A Model-driven Architecture for Collaborative Business Processes

Leila Amdah, Naima Essadi, Adil Anwar
Mohammed V University in Rabat
Siweb Team Researsh, EMI
Rabat, Morocco

***Abstract*—The Model Driven Engineering was developed to make application development more flexible, it provides a comprehensive interoperability framework for defining interconnected systems, and aims to reduce the inherent complexity that partners must face when developing their systems. In collaborative environments where systems are made through the collaboration of several departments or companies, the MDA (Model Driven Architecture) approach seems efficient in maintaining and developing this type of system. This paper show the use of MDE in the context of business process management and present in detail an architecture for the development of collaborative business processes.**

Keywords—Model Driven Engineering (MDE); Business Process Model and Notation (BPMN); Business Process Execution Language (BPEL); ATLAS Transformation Language (ATL); BPMN to BPEL Transformation

I. Introduction

Business processes have become increasingly popular in the context of modernization and business reorganization. Business process management has been used primarily at the management level, to document ongoing activities and plan business improvements through reengineering projects [1]. Information Technology (IT) was primarily viewed as an enabling technology that provided adequate support for management tools. During the last years, business process management (BPM) is increasingly integrated into IT projects and workflow engines capable of directly executing business processes have emerged. Today's businesses need to quickly create and change value chains. This leads to continuous growth and change in business processes. The goal of Business Process Management (BPM) is to help people in the business to manage these changes. Business process management is defined as the ability to discover, design, deploy, execute, interact, operate, optimize and analyze the process in a comprehensive manner, doing it at the business design level and not at the technical implementation level. This development in business processes leads to benefits as well as problems in the management and implementation of processes and these problems persist even more when it comes to collaborative business processes. Collaboration requires analysis at the organizational level, changing internal business plans and ultimately finding the most suitable approach for implementation and management.

The modeling of collaborative business processes faces several challenges: The complexity of modeling intensifies during the coupling of business processes because each participant has their own set of "private" models which are modeled with different languages. Employees have heterogeneous application environments, different cultures and different business rules. The interoperability problem persists because every evolution or change in collaborative processes requires a lot of effort. In addition, general-purpose modeling languages, including UML (Unified Modeling Language), are difficult to understand for non-IT specialists, future users of information systems. Even if they read and accept the models, their understanding is not deep enough and they undervalue the consequences of the decisions behind these models. Information systems are not flexible and it is difficult to conform to models. If changes are made directly to the generated code, subsequent regeneration may undo them; faced with these problems in practice. Several research questions are asked but the three main ones are:

Q1: How to make collaborative business processes interoperable in a system.

Q2: How to define a DSML for modeling collaborative business processes.

Q3: How to implement these collaborative business processes.

To answer this question, this paper adopts a model-oriented approach based on Model Driven engineering (MDE). It was developed to make application development more flexible, it offers a comprehensive interoperability framework for defining interconnected systems. It aims to reduce the inherent complexity that partners must bear when developing collaborative processes, and to ensure that a conceptual solution (process level) is consistent with their respective systems (technological level). To support the design of collaborative processes, a specific language, called BPMN4Coll, has been proposed, which model collaborative business processes using very specific artefacts. The implementation of BPMN4Coll model is carried out with a model transformation using the ATL (ATLAS Transformation Language). The models produced are described by the BPEL (Business Process Execution Language). The latter allows the execution of models through process engine. This model separation allows a more flexible generation of applications and the development of user-friendly information systems.

The main advantages of this approach are:

- Increased level of abstraction, main development artifacts are collaborative process models independent of technology.
- Reduced development time and costs because solutions are generated automatically.
- Guarantee of consistency in the technological solutions generated because they are automatically generated from a well-defined transformation procedure: consistency with the processes defined at company level. In addition, the process specifications and interface specifications of the corresponding partners are also consistent.
- Independence of collaborative process models from Business to Business standards, which increases the flexibility of these models.

This paper is organized as follows: the second section explain the combination of MDE and BPM and how business process management can benefit from this approach. Section 3 is related work, it present some work that used the MDE in the modeling and implementation of business processes. Section 4 is devoted to explain this approach. Section 5 is an example to illustrate the transformation from BPMN4Coll to BPEL. Finally the paper is concluded in section 6.

