ISSN 2278-3091

Volume 8, No.6, November – December 2019

International Journal of Advanced Trends in Computer Science and Engineering

Available Online at http://www.warse.org/IJATCSE/static/pdf/file/ijatcse112862019.pdf

https://doi.org/10.30534/ijatcse/2019/112862019

Modeling and Implementation of Web Services Composition Based on MARDS

Nouha ADADI[1], Mohammed BERRADA[2], Mohamed HALIM[3], Driss CHENOUNI[4]

[1]IPI Laboratory, Sidi Mohamed Ben Abdellah University , Fez, Morocoo, nouhaadadi@gmail.com
[2]IPI Laboratory, Sidi Mohamed Ben Abdellah University , Fez, Morocoo, mohammed.berrada@gmail.com
[3]IPI Laboratory, Sidi Mohamed Ben Abdellah University , Fez, Morocoo,, mohamed.halim@usmba.ac.ma
[4]IPI Laboratory, Sidi Mohamed Ben Abdellah University , Fez, Morocoo, d_chenouni@yahoo.fr

ABSTRACT

Since the emergence of Service Oriented Architecture (SOA) and its implementation with the Web services technology, combining several Web services to response to a complex request presented a hard challenge. This is an area that has attracted the interest of many research organizations and manufactories. The goal ofthis research work is proposing a new approach of modeling and implementing web services composition. This approach allows a clear and structured modeling and easy implementation by generating automatically the executable code from the model. In this paper, we firstly propose a modeling of the composed system using the BPM (Business Process Management) standards and based on Multi-agent reactive decisional system (MARDS). Secondly, we seek to generate the executable code BPEL (Business Process Executable Language) from this modeling to implement the composite web service. Finally, we present a case study to prove the feasibility and reliability of our proposed approach.

Key words: Web services composition; Modeling; MARDS; Implementation; BPEL.

1. INTRODUCTION

Nowadays, several companies expose their competences on the Internet to interact with its partners and seek new markets. Web services are presented as the ideal and expected solution to this need due to their interoperability and the possibility of reuse by other applications. However, these services themselves may not be very useful without resorting to a combination with other services.

The composition of web services is to create value-added services by reusing existing services. By creating composed services, it is possible to reduce the costs of development, the production time and therefore, respond to the growing demand for applications. Web service composition is a subject of research that has been widely studied both academically and industrially.

Our work is to propose an approach that combines the Multi agent system (MAS) with BPM (Business Process Management) standards in order to reduce time and development costs, while making the composition of services more reliable.

The layout of this paper is as follows. In the second section, we present a classification of web services composition approaches and a summary of our proposed approach. Then, we focus on the tasks of modeling and implementation respectively in the third and fourth sections. As part of a case study we consider an online item purchase problem in the fifth section. This problem is a typical web services composition scenario to apply the concepts of our approach. The sixth section is devoted to conclusion and future work.

2. PROPOSITION OF WEB SERVICES COMPOSITION APPROACH

2.1. Classification of web services composition approaches

Constructing a composition approach is not a trivial task and is a problem that has been the subject of many research studies over the last decade. The approaches presented and studied in the literature to solve this problem can be grouped into four classes (Figure 1): workflow-based approaches [1], approaches based on artificial intelligence planning techniques [2], approaches based on dependence graphs [3], and model-driven approaches [4]. All of these approaches differ in how they perceive the composition of Web services and use techniques that are compatible with that vision. In particular, they consider the composition of Web services as a problem of managing Workflows, of planning, of optimal path research or of models. And apply techniques from software engineering, artificial intelligence, graph theory or specification. We believe that they are all mature and equivalent if we consider their degree of adaptation to Web Services composition problem solving; and the proven maturity of the techniques that they use. In our work we choose to adopt the model-driven approach. Certainly the other approaches all allow to generate composition plans, however their major limitation lies in the dependence of proposed composition solutions on the semantic models of Web services description; as well as the level of composability considered which is often limited to the pairing of the inputs and outputs of the services, and not treating the different facets of composability between two or more services to connect.

