements). It indicates if the approach detects synonymy (S), hyponymy (Hp) and homonymy (Hm). In TABLE I, “**x”** indicates that a criterion is considered.

This analysis of existing approaches reveals the need for an alignment approach that preserves existing information, uses a set of BPMN models and considers most used BPMN elements. Moreover, our objective is to propose how to find real equivalences and real differences and how to get UML result model without conflicts.

IV. PROPOSED APPROACH

A. Overview of the Proposed Approach

The aim of the proposed approach is to align the software system level with the business process level, without losing information and without conflict.

We present an alignment system in Fig. 2 that takes a set of BPMN models as input and outputs the resulting UML model. It encompasses two steps:

1) Step 1: Comparison and transformation

a) Comparison of BPMN models: Our goal is to provide a semantic comparison approach that also integrates syntactic aspects. The model comparison subsystem takes BPMN models as input and returns (1) a comparison table containing the correspondence between elements (equivalent, different and hyponymic elements) and (2) isolated elements (those without equivalent, homonym or hyponym in another model). It is a syntactic and semantic rules-based system (presented in section IV.C.1), driven by a comparison process. We use strategies based on semantic properties to take semantic aspects into account. Therefore, our system refers to a domain ontology that will provide semantically relevant information for decision-making during the comparison (example: two ontology classes are semantically equivalent, two elements are hyponyms presented by an ontology class and its ontology subclass, etc.).

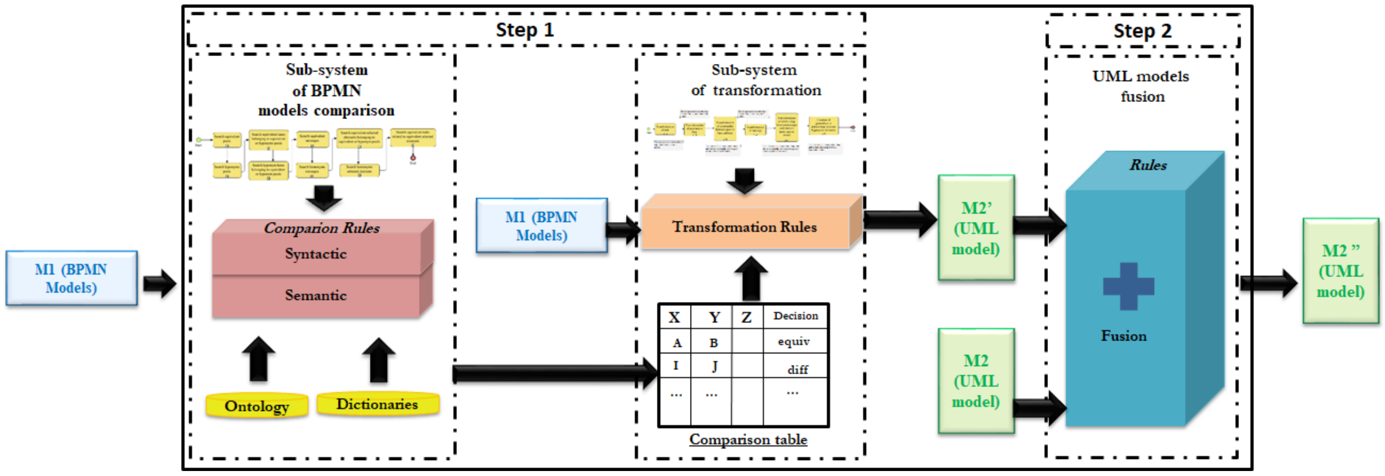

Fig. 2. Representation of Our Alignment Approach.

In addition, our system relies on several other resources to complete these comparisons. We use an acronym dictionary[1], an abbreviation dictionary[2], and a dictionary of synonyms[3].

b) Transformation of BPMN models into a UML model: This subsystem consists of the application of rules to transform a set of BPMN models (M1) into a generated UML class diagram (M2'), based on a comparison table and following a transformation process. Transformation rules based on MDA are presented in our previous work [29]. In the present work, we update the transformation process from BPMN models into a UML model, considering syntactic and semantic aspects (presented in section IV.C.2)).

By applying the rules of syntactic and semantic comparison, we can detect real equivalences and real differences, and thus obtain a UML result model without conflicts (Fig. 3). By identifying F and G as semantically equivalent, we are left with only one element, T(F), and therefore do not have redundancy in UML diagram M2'. Additionally, identifying A and B as syntactically equivalent results in only one element T(A), and thus no redundancy in M2'. Identifying I and J as homonyms produces two result elements, T(I) and T(J), in M2'. Finally, identifying K and H as hyponyms produces a generalisation relationship between the two resulting elements T(K) and T(H) in M2'.

2) Step 2: Fusion: This step consists of creating a fusion between the UML class diagram (M2') generated in step 1 and the existing UML class diagram (M2). We have previously demonstrated the results of this phase; the comparison and fusion meta-models were published in [30]. This solution also takes into account the syntactic and semantic aspects of the two class diagrams elements.

The result is a final UML class diagram (M2"). By applying the two steps illustrated in Fig. 3, the target level is completed, as it contains the information related to the existing class diagram (M2) as well as the information related to the generated class diagram (M2').

[1] https://www.dictionary.com/e/acronyms/
[2] http://theleme.enc.sorbonne.fr/dico.php
[3] http://wordnet.princeton.edu/

B. Example of BPMN Models

This approach can be applied to several BPMN models such us collaboration or process diagrams. In this section, we present a set of BPMN collaboration diagrams. To illustrate our approach, we use an example representing the business process level of a medical field.

A collaboration diagram can contain several elements: pool, lane, event, task, gateway, message flow, message, sequence flow, data, data association, artifact and association. A pool can refer to a process or can be a black box. A lane is a sub-partition within a pool; it can contain a set of events (facts that occur during the process), tasks (atomic work performed in a process), or gateways (controlling the convergence or divergence of flows in a process). A task can appear as a send task, a receive task, a service task, a user task, a manual task, a business rule task, or a script task. A message flow may contain a message (in the present work, we suppose that a message flow contains a message), and links source and target elements (pools, events, or tasks). A sequence flow connects a source and a target element (events, tasks, or gateways). Data provides information about what tasks need to be performed and/or what they produce. There are four types of data: data object, data store, data input or data output. A data association links source data or target data to a task. An artifact can be in the form of a group or a text annotation. It aims to provide more clarity to the process. An association links an artifact with a BPMN element.