II. Use of MDE in the BPM Context

Business Process Management starts with modeling processes. Process modeling is a business-driven exercise in which current and proposed process flows are documented in detail, linked to quantifiable performance metrics and optimized by simulation analysis. Standards for process modeling languages are key to achieving the goal of BPM as well as achieving platform independence of process models. Platform independence is one of the principles behind model-driven engineering (MDE). Thus, the combination of the two concepts, MDE and BPM, has become the target of several works. The MDE was designed to address several issues that have arisen over the past decade. On the one hand, the growing complexity of the platform, with thousands of classes and methods with very complicated dependencies. On the other hand, we can observe the continuous technological evolution of systems, forcing programmers to modify system code whenever a new requirement is given. In the MDE [13] paradigm, every concept must be modeled. Thus, any change in the system must be indicated in the template that represents that system. To model the systems, MDE suggests using domain-specific modeling languages (DSML). Thanks to these languages, different modeling notations are obtained for each type of system. Thereby, the software engineer has specific tools to model all types of systems. Another important concept in MDE is model transformation. By transforming models, the evolution of systems is facilitated. A model can be transformed into another model or into an XML specification as well as source code that implements the functionality of the model. The OMG Group developed the example of the Model Driven Architecture (MDA) of the MDE, which emerged, with the idea of separating the logic of the business specifications of a system from the specific platform details in which the system is implemented. The MDA adds some concepts to the MDE philosophy such as the definition of three levels of abstractions, which will be further detailed throughout this paper. By applying the concepts of MDE to BPM, this work try to reduce the gap between process description languages and formal execution languages. The idea is to have the technical team enhance the business processes transmitted by business analysts, by means of annotations and metadata. These improved business processes remain readable for trades people. Technical annotations are used to apply specific business and technical patterns to the project while hiding technical complexity and they are taken into account when generating executable workflows.

In fact, several studies show the great advantages of combining MDE and business processes. For example, in [2], the author proposes a tool that can be used in agile projects, thanks to the use of MDE, BPMN processes are systematically improved by means of metadata used to include specific transformation models business and technical oriented before generating executable BPEL processes. Thus, business analysts and technical team members can use a common modeling language like BPMN. This improves collaboration between the two teams and improves the overall efficiency and transparency of the project. From a business point of view, previously defined model fragments can easily be incorporated into new models. From a technical point of view, most of the complexity of the execution part can be hidden and the technical constructions are encapsulated in appropriate model transformations. A systematic review carried out by [3] provides a comprehensive view of the propositions and opinions existing in the literature on the application of the MDE paradigm in the management of business processes. Most of the work found indicates the use of model-driven engineering as a valid approach to business process management. There are proposals for using MDA in the context of collaborative business process management, where the driven model acts as an integration standard and allows different organizations to cooperate from a business process perspective. On the other hand, it suggested that MDA is the methodology that guides the design, implementation, maintenance and management of the operational processes of the organization. However, although there is some rejection of this idea, highlighting how far apart the two concepts of MDE and BPM are, and how difficult it is to get cooperation to achieve better results. Most authors are in favor of using MDE in business process management.

III. Related Work

In literature, for dealing with collaborative business processes, the MDE approach is identified as a key catalyst for building a model-based development method. The separation of the layers adopted by this approach has been the interest of several works. In [4], the author describes the process of developing his approach, which is based on two phases, the first being the technology independent phase, the author has created a modeling language (UP-ColBPIP) which is based on interaction protocols to describe the behavior of collaborative processes. The second being the specific platform phase, the author describes how he will implement these collaborative processes. The author's approach respects the separation of the business and technology layer of the MDA, however, the use of the language (UP-ColBPIP) which is an extension of the UML2 does not seem very interesting given that the latter is quite complex for people in the trade or non-IT specialists to understand. Other works like [5], proposes a Framework for the modeling of collaborative business processes, its methodology is based on the MDA approach. It offers a generic metamodel in order to instantiate it as a collaborative business process from which the respective internal business processes of the partners can be derived. Its Framework allows each partner to model with different languages and keep their own working tools. In [6], the author proposes a semi-automatic transformation from the CIM (Computation Independent Model) layer which uses BPMN to the PIM (Platform Independent Model) layer which uses UML, then from the PIM layer to the PSM (Platform Specific Model) layer which uses IFML (Interaction Flow Modeling Language) which is a standard for representing the web interface model. Transformations are performed using rules well defined by ATL. In [7] and [8], the author proposes a semi-automatic transformation from the CIM layer to the PIM layer structured by the (Model-View-Controller) MVC pattern. The CIM layer is represented by BPMN notation and UML activity diagram, as well as, PIM layer is represented by the state, class and package diagrams. The author in [9] proposes a semi-automatic transformation of secure processes to use cases. In CIM layer, the author models the secure process via a BPMN extension. Then, through a set of transformation, refinement rule and checklist, it transforms these process models into use case models at the PIM layer. In [10], at CIM layer the author models the processes with the BPMN standard and uses the value models to identify the services. Then, at the level of the PIM layer, the models are modeled thanks to an extension of the UML activity diagram and the UML use case diagram.