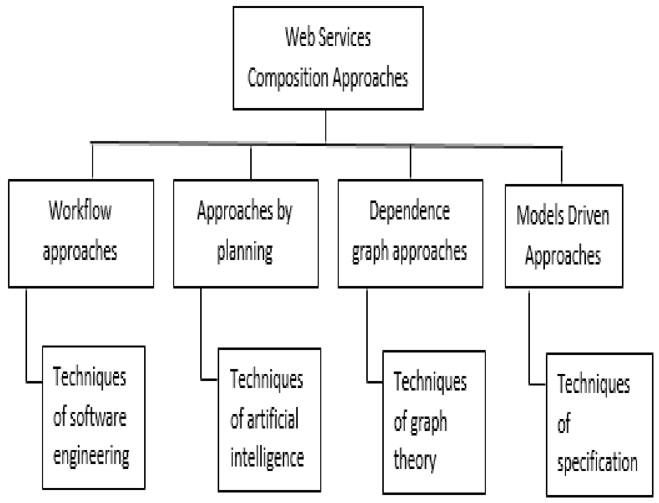

Web Services Composition Approaches
Workflow approaches
Approaches by planning
Dependence graph approaches
Models Driven Approaches
Techniques of software engineering
Techniques of artificial intelligence
Techniques of graph theory
Techniques of specification

Figure 1: classification of web services composition approaches

2.2. Proposed Approach

In our work, we choose to adopt the Models Driven Approach (MDA), which concentrates on the realization of abstract models rather than on computer or algorithmic concepts. Thus, the phase of specification represents an important part of the cycle of development of composite web service. To proceed to the cycle of development of MDA, The developer specifies the composition scenario using existing services and a modeling language. In accordance with the standard web services architecture defined by the W3C [5], we consider that the developer can find these existing services using a directory such as UDDI [6]. Once the specification is complete, the developer proceeds to generate the code for the new composed service.

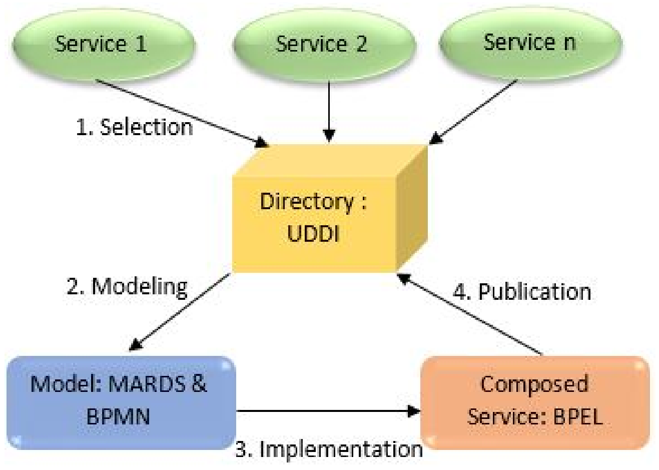


Figure 2: Proposed approach of web services composition

In the process presented in figure1, the first step in the cycle is the discovery of services using search mechanisms such as the UDDI directory. Once the existing services are selected, we pass to the second step which is specification. At this level we propose a modelling using BPMN (Business Process Model and Notation) [7] and MARDS model. The BPMN notation, is a modeling language, it is more adapted to the domain of the Web services and sufficiently expressive to allow the generation of executable code from it. The MARDS model constitutes an approach among the newest and most useful ones for the composing and modeling of complex system [8] [9]. We have used this system in our approach because it allows to model the composition of services in a simple and powerful way, and in well-structured architecture. The next step is the implementation of the system by generating BPEL [10] executable code from the BPMN specification. Finally, once the composed service is implemented, the last step is usually to publish it in the directory to facilitate its future use. The two stages of modelling and implementation are presented in the following sections.

3. PHASE OF MODELING

The objective of this work is to develop an approach allowing the modeling and the implementation of the business processes generated by the composition of Web services. The composition of web services can describe and model these processes using languages and standards based essentially on the BPM (Business Process Management)[11] concept such as BPMN notation. This modeling becomes more complex and unstructured when the number of services is increasing, making the implementation phase difficult to afford. This problem requires providing a well-structured architecture that allows services to be composed in powerful way, and allows developer to add and remove services transparently without affecting other services. In turn, multi-agent systems offer complex and distributed compound systems that can be modeled by BPM languages, so we will use this system for the composition of web services. Some multi-agent models, such as the Multi-Agent Decision-Reactive System (MARDS), have a well-structured hierarchical architecture and can be used to model business processes in a simple, powerful and transparent way to facilitate the generation of executable code. This model introduces flexible concepts at the level of its organization, its modes of interaction and its agents, this justifies its use in several areas and especially in the modeling of complex systems [8] [9].

3.1. MARDS Model

The Multi-Agent reactive decisional System (MARDS) [12] is one of the newest approaches to model reactive system. It is composed by the decisional reactive agents (DRA), which are interconnected by communication interfaces. DRA receives actions (or goals) and can act autonomously until they are realized within the appropriate timeframe. To achieve a goal, we can define one or a series of objectives chained in a certain way that we propose to solve this goal. The internal structure of a MARDS is based on a two-level tree (Figure 3), composed in parallel of an Supervisor DRA (SDRA), of two or more component sub-agents, and of interfaces communication between the supervisor and his sub-agents. For a sub-agent MARDSi, it can be either a simple DRA or a MARDS built recursively.