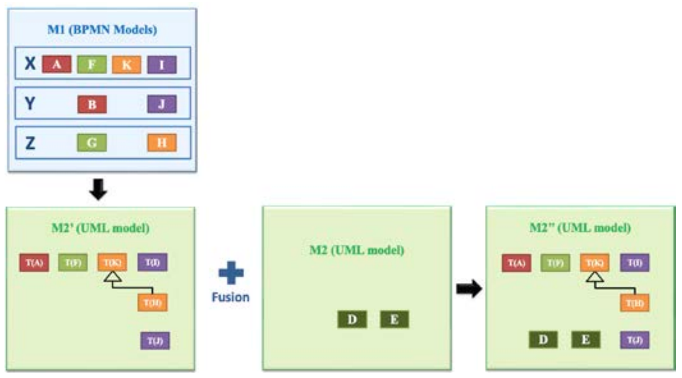

Fig. 3. Result Model without Conflicts or Information Loss.

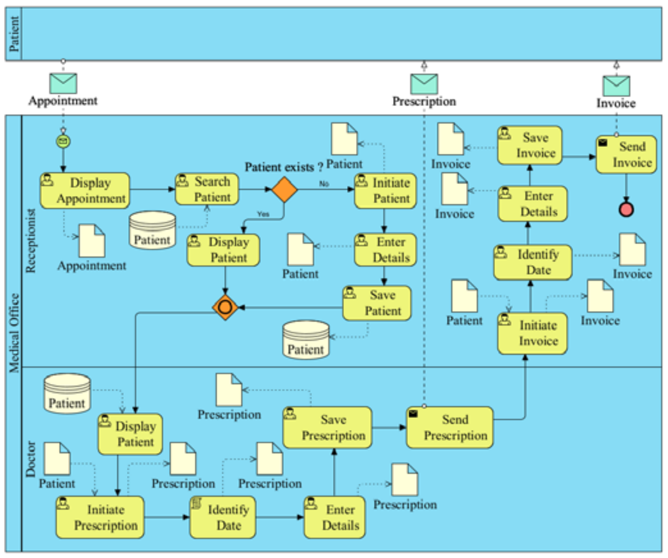

Fig. 4. Model X - Patient Consultation in a Medical Office.

Fig. 4 can illustrate the collaboration diagram (model X) for a patient consultation in a medical office. It is composed of two pools: “Patient”, which is a black box and “Medical Office”, which contains the lanes “Receptionist” and “Doctor”. The diagram begins when a patient arrives at the medical office. The receptionist performs the first task, “Display Appointment”, which has a data object “Appointment” as output. The receptionist then searches for the patient in a patient list. If a patient file exists, the receptionist displays the patient’s information. If not, the receptionist initiates the patient file, adds details and saves the file. The doctor then displays the patient file and initiates a prescription. The date of the prescription is identified. Then, the doctor enters details, saves the file, and sends the prescription. Next, the receptionist initiates an invoice, and its date is identified. The receptionist then enters details, saves the file, and sends the invoice.

Fig. 5 can represent the second collaboration diagram (model Y) placing orders with suppliers. It contains two pools: “Supplier”, represented as a black box, and “Office” which contains two lanes (“Assistant” and “Doctor”). The first task is “Initiate Purchase Order” performed by the assistant. It has as an output the data object “Purchase Order”. Then the date of the purchase order is identified. Next, the assistant adds details, and saves the purchase order. Then, the doctor displays, modifies and validates the purchase order. Next, the assistant sends the purchase order and then receives the invoice from the supplier.

Fig. 6 can illustrate the third collaboration diagram (model Z), for monitoring patients with diabetes. It contains two pools: “Diabetic”, represented as a black box, and “Medical Office”, which contains two lanes (“Assistant” and “Medical Practitioner”). The first task is “Search Patient File” executed by the assistant. Its input is the data store “Patient File”. If the patient is diabetic, the patient file is exposed. If the patient needs an appointment, the assistant initiates the appointment, adds details, saves and sends the appointment to the diabetic. Then the medical practitioner exposes the appointment.

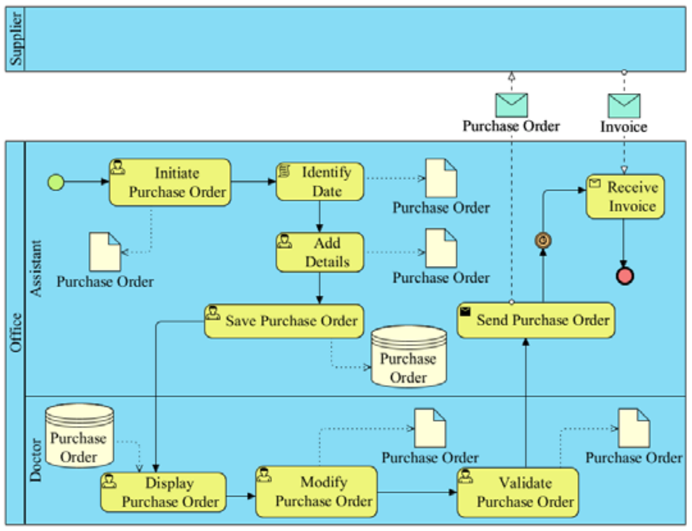

Fig. 5. Model Y - Ordering from Suppliers.