IV. Approach for Collaborative Processes

A. *Definition of the Approach*

In the business world, before collaborating or developing an information exchange interface, collaborators must be involved, mutually engaged, have mutual trust, shared risks, responsibilities and rewards. Indeed, the collaborators must agree on a final goal, and define common objectives, which represents a synthesis of the individual objectives. Identify the participants in the process, the roles that perform and their relationships. Define the management rules that explain the confidentiality chart and the type of data or shared tasks. This work try to respect the concepts of collaboration defined in the previous work [9]. This approach consists of three phases, see Fig. 1:

- Business Level (Phase1): In this level no IT considerations appear. It is a business layer that allows employees to analyze the problematic of the field. It consists in analyzing the requirements of the collaborators and identifying the business need. The Output of this phase is a set of requirements that helps business analysts to understand the problem and identify the collaborative business process that needs to be designed in the next phase.

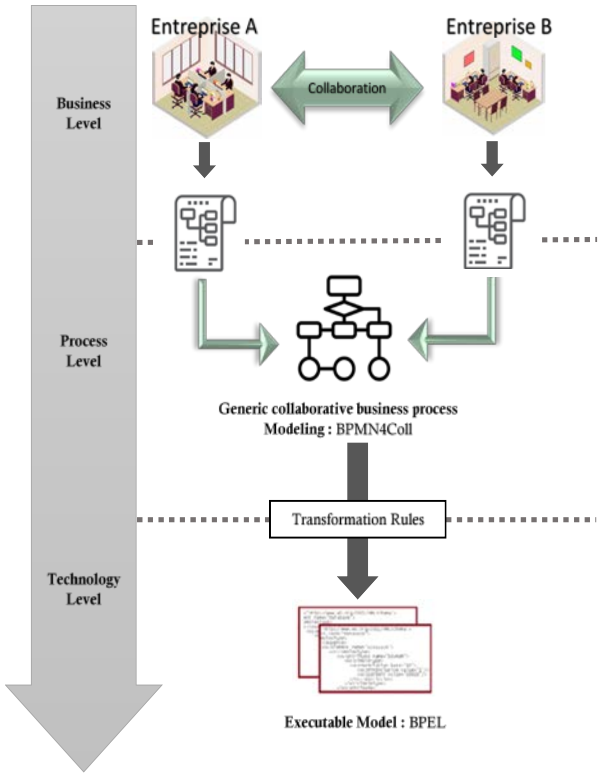


Fig. 1. Model-Driven Architecture for Collaborative Business Processes.

- Process Level (Phase2): This phase represents a layer for modeling the collaborative aspects of the participants. For that, we create a specific modeling language for collaborative environments. It is called BPMN4Coll (it will be further explained in section 4.2) is an extension of the BPMN language that allows a graphical, clear and understandable representation of collaborative business processes. The Output of this phase is a BPMN4Coll Model that describes collaborative process and will be transformed in the next phase in an executable Model.

- Technological Level (Phase3): It is a technical layer, which allows the generation of collaborative processes. The last step of the approach corresponds to the implementation of collaborative business processes. In practice, at this level, collaborative business process are modeled by BPMN4coll DSL. The advantage of this DSL is that it can be transformed into BPEL. We chose BPEL for the execution since the BPMN4Coll is an extension of the BPMN that can be executed thanks to the BPEL. In addition, BPEL is an executable language supported by several tools, it is used to specify interactions with web services and it defines business processes using XML-based language. The transformation is explained further more in section C.

B. *BPMN4Coll: Collaborative Business Process Modeling Language*

The level of modeling process represent a general view of the system independent of the platform. Describe the system without showing the details of using this platform. It can be adapted to different architectures. In this case to deal with the problematic related to collaborative business process, a platform-independent DSL (Domain Specific language) called BPMN4Coll is defined. The latter does not present any technical details and can be transformed into an executable model.

The BPMN4Coll language defined in the previous work [11] is a DSL that allows both technical people (engineers or technicians) and people in the trade to represent their business processes. In addition, the language is dedicated to the representation of collaborative processes that require consistent interaction between participants. The language can be used to model all types of systems. Fig. 2 illustrate an extract of BPMN4Coll metamodel used to perform the transformation in the next step.