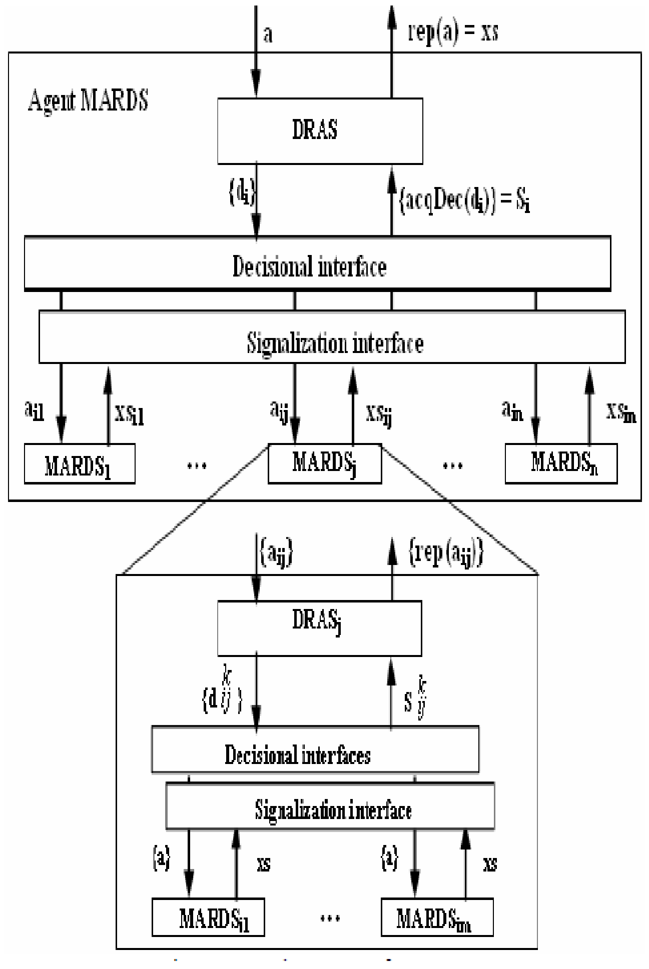


Figure 3: Internal Structure of a MARDS.

According to Figure 3, the Supervisor DRA generates several decisions {di, i = 1 ... m} on receipt of the initial action {a}. Each of these decisions will be translated by the decision interface to one or more actions {aik, k = 1 ... n} appropriate to the lower level agents {MARDS$_k$}. The external states issued by these agents will be translated by the signaling interface to a single signaling {si}, which represents the acquittal of the decision {di}. The MARDS agent will subsequently generate the decision {di + 1} or just the final external state {e} considered by the MARDS agent as a response to the initial action.

The specification phase of our approach is based on the MARDS model and using BPMN standards. It is called BPMN-MARDS profile. In the following we present the prototype modeling of MARDS by BPMN.

3.2. BPMN-MARDS profile

The business model shown in Figure 4 represents a prototype model for any MARDS agent. The initial action generates several decisions {dec_i, i = ... m} each of which must trigger one or more actions {act_ki} intended and completed in parallel by the agents {MARDSk}.