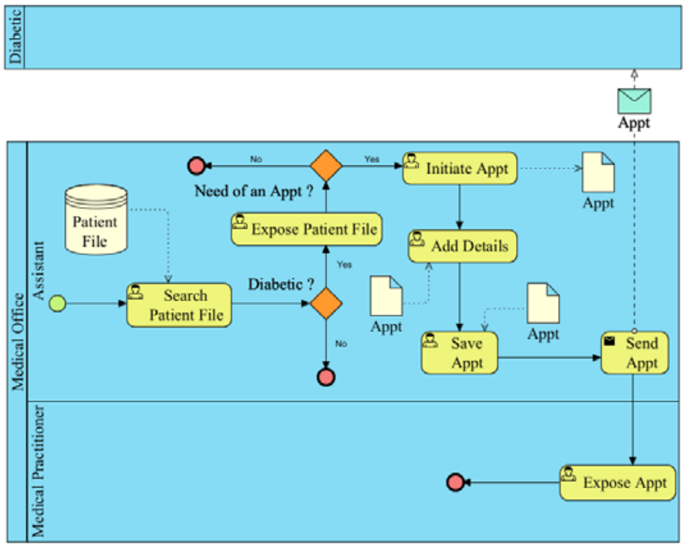

Fig. 6. Model Z - Monitoring Diabetics.

C. Comparison and Transformation Subsystems

In this section, we present the details of our approach. We first present the comparison of BPMN models, then the transformation of those BPMN models into a UML class diagram.

1) Comparison of BPMN models: Our goal is to compare BPMN models syntactically and semantically. To do so, we follow the model comparison process detailed in Section IV.C.1)a), which applies the BPMN element comparison rules described in Section IV.C.1)b).

a) Comparison process: We apply a comparison between each two models. To create a UML class diagram, we create classes before their operations. To do this, we first compare BPMN elements that will be transformed into classes: 1) *Pools, 2)* lanes, 3) messages and 4) data of non manual task and direct object (DO) of all types of task (except manual task) without data, send task that have a data in its input and receive task that have a data in its output). We call the fourth category of elements: "selected elements". Next, we compare BPMN elements that will be transformed to operations: tasks related to equivalent selected elements.

The BPMN elements that will be transformed to associations, aggregations, attributes, and multiplicities [presented in Section IV.C.2)] are not concerned by comparison, because they are not named elements and therefore cannot be compared syntactically and semantically. The elements events, gateways, manual tasks, data related to manual tasks, sequence flows, and artifacts are not considered also in this comparison, because they will not appear in the resulting UML model (they do not have an equivalent in the UML class diagram).

In order to compare BPMN models, we follow the steps of the comparison process (Fig. 7), in this order: search equivalent pools, search hyponym pools, search equivalent lanes belonging to equivalent or hyponym pools, search hyponym lanes belonging to equivalent or hyponym pools, search equivalent messages, search homonym messages, search equivalent selected elements belonging to equivalent or hyponym pools, search homonym selected elements, and finally search equivalent tasks related to equivalent selected elements.

Pools are not concerned with homonymy because BPMN models are designed for the same domain. For example, if "office" is found in the first model meaning medical office, and "office" is also found in the second model, it will refer to a medical office there as well, and not, for example, a lawyer's office. The same is true for lanes: two homonymic lanes cannot belong to equivalent pools.

b) Comparison rules: We define a mathematical framework to express formally our approach. We present the comparison rules by the predicate language. So, we needed to express, in predicate logic, BPMN model (representing input of system), ontology and other resources (representing the system references). For that, we realised transformations, in the Model Driven Architecture (MDA) context:

i) The first transformation concerns BPMN model into logical model that generates a set of predicates representing BPMN elements to compare:

- Element(e,M): The element Pool or Lane "e" belonging to the model "M".
- Pool(P,M): The pool "P" belonging to the model "M".
- Lane (L,P,M) : The Lane "L" belonging to the pool "P"and the model "M".
- Message (m, SourceP, TargetP,M): The message "m" relating the source Pool "SourceP" and the target Pool "TargetP", belonging to the model "M".
- SelectedElement(E, P, M) : The selected element "E", belonging to the pool "P" and the model "M".
- Task (T, in, out, L, P, M) : The task "T", with the input data "in", and the output data "out", belonging to the Lane "L", the pool "P" and the model "M".

ii) The second transformation concerns OWL ontology (conform to an extract of OWL metamodel [31]) into logical model. The transformation generates a set of predicates representing OWL ontology, such as:

- equivalentOntoClass(C1, C2): The ontology classes "C1" and "C2" are equivalent.
- OntoSuperClassOf(C1, C2): The ontology class "C1" is subclass of the ontology class "C2".

Other system references are presented as follows:

- DicAcronyms(elt1,elt2): "elt1" and "elt2" are acronyms.
- DicAbbreviation(elt1,elt2) :"elt1" and "elt2" are on abbreviation relation.
- DicSynonymy(elt1,elt2) :"elt1" and "elt2" are synonyms.

To complete rules expression, we define a set of facts from programming languages, such as equality of two strings, and inclusion of two strings:

- String(elt) : "elt" is a character string.
- InclusionString(s1,s2) : means that the character string "s1" is included in the character string "s2".
- EqualString(s1,s2): The two strings "s1" and "s2" are equal.

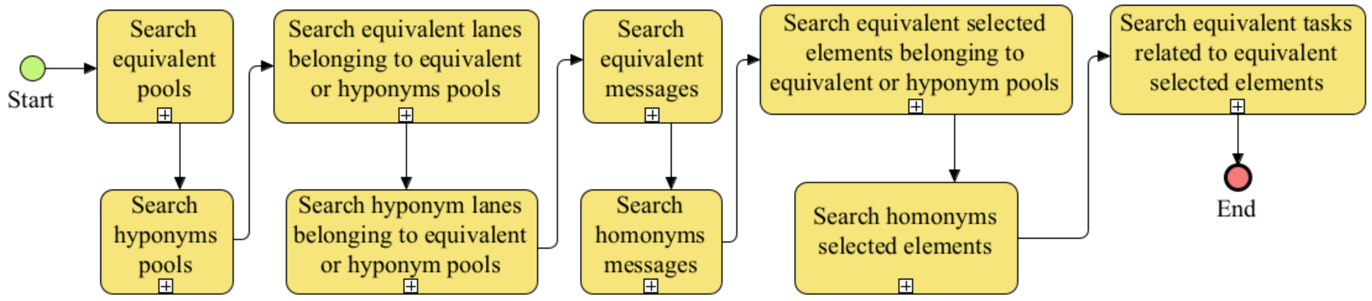
Start
Search equivalent pools
Search hyponyms pools
Search equivalent lanes belonging to equivalent or hyponyms pools
Search hyponym lanes belonging to equivalent or hyponym pools
Search equivalent messages
Search homonyms messages
Search equivalent selected elements belonging to equivalent or hyponym pools
Search homonyms selected elements
Search equivalent tasks related to equivalent selected elements
End

Fig. 7. Comparison Process.