The language is defined using the BPMN extension mechanism [12]. It extends two main elements of BPMN (Activity and DataObject) by associating new collaboration elements. This also means that the use of other basic elements of BPMN (such as flowObject, gateway, etc.) remains valid. BPMN4Coll presents an abstract syntax as a metamodel conforms to MOF and their instances models have “.bpmn4coll” extension.

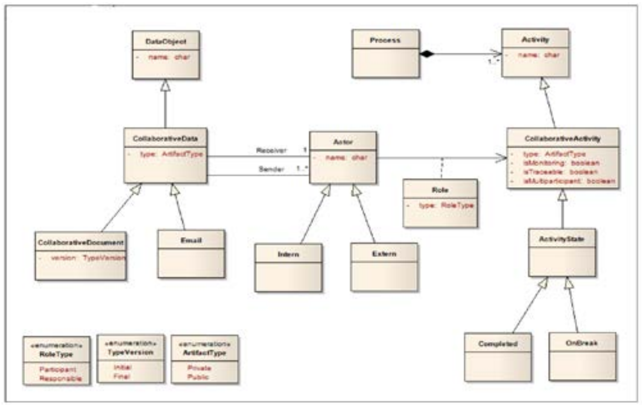


Fig. 2. An Extract of BPMN4Coll Metamodel.

C. *Model Transformation from BPMN4Coll to BPEL*

1) Transformation steps: In order to make development by model operational, flexible and more productive, the MDE has enriched them through the concept of model transformation. Transformation is an operation that allows the generation of one or more target models, from one or more source models in accordance with well-defined rules. The transformation is exogenous since the source and target model are conform to different metamodels. And it is unidirectional from BPMN4Collaboration to BPEL, see Fig. 3.

The transformation is based on a model-to-model Transformation. However, the underlying metamodel BPMN4Coll and the underlying metamodel BPEL are not identical; in fact, they are very different. Thus, the transformation is not straightforward and sometimes requires mappings between the two languages. The process of transformation shown in Fig. 3 present the steps of the transformation:

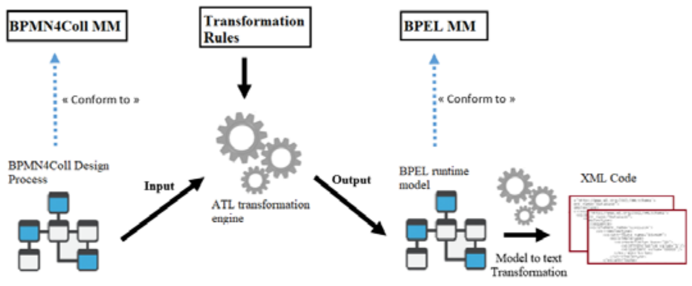


Fig. 3. ATL Transformation from BPMN4Coll to BPEL.

- The use of ATL language (Atlas Transformation Language) to perform this transformation.
- The BPMN4Coll metamodel: represent the source model of transformation (see Fig. 2).
- The BPEL metamodel: represent the target model of transformation (see Fig. 4).

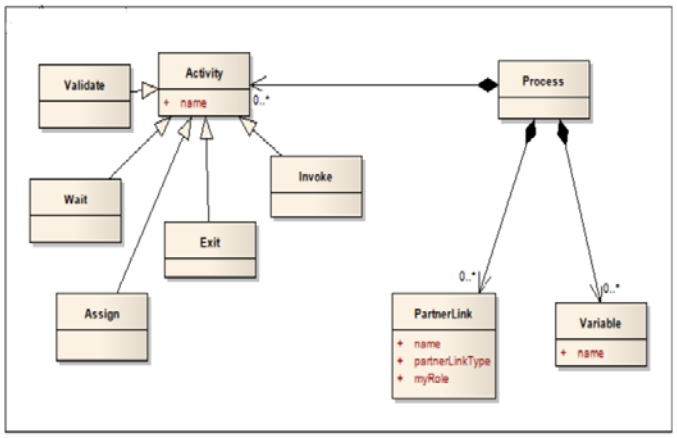


Fig. 4. An Extract of BPEL Metamodel.

- The rules transformation: represent the mappings between the two languages (see the next section).
- Identifying the correspondences between concepts of the source and target models at the level of their metamodels, Table I:

TABLE I. THE CORRESPONDENCES BETWEEN THE BPMN4COLL AND BPEL

BPMN4Coll Concepts	***BBEL Concepts***
Process	Process
Sequence flow	Sequence
Role	PartnerRole
Collaborative Activity	Invoke
Activity OnBreak	Wait
Activity Blocked	Exit
{Intern/Externe} Actor	PartnerLink
CollaborativeData	Variable

2) Transformation rules: This section present some ATL rules used to transform BPMN4Coll metamodel to BPEL metamodel:

- R1: CollaborativeActivity2 Invoke

– Invoke keeps the same name as Collaborative activity concatenated to the word "_coll" to define that is a collaborative Activity.