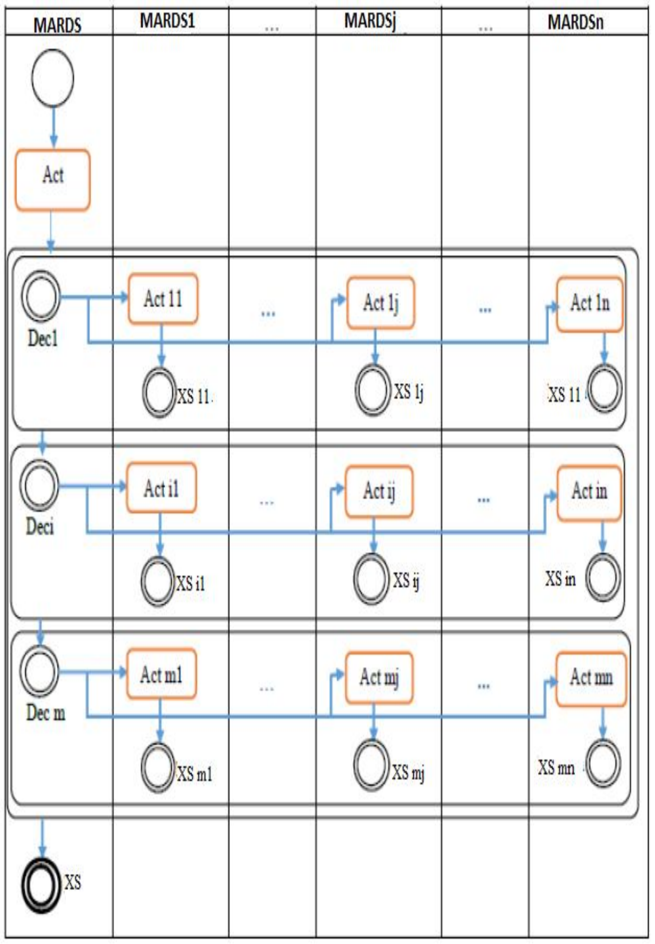


Figure 4: Business model of MARDS agent

So, any decision represents the starting event; actions are parallel activities; external states are intermediate events; and the signaling translated by the signaling interface, is intended for the supervisor as a final event. All these interrelated elements, as they are described in sequence, compose an independently modeled sub-process.

Thus, the modeling of the last sub-process, triggered by the last decision, will be followed by the final event that represents the final external state of the base process.

Concerning the activities achieved by {MARDSk} agents, they can be modeled as simple tasks or sub-processes according to the nature of the agents in question. For a composite MARDSk agent, it uses other sub agents to perform an activity that it will be modeled as a sub process. In addition, an activity performed by a simple DRA agent will be modeled by a simple task.

4. PHASE OF IMPLEMENTATION

Standards have been proposed for the implementation of web services like BPEL language, which is an initiative of BPMI. The purpose of BPEL language is describe the overall dynamics of web services by proposing an XML representation of the activities related to the execution of a process.

The purpose of this section is to simulate the MARDS system using the BPEL language. This simulation is done by

BPEL translation of MARDS's BPMN business model. The principle of this simulation is to consider any agent composing the MARDS system as a web service and the supervisor agent that controls and manages these components as the principal BPEL process. The Structure of this simulation is divided into two parts, Definition part (WSDL interface) and Process part (BPEL process).

4.1. WSDL interface

The goal of WSDL [13] is to describe the services as a set of operations and abstract messages, connected to protocols and network servers. So WSDL can locate and describe in detail the use of a web service. This description contains operational information about the service such as: a definition of messages, portType, and partnerLinkType. Table 1 describes the WSDL elements defined for an agent MARDS.

Table 1: WSDL components of the MARDS

WSDL elements	WSDL components of the MARDS
message	• Input message (action: text). • Output message (external state: text).
portType	A first portType "MARDS_PT" which groups operations "action_i" to receive requests. • A second portType "MARDSCallBack_PT" that groups operations "action_iCallback" to respond to requests.
partnerLinkType	A partnerLinkType "MARDS_action_i_LT" for each action with two roles: "action_i_Role" and "action_iCallBack_Role".

4.2. Process description

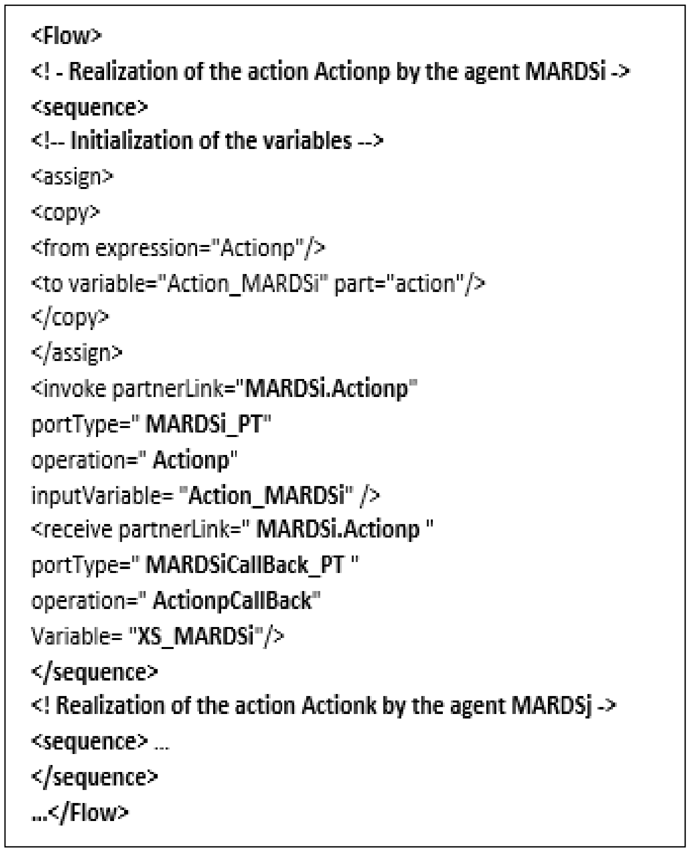
The BPEL file includes:

A. *The definition of elements (variables, partnerLink ...)*

- The declaration of the partnerLinks, the only partner is "MARDS_action" with two roles.
  - The first "myRole = action_Role" indicates the role of the BPEL process.
  - The second "partnerRole = actionCallBack_Role" indicates the role of the partner.
- The definition of the initial activity *receive* as a request from the partner "MARDS_Action", a request made by the operation "Action" integrated in the portType "MARDS_Action_PT".
- A definition of the last activity *invoke* as a response to the request of the partner "MARDS_Action" by the operation "ActionCallBack" integrated in the portTpe "MARDS_ActionCallBack_PT".

B. *The description of the process and the sequence of his actions.*

In a MARDS system, each decision issued by the supervisor is translated as several actions corresponding to the agents at the next lower level. A decision is represented by *Flow* activity that allows the execution of several parallel actions (Figure 5). These actions will be considered as calls (invocations) to the functions implemented by these agents (web services). Then, the external states deliberated by these agents are answers to the functions invoked by the supervisor. For completion of the initial action, each composite agent will, in turn, call other functions implemented by its own sub agents to answer the supervisor's call. So, the hierarchical structure of the MARDS system decomposes the BPEL process into several sub-processes. Therefore, the definition of the BPEL process must contain those of these sub-processes.



```
<Flow>
<! - Realization of the action Actionp by the agent MARDSi ->
<sequence>
<!-- Initialization of the variables -->
<assign>
<copy>
<from expression="Actionp"/>
<to variable="Action_MARDSi" part="action"/>
</copy>
</assign>
<invoke partnerLink="MARDSi.Actionp"
portType=" MARDSi_PT"
operation=" Actionp"
inputVariable= "Action_MARDSi" />
<receive partnerLink=" MARDSi.Actionp "
portType=" MARDSiCallBack_PT "
operation=" ActionpCallBack"
Variable= "XS_MARDSi"/>
</sequence>
<! Realization of the action Actionk by the agent MARDSj ->
<sequence> ...
</sequence>
...</Flow>
```

Figure 5: BPEL description of a decision

In the next section, we will provide a case study that allows modeling and implementation of a web service composition system using our development approach described previously.

5. CASE STUDY

As case study, we will consider in this paper an online item purchase problem. This is a simple illustrative example that present a typical scenario for web services composition problem. As far as creating the e-commerce composite service, we can use seven basic services ("Item", "Provider", "Promotion", "Cart", "Payment_Detail", "Bank" and "Transport") that will internally execute the e-commerce service, each one executes a set of tasks.

5.1. Modelling phase

A. Structure of Web services composition

For modeling the example of Web Service composition we are going to follow these steps for composing MARDS agents:

- To organize agents in layers depending on the tasks and activities that they execute.
- To specify atomic agents which execute simple services, this agents present the basic components (DRA agents) of each layer.
- To identify the first and the intermediate composite agents (MARDS agent) of each layer, if they exist.
- To specify the principal composite agent that represents the agent which receives the main composite request from client.

Applying these steps to the example, we obtain the MARDS structure of this example shown in figure 6.

In this model of service composition, the basic components are: ("Item", "Provider", "Promotion", "Cart", "Payment_Detail", "Bank" and "Transport". The intermediate components are: "Amazon"; "Pay"; "Research". The main composite component is "E-commerce".

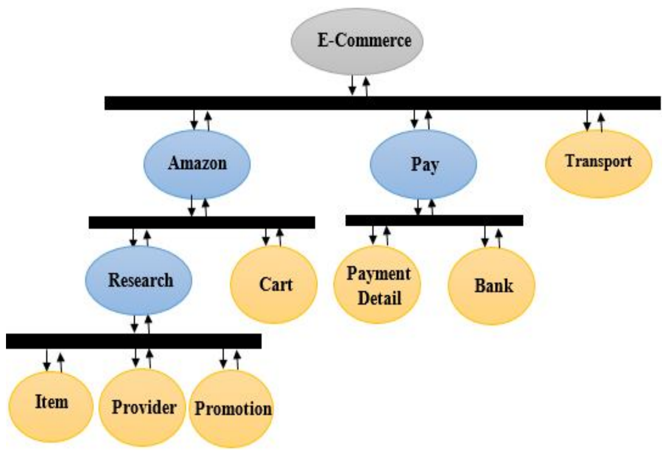


Figure 6: Web service composition model based on MARDS

B. Business Model of the Composite Service "E-commerce"

The Figure 6 shows the Business model of the composition services model based on MARDS. The action "A_Online Bay" received by "E-commerce" component generates three decisions {D1_Choose Items; D2_Pay; D3_ Deliver}. Each decision corresponds to a several sub-actions received by "Amazon" component {D1_Choose Items; A_Select Items}, by "Pay" component {D2_Pay, A_ Pay} and by "Transport" component {D3_Deliver, A_Deliver}. Every sub-action received by any composite component will be realized and modeled as a sub-process.