Our system is based on a set of comparison rules which are characterized by a name and it is composed of a set of parameters (e.g. elements to compare, first model and second model) that have a name. A rule can call one or more other rules.

We present the comparison rules of models below.

- CR1 : Rule of syntactic identity of elements

$$\text{Syntactic\_Identity}(\text{elt}_i, \text{elt}_j) \Leftrightarrow \text{EqualString}(\text{elt}_i, \text{elt}_j)$$

Two elements elt_i and elt_j are syntactically identical if and only if they have the same name.

Example: The pool “Medical office” in X and the pool “Medical office” in Z.

- CR2: Rule of syntactic equivalence of elements

$$\begin{array}{l}\text{Equivalence\_Syntactic\_Elts}(\text{elt}_i, \text{elt}_j) \Leftrightarrow [\text{Syntactic\_Identity}(\text{elt}_i, \text{elt}_j) \vee \\ \text{InclusionString}(\text{elt}_i, \text{elt}_j) \vee \text{InclusionString}(\text{elt}_j, \text{elt}_i) \vee \\ \text{DicAcronyms}(\text{elt}_i, \text{elt}_j) \vee \text{DicAbbreviation}(\text{elt}_i, \text{elt}_j)]\end{array}$$

Two elements elt_i and elt_j are syntactically equivalent if and only if they are identical, or if there is a relationship of inclusion, acronym or abbreviation of them (according to dictionaries).

Example: The pools “Medical Office” in X and “Office” in Y.

- CR3: Rule of semantic equivalence of elements

$$\text{Equivalence\_Semantic\_Elements}(\text{elt}_i, \text{elt}_j) \Leftrightarrow$$

$$\begin{array}{l}[(\text{DicSynonymy}(\text{elt}_i, \text{elt}_j) \vee (\exists e\ \text{String}(e) \wedge \\ \text{Equivalence\_Syntactic\_Elts}(\text{elt}_i, e) \wedge \text{DicSynonymy}(e, \text{elt}_j)) \vee \\ (\exists c\ \text{String}(c) \wedge \text{Equivalence\_Syntactic\_Elts}(\text{elt}_j, c) \wedge \\ \text{DicSynonymy}(c, \text{elt}_i)) \vee (\exists e\ \text{String}(e) \wedge \exists c\ \text{String}(c) \wedge \\ \text{Equivalence\_Syntactic\_Elts}(\text{elt}_j, e) \wedge \text{Equivalence\_Syntactic\_Elts}(\text{elt}_i, c) \wedge \\ \text{DicSynonymy}(e, c)))\end{array}$$

$$\begin{array}{l}\vee (\text{equivalentOntoClass}(\text{elt}_i, \text{elt}_j) \vee (\exists e\ \text{String}(e) \wedge \\ \text{Equivalence\_Syntactic\_Elts}(\text{elt}_i, e) \wedge \text{equivalentOntoClass}(e, \text{elt})) \vee \\ (\exists c\ \text{String}(c) \wedge \text{Equivalence\_Syntactic\_Elts}(\text{elt}_j, c) \wedge \\ \text{equivalentOntoClass}(c, \text{elt}_i)) \vee (\exists e\ \text{String}(e) \wedge \exists c\ \text{String}(c) \wedge \\ \text{Equivalence\_Syntactic\_Elts}(\text{elt}_j, e) \wedge \text{Equivalence\_Syntactic\_Elts}(\text{elt}_i, c) \wedge \\ \text{equivalentOntoClass}(e, c)))]\end{array}$$

Two elements elt_i and elt_j are semantically equivalent if and only if one of these conditions is satisfied:

- elt_i (or a character string syntactically equivalent to elt_i) is synonymous with elt_j (or a character string syntactically equivalent to e_j), according to a synonym dictionary.
- there are two classes in the domain ontology with the same names of elt_i and elt_j (or a character string syntactically equivalent to elt_i and elt_j), which are equivalent.

Example: The lanes “Receptionist” in X and “Assistant” in Y.

- CR4: Rule of hyponyms of elements (Pool or Lane)

$$\begin{array}{l}\text{Hyponym\_Elements}(\text{elt}_i, \text{elt}_j, M_i, M_j) \Leftrightarrow [\text{Element}(\text{elt}_i, M_i) \wedge \\ \text{Element}(\text{elt}_j, M2) \wedge (\text{OntoSuperClassOf}(\text{elt}_i, \text{elt}_j) \vee (\exists e\ \text{String}(e) \wedge \\ \text{Equivalence\_Syntactic\_Elts}(\text{elt}_i, e) \wedge \text{OntoSuperClassOf}(e, \text{elt}_j)) \vee \\ (\exists c\ \text{String}(c) \wedge \text{Equivalence\_Syntactic\_Elts}(\text{elt}_j, c) \wedge \\ \text{OntoSuperClassOf}(\text{elt}_i, c)) \vee (\exists e\ \text{String}(e) \wedge \exists c\ \text{String}(c) \wedge \\ \text{Equivalence\_Syntactic\_Elts}(\text{elt}_i, e) \wedge \text{Equivalence\_Syntactic\_Elts}(\text{elt}_j, c) \wedge \\ \text{OntoSuperClassOf}(e, c)))]\end{array}$$

An element (Pool or Lane) elt_i is a hyponym of an element elt_j if and only if, in a domain ontology, one ontology class with the same name of elt_i (or a character string syntactically equivalent to elt_i) is an ontology subclass of elt_j (or a character string syntactically equivalent to elt_j).

Example: The pool “Diabetic” in Z is a hyponym of the pool “Patient” in X.