– Define the boolean type isMultiparticipant helper, which shows that the activity is performed by another participant. If isMultiparticipant returns "true" then all the actors of the activity must be transformed into partnerlink;

– If the type attribute of CollaborativeActivity is "Private" then all CollaborativePartner must have the type "intern"

– If the type attribute of the CollaborativeActivity is "Public" then invoke must have at least one "extern" type actor.

– If the isTraceable attribute of the CollaborativeActivity is "True" then Invoke must have as output a document that ranks the activity traces.

– If the isMonitoring attribute of the CollaborativeActivity is "True" then Invoke must be attached to an actor whose role is responsible.

- R2: Role2Partnerlink

– The attribute Type of the activity Role corresponds to the myRole attribute of the activity partnerlink.

- R3: Actor2PartnerLink

– The Actor activity of BPMN4Coll corresponds to the PartnerLink activity, by adding to partnerLink a new attribute "collaborativePartner" which takes two values {Intern | Extern}.

- R4 : Completed2Assign

– The Completed activity is transformed to assign activity with the "validate" option enabled.

- R6 : CollaborativeData2Variable

– The activity Variable keeps the same name as CollaborativeData concatenated to the word "coll_"

– If the type attribute of the CollaborativeData activity is of the value "Private" then all associated CollaborativePartner must be of type "intern"

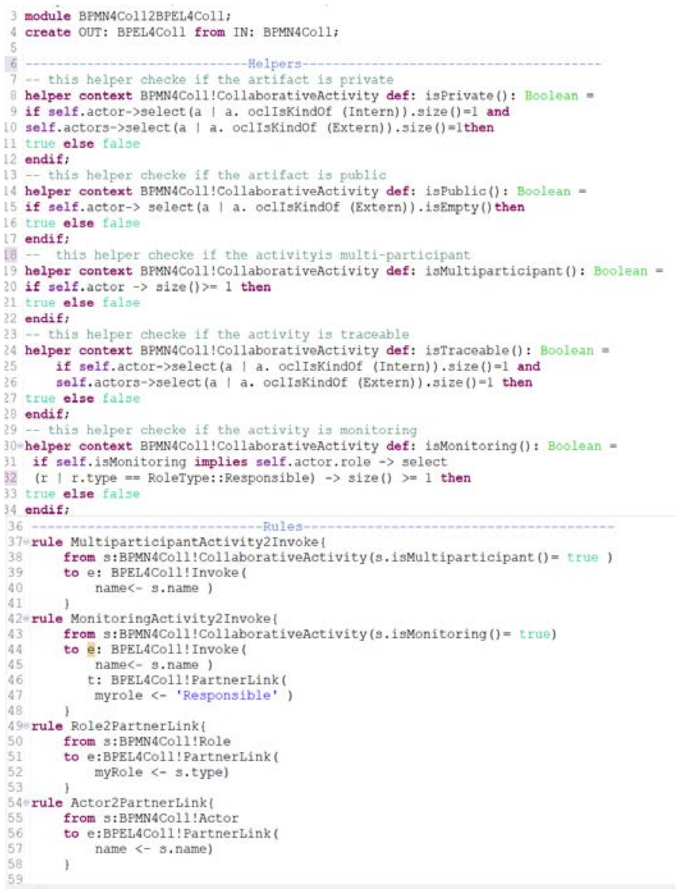
– If the type attribute of the CollaborativeData activity has the value "Shared" then Variable must have at least one "intern" type actor and one "extern" type actor.

- R7 : CollaborativeDocument2Variable

– The CollaborativeDocument is transformed the Variable activity by adding the attribute version.

3) Implementation of transformation: To achieve the goal of transformation between BPMN4Coll and BPEL, we use the ATL plugin of the eclipse environment. Eclipse is a universal platform that offers an integrated development environment. Its goal is to provide a modular environment to easily carry out IT developments. The development of new functionalities is done thanks to the concept of modules called plugins. This concept allows providing a mechanism for the extension of the platform and thus offers the possibility to third parties to develop new functionalities, which are not provided by the tool. Eclipse includes a framework for manipulating MOF metametamodel compliant models, including EMF (Eclipse Modelling Framework). The latter is based on a metamodel called Ecore, which resembles the EMOF 2.0 metamod (subset of MOF 2.0) and helps describe models and provides support for their execution and manipulation in the form of Java objects. It stands as a benchmark in model-driven development.