The sub-action "A_Select Items" received by the "Amazon" component generates two decisions {D1_Select Items; D2_ Add to Cart }. The first sub-decision "D1_ Select Items" generates the {A_ Search} action for "Research" composite component.

The second sub-decision "D2_ Add to Cart" generates the "A_Add to Cart" action for "Cart" basic component. The sequencing of the two sub-decisions corresponds to the sub-process of the "A_Select Items" sub-action.

The sub-action "A_ Search" received by the "Research" composite component generate the sub-decision "D_Search". On his part, this sub-decision generates three parallel sub-actions {A_Search Item; A_Search Provider; A_Search Promotion} for the components "Item"; "Provider" and "Promotion". The three sub-actions correspond to the subprocess of the sub-action "A_Search". The sub-action "A_ Pay" received by the "Pay" component generates two decisions {D1_Call for payment detail; D2_ Invoice}. The first sub-decision "D1_ Call for payment detail" generates the {A_ Call for payment detail} action for "Payment_Detail" basic component. The second sub-decision "D2_ Invoice" generates the "A_ Invoice" action for "Bank" basic component. The sequencing of the two sub-decisions corresponds to the sub-process of the "A_ Pay" sub-action.

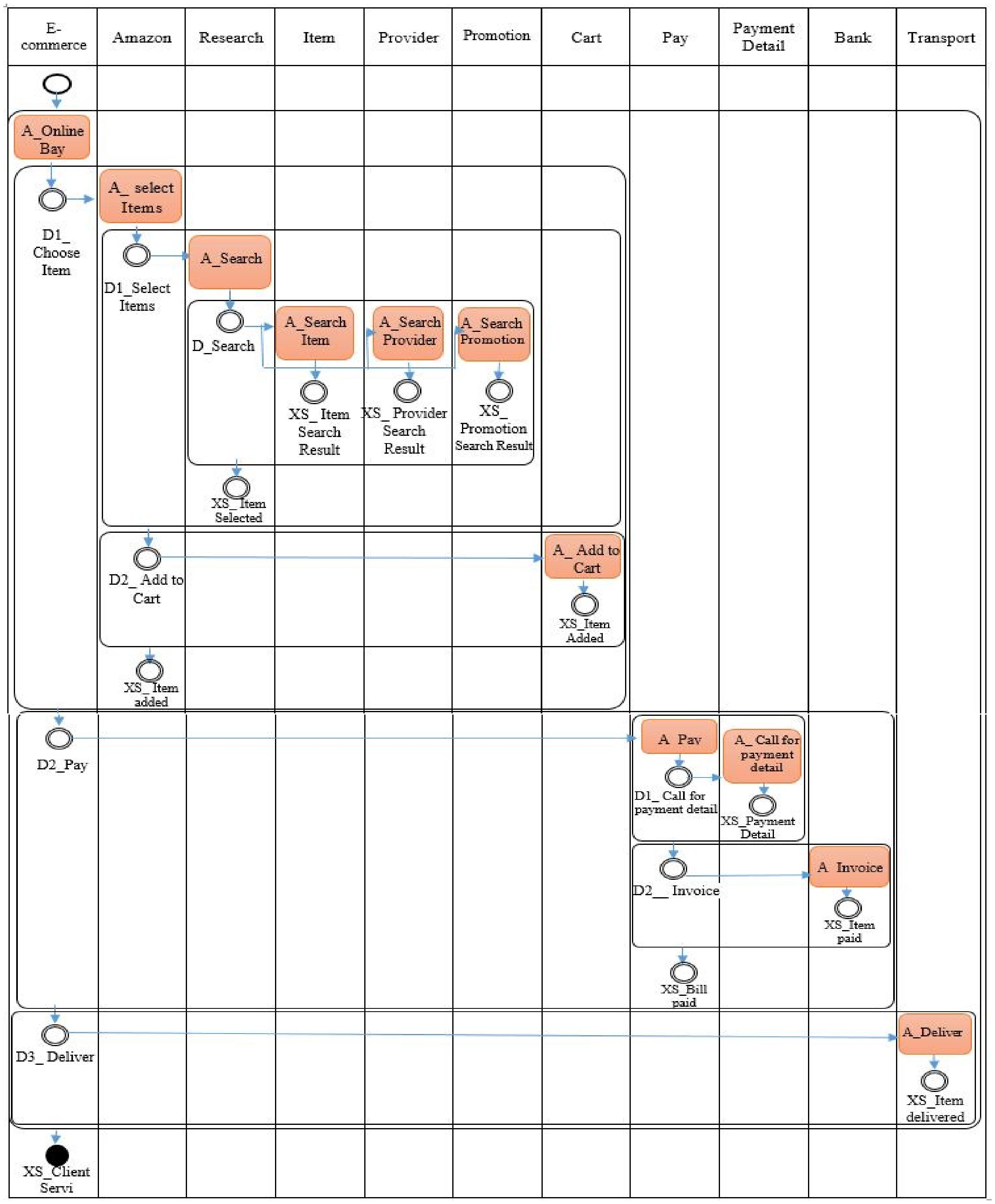
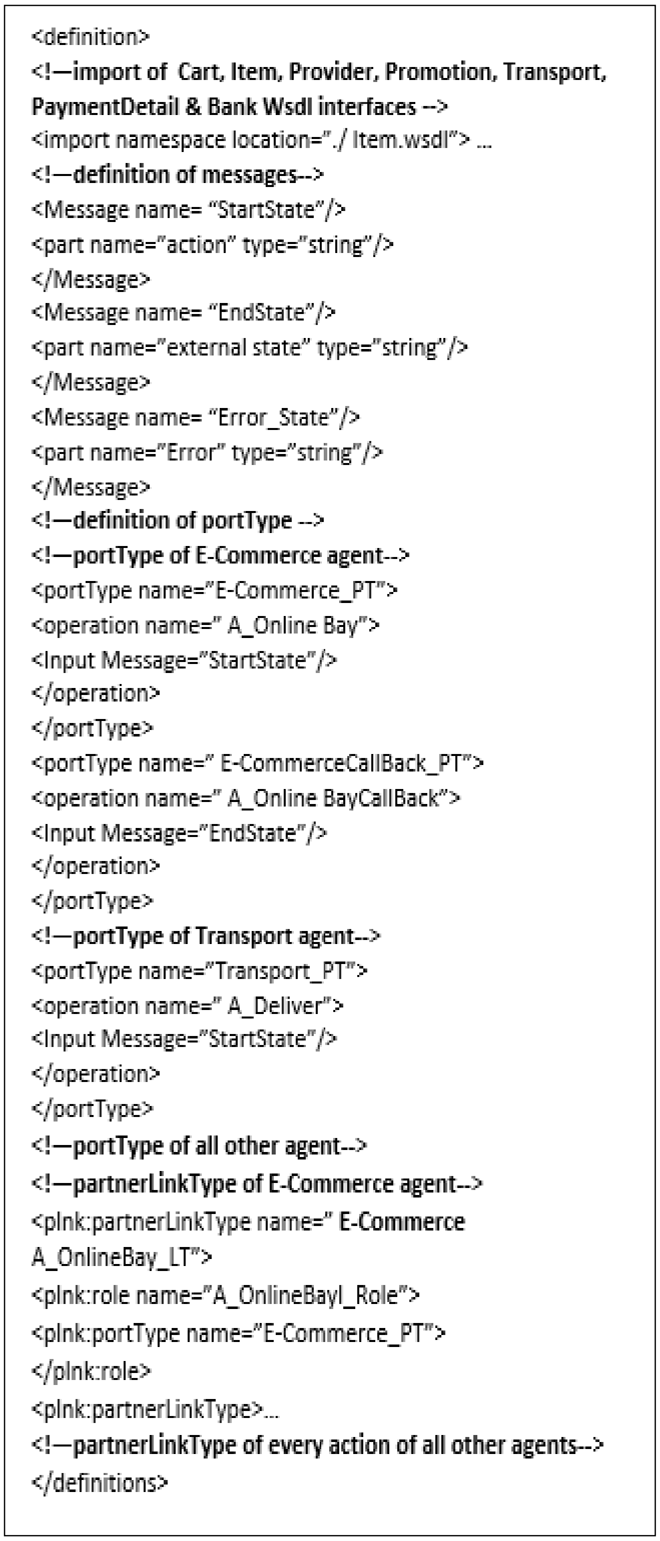