- CR5: Rule of Equivalent Pools

$$\begin{array}{l}\text{Equivalence\_Pools}(P_i, P_j, M_i, M_j) \Leftrightarrow [\text{Pool}(P_i, M_i) \wedge \text{Pool}(P_j, M_j) \wedge \\ (\text{Equivalence\_Syntactic\_Elts}(P_i, P_j) \vee \\ \text{Equivalence\_Semantic\_Elements}(P_i, P_j))]\end{array}$$

Two pools P_i and P_j are equivalent if and only if they are syntactically or semantically equivalent.

Example: The pool “Medical Office” in X and the pool “Medical Office” in Z.

- CR6: Rule of Equivalent lanes

$$\begin{array}{l}\text{Equivalence\_Lanes}(L_i, L_j, M_i, M_j) \Leftrightarrow [\text{Lane}(L_i, P_i, M_i) \wedge \text{Lane}(L_j, P_j, M_j) \wedge \\ (\text{Equivalence\_Syntactic\_Elts}(L_i, L_j) \vee \\ \text{Equivalence\_Semantic\_Elements}(L_i, L_j))]\end{array}$$

Two lanes L_i and L_j are equivalent if and only if they are syntactically or semantically equivalent.

Example: The lane “Doctor” belonging to the pool “Medical Office” in X and the lane “Medical Practitioner” belonging to the pool “Medical Office” in Z.

- CR7: Rule of Equivalent messages

$$\begin{array}{l}\text{Equivalence\_Messages}(m_i, m_j, M_i, M_j) \Leftrightarrow \\ [\text{Message}(m_i\ \text{SourceP}_i, \text{TargetP}_i, M_i) \wedge \\ \text{Message}(m_j, \text{SourceP}_j, \text{TargetP}_j, M_j) \wedge \\ (\text{Equivalence\_Syntactic\_Elts}(m_i, m_j) \vee \\ \text{Equivalence\_Semantic\_Elements}(m_i, m_j)) \wedge \\ ((\text{Equivalence\_Pools}(\text{SourceP}_i, \text{SourceP}_j, M_i, M_j) \vee \\ \text{Equivalence\_Pools}(\text{SourceP}_i, \text{TargetP}_j, M_i, M_j) \vee \\ (\text{Hymonym\_Elements}(\text{SourceP}_i, \text{SourceP}_j, M_i, M_j) \vee \\ \text{Hymonym\_Elements}(\text{SourceP}_j, \text{SourceP}_i, M_i, M_j)) \vee \\ (\text{Hymonym\_Elements}(\text{SourceP}_i, \text{TargetP}_j, M_i, M_j) \vee \\ \text{Hymonym\_Elements}(\text{TargetP}_j, \text{SourceP}_i, M_i, M_j))) \wedge \\ (\text{Equivalence\_Pools}(\text{TargetP}_i, \text{SourceP}_j, M_i, M_j) \vee \\ \text{Equivalence\_Pools}(\text{TargetP}_i, \text{TargetP}_j, M_i, M_j) \vee \\ (\text{Hymonym\_Elements}(\text{TargetP}_i, \text{SourceP}_j, M_i, M_j) \vee \\ \text{Hymonym\_Elements}(\text{SourceP}_j, \text{TargetP}_i, M_i, M_j)) \vee \\ (\text{Hymonym\_Elements}(\text{TargetP}_i, \text{TargetP}_j, M_i, M_j) \vee \\ \text{Hymonym\_Elements}(\text{TargetP}_j, \text{TargetP}_i, M_i, M_j))))]\end{array}$$

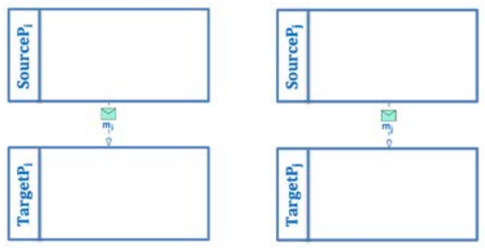

Fig. 8. Representation of Two Messages Mi and Mj.

Two messages m_i and mj (Fig. 8) are equivalent if and only if:

- they are syntactically or semantically equivalent.
- and their source and target pools are equivalent or hyponyms.

Example: The message “Appointment” connecting the pools “Medical Office” and “Patient” in X and the message “Appt” connecting the pools “Medical Office” and “Diabetic” in Z.

- CR8: Rule of homonym message

$$\begin{aligned}
&\text{Homonym\_Messages}(m_i, m_j, M_i, M_j) \\
&\quad \Leftrightarrow [\text{Message}(m_i\ \text{SourceP}_i, \text{TargetP}_i, M_i) \\
&\quad \wedge \text{Message}(m_j, \text{SourceP}_j, \text{TargetP}_j, M_j) \\
&\quad \wedge \text{Equivalence\_Syntactic\_Elts}(m_i, m_j) \\
&\quad \wedge (\neg\text{Equivalence\_Pools}(\text{SourceP}_i, \text{SourceP}_j, M_i, M_j) \\
&\quad \vee \neg\text{Equivalence\_Pools}(\text{SourceP}_i, \text{TargetP}_j, M_i, M_j) \\
&\quad \vee \neg\text{Equivalence\_Pools}(\text{TargetP}_i, \text{SourceP}_j, M_i, M_j) \\
&\quad \vee \neg\text{Equivalence\_Pools}(\text{TargetP}_i, \text{TargetP}_j, M_i, M_j))]
\end{aligned}$$

Two messages m_i and m_j (Fig. 8) are homonyms if and only if:

- they are syntactically equivalent.
- if at least one of their source or target pools are not equivalent.

Example: The message “Invoice” in X and the message “Invoice” in Y.

- CR9: Rule of equivalent selected elements.

$$\begin{aligned}
&\text{Equivalence\_SelectetElements}(e_i, e_j, M_i, M_j) \ \Leftrightarrow \\
&[\text{SelectedElement}(e_i, P_i, M_i) \wedge \text{SelectedElement}(e_j, P_j, M_j) \wedge \\
&(\text{Equivalence\_Syntactic\_Elts}(e_i, e_j) \vee \\
&\text{Equivalence\_Semantic\_Elements}(e_i, e_j)) \wedge \\
&(\nexists m_i \text{Message}(m_i, \text{SourceP}_i, \text{TargetP}_i, M_i) \wedge \\
&\text{Equivalence\_Syntactic\_Elts}(e_i\ , m_i)\ \wedge \\
&\nexists m_j \text{Message}(m_j, \text{SourceP}_j, \text{TargetP}_j, M_j) \wedge \\
&\text{Equivalence\_Syntactic\_Elts}(e_j\ , m_j) \wedge \\
&\text{Homonym\_Messages}(m_i, m_j, M1, M2))]
\end{aligned}$$

Two selected elements e_i and e_j are equivalent if and only if:

- they are syntactically or semantically equivalent.
- and there is no homonyms messages with the same name of selected elements.