To achieve the transformation we first present BPMN4Coll and BPEL meta-model into Ecore format. Then, define the transformation engine. On the on hand, ATL language is a hybrid transformation language that clearly separates the two declarative and imperative parts. On the other hand, the source domain is read-only and the target domain is write-only, which allows greater flexibility to the language.


```
module BPMN4Coll2BPEL4Coll;
create OUT: BPEL4Coll from IN: BPMN4Coll;

-----------------------------Helpers---------------------------------------
-- this helper checke if the artifact is private
helper context BPMN4Coll!CollaborativeActivity def: isPrivate(): Boolean =
if self.actor->select(a | a. oclIsKindOf (Intern)).size()=1 and
self.actors->select(a | a. oclIsKindOf (Extern)).size()=1then
true else false
endif;
-- this helper checke if the artifact is public
helper context BPMN4Coll!CollaborativeActivity def: isPublic(): Boolean =
if self.actor-> select(a | a. oclIsKindOf (Extern)).isEmpty()then
true else false
endif;
--  this helper checke if the activityis multi-participant
helper context BPMN4Coll!CollaborativeActivity def: isMultiparticipant(): Boolean =
if self.actor -> size()>= 1 then
true else false
endif;
-- this helper checke if the activity is traceable
helper context BPMN4Coll!CollaborativeActivity def: isTraceable(): Boolean =
    if self.actor->select(a | a. oclIsKindOf (Intern)).size()=1 and
    self.actors->select(a | a. oclIsKindOf (Extern)).size()=1 then
true else false
endif;
-- this helper checke if the activity is monitoring
helper context BPMN4Coll!CollaborativeActivity def: isMonitoring(): Boolean =
 if self.isMonitoring implies self.actor.role -> select
 (r | r.type == RoleType::Responsible) -> size() >= 1 then
true else false
endif;
 -----------------------------Rules---------------------------------------
rule MultiparticipantActivity2Invoke{
     from s:BPMN4Coll!CollaborativeActivity(s.isMultiparticipant()= true )
     to e: BPEL4Coll!Invoke(
         name<- s.name )
     }
rule MonitoringActivity2Invoke{
     from s:BPMN4Coll!CollaborativeActivity(s.isMonitoring()= true)
     to e: BPEL4Coll!Invoke(
         name<- s.name )
        t: BPEL4Coll!PartnerLink(
         myrole <- 'Responsible' )
     }
rule Role2PartnerLink{
     from s:BPMN4Coll!Role
     to e:BPEL4Coll!PartnerLink(
         myRole <- s.type)
     }
rule Actor2PartnerLink{
     from s:BPMN4Coll!Actor
     to e:BPEL4Coll!PartnerLink(
         name <- s.name)
     }

```
Fig. 5. Code Source of Some Helpers and Rules.

The BPMN4Coll language is an extension of BPMN so it is obvious that it consist of the same BPMN elements. Thus, in this work it is not a question of transforming all the elements of the BPMN4Coll language to BPEL because in the literature there are several works dedicated to the problem of transforming BPMN to BPEL. The objective of this work is to accomplish the transformation of new collaboration concepts added by the BPMN4Coll extension. For this, we need to create helpers and rules, which define the new collaboration concepts and then define the declarative rules for producing the input elements. The Fig. 5 present some helpers and rules used in the ALT Transformation.

V. Illustrative Example

This section try to give examples of transforming a simple of a client/Provider process Fig. 6 which is modeled with BPMN4Coll language onto the corresponding BPEL schema; using rules define in the previous section.