Examples: The selected element “Patient” in X and the selected element “Patient File” are equivalent.

- CR10: rule of Homonyms selected elements

$$\begin{aligned}
&\text{Homonym\_SelectetElements}(e_i, e_j, M_i, M_j) \ \Leftrightarrow \\
&[\text{SelectedElement}(e_i, P_i, M_i) \wedge \text{SelectedElement}(e_j, P_j, M_j) \wedge \\
&((\exists m_i \text{Message}(m_i, \text{SourceP}_i, \text{TargetP}_i, M_i) \wedge \\
&\text{Equivalence\_Syntactic\_Elts}(e_i\ , m_i)\ \wedge \\
&\exists m_i \text{Message}(m_j, \text{SourceP}_j, \text{TargetP}_j, M_j) \wedge \\
&\text{Equivalence\_Syntactic\_Elts}(e_j\ , m_j) \wedge \\
&\text{Homonym\_Messages}(m_i, m_j, M_i, M_j)) \vee \\
&((\text{Equivalence\_Syntactic\_Elts}(e_i, e_j) \wedge \\
&(\neg\text{Equivalence\_Pools}(P_i, P_j, M_i, M_j)) \vee \\
&\neg\text{Hymonym\_Elements}(P_i, P_j, M_i, M_j))]
\end{aligned}$$

Two selected elements e_i and e_j are homonyms if and only if:

- there are homonyms messages with the same name of selected elements.
- or, they are syntactically equivalent and belong to non-equivalent and non-hyponym pools.

Example: The selected element “Invoice” in X and the selected element “Invoice” in Y.

- CR11: Rule of Equivalent tasks

$$\begin{aligned}
&\text{Equivalence\_Tasks}(T_j, T_j, M_i, M_j) \Leftrightarrow [\text{Task}(T_i, \text{type}_{ti}, \text{dIN}_i, \text{dOUT}_i, L_i, P_i, M_i) \\
&\wedge \text{Task}(T_j, \text{type}_{tj}, \text{dIN}_j, \text{dOUT}_j, L_j, P_j, M_j) \\
&\qquad \wedge (\text{Equivalence\_Syntactic\_Elts}(T_i, T_j)) \\
&\qquad \vee \text{Equivalence\_Semantic\_Elements}(T_i, T_j)]]
\end{aligned}$$

Two tasks T_i and T_j are equivalent if and only if are syntactically or semantically equivalent.

Example: The task “Display Appointment” in X and the task “Display Appt” in Z.

c) Comparison table: We apply the process and the comparison rules on the examples of BPMN models presented in section IV.B, by referring to an ontology of the medical field as well as dictionaries. We choose OWL (Ontology Web language) ontology because it is a W3C[4] recommendation, and the metamodel OWL was defined by Ontology Definition Metamodel specification of OMG. To compare these models, we can find in ontology of medical domain several information. For a reason of space, we present bellow two information: Two equivalent ontology OWL classes, and the ontology OWL class “Patient” and its sub class “Diabetic”:

```
<owl:Class rdf:ID="Doctor">

<owl:equivalentClass rdf:resource="#Medical Practionnar"/>

   </owl:Class>

<owl:Class rdf:ID="Diabetic">

      <rdfs:subClassOf rdf:resource="#Patient"/>

</owl:Class>
```

[4] www.w3.org

TABLE II. COMPARISON TABLE

	X	Y	Z	Equiv syn/sem	Hypo	Hom	Decision	RCX	RCY	RCZ
Pools	Medical Office	Office	Medical Office	Yes	-	-	Equivalent	Medical Office	Medical Office	Medical Office
	Patient	-	Diabetic	-	Yes	-	Hyponymy	Patient	Diabetic	-
Lanes	Receptionist	Assistant	Assistant	Yes	-	-	Equivalent	Receptionist	Receptionist	Receptionist
	Doctor	Doctor	Medical Practitioner	Yes	-	-	Equivalent	Doctor	Doctor	Doctor
Messages	Appointment	-	Appt	Yes	-	No	Equivalent	Appointment	Appointment	-
	Invoice	Invoice	-	Yes	-	Yes	Different	Invoice_Patient	Invoice_ Supplier	-
Selected Elements	Invoice	Invoice	-	Yes	-	Yes	Different	Invoice_Patient	Invoice_ Supplier	-
	Appointment	-	Appt	-	-		Equivalent	Appointment	Appointment	-
	Patient	-	Patient File	Yes	-	No	Equivalent	Patient	-	Patient
Tasks	Display Appointment	-	Expose Appt	-	-	-	Equivalent	Display Appointment	-	Display Appointment
	Display Patient	-	Expose Patient File	-	-	-	Equivalent	Display Patient	-	Display Patient
	Search Patient	-	Search Patient File	-	-	-	Equivalent	Search Patient	-	Search Patient

We obtain the comparison table (TABLE II), after the elimination of duplicate elements. The first four columns present the elements of the models X, Y and Z to be compared, the fifth column presents whether or not the elements are syntactically or semantically equivalent. The sixth column shows whether or not the elements are hyponyms. The seventh column presents whether or not the elements are homonyms. The eighth column presents the final decision (equivalence, hyponymy or difference).

In order to choose the appropriate name of UML result elements (in the second subsystem of transformation), we base on the decision of column 8 to rename elements in the three last columns called RCM$_i$ (M$_i$ represents X, Y or Z), as following: (1) If the decision is "Equivalent", we rename the elements of model X, Y and or Z using the name of the first column. (2) If the decision is "Hyponymy", we keep the same name of the elements of the model. (3) If the decision is "Different", we use the name of the element and we add "_the name of the pool that is different to the pool of the other element".

2) Transformation of BPMN models into UML class diagram: In order to transform BPMN models into a UML class diagram we apply a transformation process (Fig. 9) based on the comparison table presented in section IV.C.1)c). This process is based on 18 transformation rules presented in detail in our previous work [29]. This process is presented in the form of a series of steps based on BPMN notation as follow (Fig. 9):

- Transformation of task: this sub-process is based on 12 rules related to tasks (considering their types and if they are linked to data or not) and call one other rule related to data. At the end of this step, the UML class diagram can be constituted of classes, attributes, operations, associations and multiplicities.
- Transformation of pool and/or lane: this sub-process generates classes, attributes, and aggregations after this step.
- Transformation of relationship between (pool or lane) and task: it generates associations and multiplicities.
- Transformation of message: it generates classes and attributes.
- Transformation of relationship between message and element (pool, task or event): it generates associations and multiplicities.

For those first fifth sub process, we:

- Identify an element "i" of a model M$_i$
- Apply the corresponding rule (according to the element identified). If the element belongs to the column M$_i$ in the comparison table, the name of the element (class or operation) that will be used corresponds to the RCM$_i$ column. In addition, a check is performed to determine whether an element (class, operation or association) of the class diagram has already been created by another rule. If it has:
  - o All the instructions associated with that rule are applied, except creation of the element.
  - o 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.

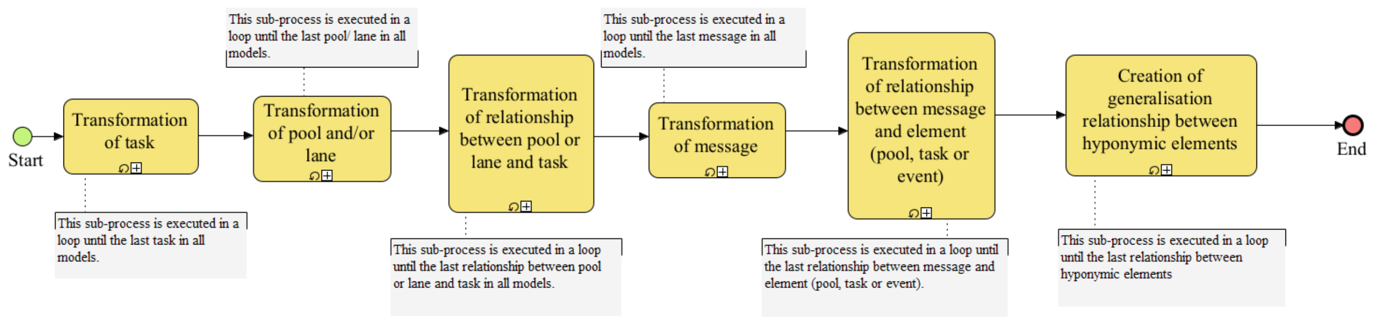
Start
Transformation of task
This sub-process is executed in a loop until the last task in all models.
This sub-process is executed in a loop until the last pool/ lane in all models.
Transformation of pool and/or lane
Transformation of relationship between pool or lane and task
This sub-process is executed in a loop until the last relationship between pool or lane and task in all models.
This sub-process is executed in a loop until the last message in all models.
Transformation of message
Transformation of relationship between message and element (pool, task or event)
This sub-process is executed in a loop until the last relationship between message and element (pool, task or event).
Creation of generalisation relationship between hyponymic elements
This sub-process is executed in a loop until the last relationship between hyponymic elements
End

Fig. 9. The Transformation Process.

- Creation of a generalisation relationship between hyponymic elements. In this sub-process, we:
  - o Identify hyponymic elements.
  - o Create a generalisation relationship between elements. When transforming an element “i” of model Mi, if the element belongs to the column Mi in the comparison table, the name of the element class that will be used corresponds to the RCMi column.

V. Case Study

In order to illustrate the application of the first step of our proposed approach, the three BPMN models represented in section IV.B, are transformed into a UML class diagram (Fig. 10) by applying our transformation process and rules and by referring to the comparison table. Resulted model represents the output of step 1 (M2’ UML model).

We note that there is no redundancy in the classes. Indeed, for example, we find only the class "Medical Office" instead of having in addition the class "Office", only the class "Doctor, instead of having in addition the class "Medical Practitioner", only the class "Receptionist" instead of having in addition another class "Assistant", a single class "Appointment" instead of having in addition the class "Appt". Moreover, we don’t find redundancies in class operations. In fact, for example, we find a single "displayAppointment()" operation, and not a second "exposeAppt()", we also found a single "displayPatient()" operation, and a single "searchPatient()" operation. Also, we don’t find a loss information by finding the two classes "Invoice_Patient" and "Invoice_Supplier". Finally, we find a generalisation relationship between the "Diabetic" and "Patient" classes.

We present the resulted elements of the first step of transformation process (classes, attributes, associations, multiplicities) by black color. At the second step, the resulted elements (classes, attributes, aggregations, multiplicities) are illustrated using blue color. The resulted elements (associations and multiplicities) after the third step are mentioned using green color. After the fourth step, we didn’t add new element because they were created in the previous steps. After the fifth step, the added or modified elements (associations, multiplicities) are illustrated by red color. Finally, we illustrate the resulted generalisation relationship, using yellow color.

To create the result UML model M2", we merge (by applying the fusion subsystem, already developed in our previous works), M2 model which can represent the existing software level in medical field and this M2’ resulted model.