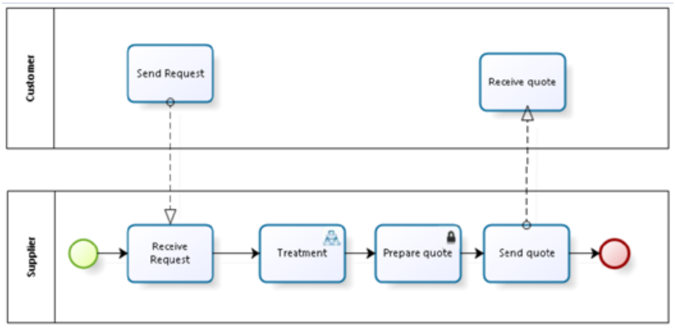


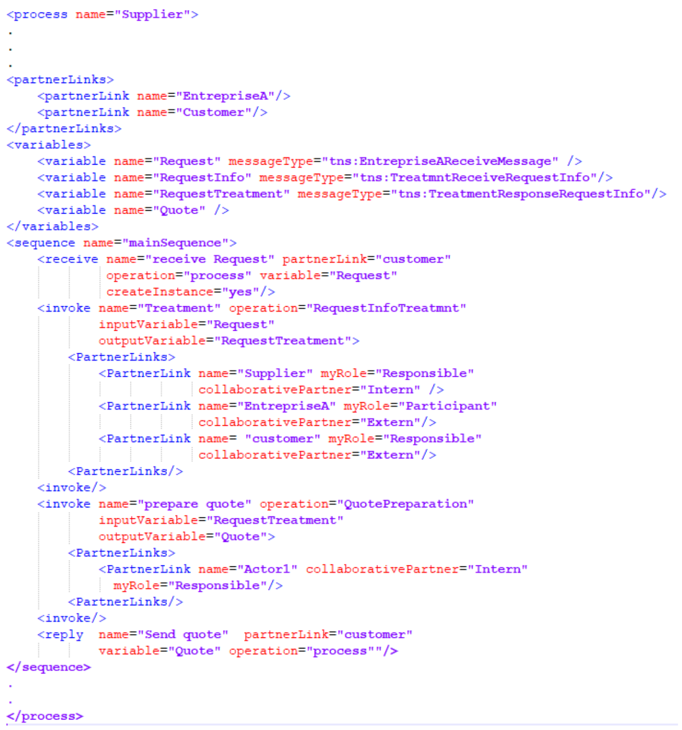
Fig. 6. Example of a Simple Process Modeled by BPMN4Coll.

The example represents a simple process that illustrates the request for quotation operation from a supplier. The process consists of two participants (the customer and the supplier), It begins with the receipt of a request for quotation. Then the supplier processes the request. Then he calculates and prepares the estimate. Finally, the process ends by sending the quote to the customer. The two activities "treatment" and "Prepare quote" are collaborative activities; the first one is of the multi-participant type as well as the second is a private or confidential activity.

The result of the ATL transformation of the previous example is the extract of the BPEL code presented in Fig. 7. The first part <process> defines the process information, in this case the process name is "supplier". Then we define the other processes (or services) with which it will interact in this case "EntrpriseA" and "Customer". Then, we define the variables accessible throughout the process.

The second part represents the main body of the process which consists of a set of activities. The <sequence> part represents a set of activities executed sequentially, it contains a <receive> reception activity to obtain the input message to start the process. <Invoke> represents collaborative activity and <Reply> element is used to send the final message for closing the process.

Finally we close the process with </process>.



```
<process name="Supplier">
.
.
.
<partnerLinks>
    <partnerLink name="EntrepriseA"/>
    <partnerLink name="Customer"/>
</partnerLinks>
<variables>
    <variable name="Request" messageType="tns:EntrepriseAReceiveMessage" />
    <variable name="RequestInfo" messageType="tns:TreatmntReceiveRequestInfo"/>
    <variable name="RequestTreatment" messageType="tns:TreatmentResponseRequestInfo"/>
    <variable name="Quote" />
</variables>
<sequence name="mainSequence">
    <receive name="receive Request" partnerLink="customer"
             operation="process" variable="Request"
             createInstance="yes"/>
    <invoke name="Treatment" operation="RequestInfoTreatmnt"
            inputVariable="Request"
            outputVariable="RequestTreatment">
        <PartnerLinks>
            <PartnerLink name="Supplier" myRole="Responsible"
                         collaborativePartner="Intern" />
            <PartnerLink name="EntrepriseA" myRole="Participant"
                         collaborativePartner="Extern"/>
            <PartnerLink name= "customer" myRole="Responsible"
                         collaborativePartner="Extern"/>
        <PartnerLinks/>
    <invoke/>
    <invoke name="prepare quote" operation="QuotePreparation"
            inputVariable="RequestTreatment"
            outputVariable="Quote">
        <PartnerLinks>
            <PartnerLink name="Actor1" collaborativePartner="Intern"
              myRole="Responsible"/>
        <PartnerLinks/>
    <invoke/>
    <reply  name="Send quote"  partnerLink="customer"
            variable="Quote" operation="process""/>
</sequence>
.
.
</process>
```

Fig. 7. The Example Transformed into BPEL.

VI. Conclusion

The modeling of collaborative business processes faces several challenges: The complexity of modeling intensifies during the coupling of business processes because each participant has their own set of "private" models that are modeled with different languages. Employees have heterogeneous application environments, different cultures and different business rules. The interoperability problem persists because an evolution or change in a collaborative process requires a lot of effort. To solve these problems we opted for an of MDA approach which allows to:

- Increased level of abstraction because collaborative process models are technology independent.
- Guarantee of consistency in the technological solutions generated because they are automatically generated from a well-defined transformation procedure.
- Independence of collaborative process models from Business to Business standards, which increases the flexibility of these models.

The main objective of this architecture is the separation of the two business and technological layers. Thus, achieve the design of collaborative processes independent of the particularities of B2B standards. In fact, to support the design of collaborative processes, this work propose a graphical DSL named BPMN4Coll based on BPMN and specific to the field of collaborative business processes. Then, the automatic generation of executable processes from conceptual models of collaborative processes. This second phase of the approach focuses on the implementation of these collaborative processes, thanks to a transformation of the BPMN4Coll models into an executable process written in BPEL.

The next work will provide more details on the ATL transformation engine as well as a use case to make our approach a reality.

REFERENCES

[1] M. Hammer and J. Champy, Reengineering the Corporation: A Manifesto for Business Revolution. HarperBusiness, 1993.

[2] P. Bauler, F. Feltz, E. Frogneux, B. Renwart,and C. Thomase, « Usage of Model Driven Engineering in the context of Business Process Management », 2008.

[3] J. M. Perez, F. Ruiz, and M. Piattini, « MDE for BPM: A Systematic Review », in Software and Data Technologies, vol. 10, J. Filipe, B. Shishkov, and M. Helfert, Éd. Berlin, Heidelberg: Springer Berlin Heidelberg, 2008, p. 127-135.

[4] P. D. Villarreal, I. Lazarte, J. Roa, and O. Chiotti, « A Modeling Approach for Collaborative Business Processes Based on the UP-ColBPIP Language », in Business Process Management Workshops, vol. 43, S. Rinderle-Ma, S. Sadiq, and F. Leymann, Éd. Berlin, Heidelberg: Springer Berlin Heidelberg, 2010, p. 318-329.

[5] K. Bouchbout, J. Akoka, and Z. Alimazighi, « An MDA-based framework for collaborative business process modelling », Business Process Mgmt Journal, vol. 18, no 6, p. 919-948, nov. 2012, doi: 10.1108/14637151211283357.

[6] M. Melouk, Y. Rhazali, and H. Youssef, « An Approach for Transforming CIM to PIM up To PSM in MDA », Procedia Computer Science, vol. 170, p. 869-874, 2020, doi: 10.1016/j.procs.2020.03.122.

[7] Y. Rhazali, Y. Hadi, and A. Mouloudi, « Model Transformation with ATL into MDA from CIM to PIM Structured through MVC », Procedia Computer Science, vol. 83, p. 1096-1101, 2016, doi: 10.1016/j.procs.2016.04.229.

[8] Y. Rhazali, Y. Hadi, and A. Mouloudi, « Transformation approach CIM to PIM: from business processes models to state machine and package models », in 2015 International Conference on Open Source Software Computing (OSSCOM), Amman, Jordan, sept. 2015, p. 1-6, doi: 10.1109/OSSCOM.2015.7372686.

[9] A. Rodríguez, E. Fernández-Medina, and M. Piattini, « Towards CIM to PIM Transformation: From Secure Business Processes Defined in BPMN to Use-Cases », in Business Process Management, vol. 4714, G. Alonso, P. Dadam, and M. Rosemann, Éd. Berlin, Heidelberg: Springer Berlin Heidelberg, 2007, p. 408-415.

[10] V. De Castro, E. Marcos, and J. M. Vara, « Applying CIM-to-PIM model transformations for the service-oriented development of information systems », Information and Software Technology, vol. 53, no 1, p. 87-105, janv. 2011, doi: 10.1016/j.infsof.2010.09.002.

[11] L. Amdah and A. Anwar, « A DSL for collaborative Business Process », in 2020 International Conference on Intelligent Systems and Computer Vision (ISCV), Fez, Morocco, juin 2020, p. 1-6, doi: 10.1109/ISCV49265.2020.9204044.

[12] L. Amdah and A. Anwar, « BPMN4 Collaboration: An Extension for collaborative Business Process », Adv. sci. technol. eng. syst. j., vol. 4, no 6, p. 297-305, 2019, doi: 10.25046/aj040638.

[13] Schmidt, D.C. 2006. “Guest Editor’s Introduction: Model-Driven Engineering.” Computer 39 (2): 25–31. https://doi.org/10.1109/MC.2006.58.