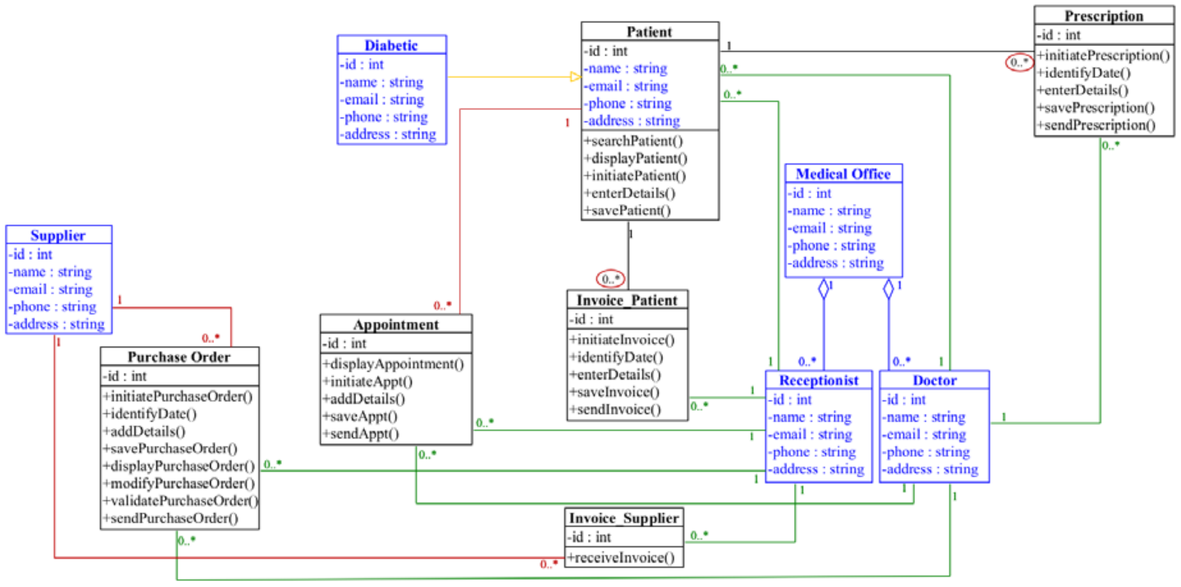

Fig. 10. Resulted Class Diagram UML Model M2’.

VI. Conclusion

In this paper, we propose a method for aligning software system level with business process level without conflicts, by considering a set of models at the source level.

We responded to the limitations of existing work. In fact, by applying the rules of syntactic and semantic comparison, we can detect real equivalences and real differences, and thus obtain a UML result model without conflicts. Indeed, by identifying two elements as semantically equivalent, we are left with only one element, and therefore do not have redundancy in UML diagram; additionally, identifying two elements as syntactically equivalent results in only one element, and thus no redundancy in result model. Identifying two elements as homonyms produces two result elements in UML model. Finally, identifying two elements as hyponyms produces a generalisation relationship between the two resulting elements in result model.

We have implemented a first version of our approach, which allows us to facilitate and automate the transformation phase. This was accomplished using the Java programming language in the Eclipse development environment. The tool takes a set of xpdl files as input. Each file is a representation of a BPMN model. An xmi file is generated from this input, which is later converted into a UML class diagram. It is tested through a case study in the field of telecommunications. Our intent is to update this module by considering syntactic and semantic aspects. We aim also proposing the alignment of BPMN models with UML model to realise a more complete alignment.

The syntactic and semantic comparison of BPMN models resulted in our work can be used in the first step of a research theme, which is the fusion of the BPMN models, in the context of organizations' merge.

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] O. M. G. UML, "OMG (2017) Unified Modeling Language®(OMG UML®) Version 2.5. 1 https://www. omg. org/spec," 2017.

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

[8] 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.

[9] B. H. Reich and I. Benbasat, "Measuring the linkage between business and information technology objectives," MIS Q., pp. 55–81, 1996.

[10] C. U. Ciborra, "Deconstructing the concept of strategic alignment," Scand. J. Inf. Syst., vol. 9, no. 1, pp. 67–82, 1997.

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

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

[13] D. W. Nickels, "IT-Business Alignment: what we know that we still don't know," in Proceedings of the 7th Annual Conference of the Southern Association for Information Systems, vol. 79, pp. 79-84. 2004.

[14] T. R. Gruber, "A translation approach to portable ontology specifications," Knowl. Acquis., vol. 5, no. 2, pp. 199–220, 1993.

[15] P. Zweigenbaum, "MENELAS: an access system for medical records using natural language," Comput. Methods Programs Biomed., vol. 45, no. 1–2, pp. 117–120, 1994.

[16] T. R. Gruber, "Toward principles for the design of ontologies used for knowledge sharing?," Int. J. Hum. Comput. Stud., vol. 43, no. 5–6, pp. 907–928, 1995.

[17] M. Habba, M. Fredj, and S. B. Chaouni, "Towards an operational alignment approach for organizations," in Proceedings of the 9th International Conference on Information Management and Engineering, pp. 29–34, 2017.

[18] M. Habba, M. Fredj, and S. Benabdellah Chaouni, "Alignment between Business Requirement, Business Process, and Software System: A Systematic Literature Review," J. Eng. (United Kingdom), vol. 2019, 2019, doi: 10.1155/2019/6918105.

[19] 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.

[20] 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.

[21] 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.

[22] W. Khlif, N. Elleuch, E. Alotabi, and H. Ben-Abdallah, "Designing BP-IS Aligned Models: An MDA-based Transformation Methodology," pp. 258– 266, 2018.

[23] 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.

[24] E. F. Cruz, R. J. Machado, and M. Y. Santos, "From business process modeling to data model: A systematic approach," in 2012 Eighth International Conference on the Quality of Information and Communications Technology, pp. 205–210, 2012.

[25] 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.

[26] 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.

[27] B. Bousetta, O. El Beggar, and T. Gadi, "A methodology for CIM modelling and its transformation to PIM," Journal of Information Engineering and Applications, vol. 3, no. 2, pp. 1–21, 2013.

[28] B. P. M. OMG, "Notation (BPMN) Version 2.0. 2, Object Management Group, 2013," 2016.

[29] M. Habba, S. B. Chaouni, and M. Fredj, “Aligning Software System Level with Business Process Level through Model-Driven Architecture,” International Journal of Advanced Computer Science and Applications, vol. 12, no. 10, pp. 174–183, 2021, doi: 10.14569/IJACSA.2021.0121020.

[30] S. B. Chaouni, M. Fredj, and S. Mouline, “Metamodels for models complete integration,” in 2011 IEEE International Conference on Information Reuse & Integration, 2011, pp. 496–497.

[31] Ontology Definition Metamodel, OMG Adopted Specification, OMG Document Number: ptc/2008-09-07, 2008.