Figure 7: BPMN Diagram of "E-Commerce" Composition Scenario

5.2. Phase of implementation

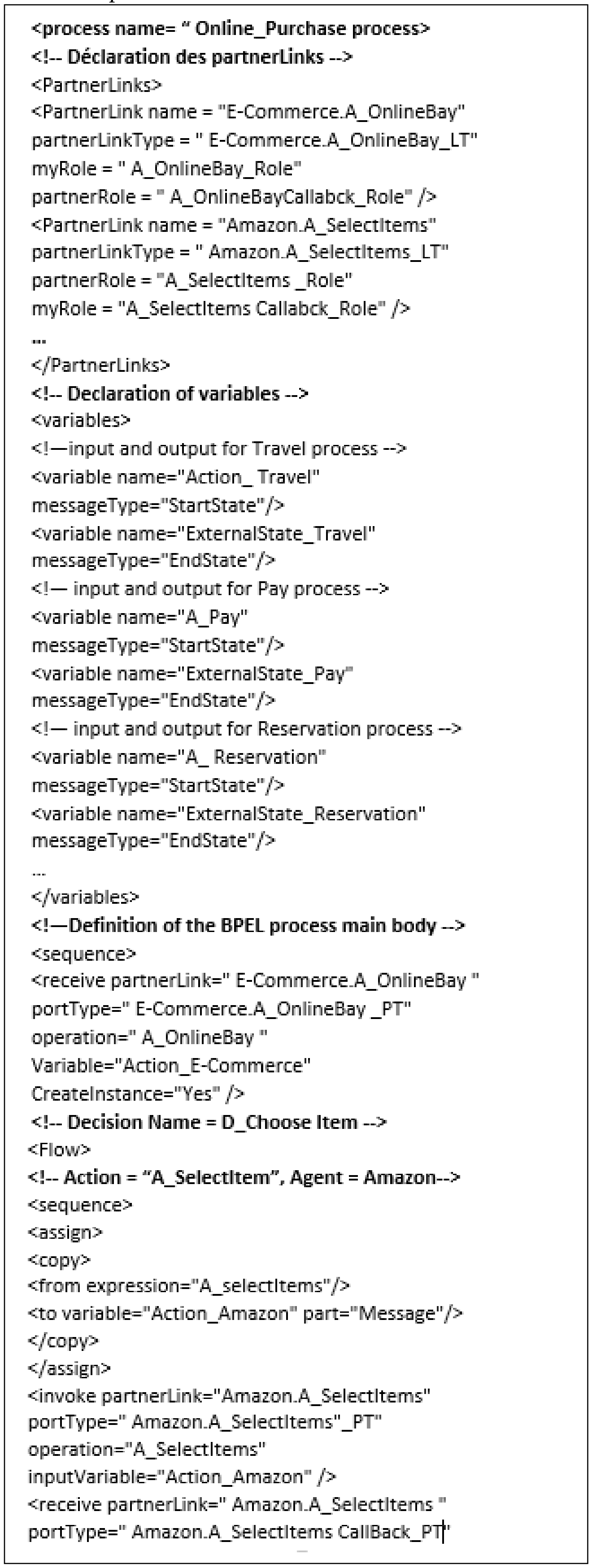
Applying the rules and methods described in Section 4 to the BPMN models presented in Figure 7, we obtain the following BPEL description.

A. WSDL Interface



```
<definition>
<!—import of  Cart, Item, Provider, Promotion, Transport,
PaymentDetail & Bank Wsdl interfaces -->
<import namespace location="./ Item.wsdl"> ...
<!—definition of messages-->
<Message name= "StartState"/>
<part name="action" type="string"/>
</Message>
<Message name= "EndState"/>
<part name="external state" type="string"/>
</Message>
<Message name= "Error_State"/>
<part name="Error" type="string"/>
</Message>
<!—definition of portType -->
<!—portType of E-Commerce agent-->
<portType name="E-Commerce_PT">
<operation name=" A_Online Bay">
<Input Message="StartState"/>
</operation>
</portType>
<portType name=" E-CommerceCallBack_PT">
<operation name=" A_Online BayCallBack">
<Input Message="EndState"/>
</operation>
</portType>
<!—portType of Transport agent-->
<portType name="Transport_PT">
<operation name=" A_Deliver">
<Input Message="StartState"/>
</operation>
</portType>
<!—portType of all other agent-->
<!—partnerLinkType of E-Commerce agent-->
<plnk:partnerLinkType name=" E-Commerce
A_OnlineBay_LT">
<plnk:role name="A_OnlineBayl_Role">
<plnk:portType name="E-Commerce_PT">
</plnk:role>
<plnk:partnerLinkType>...
<!—partnerLinkType of every action of all other agents-->
</definitions>
```

B. BPEL process



```
<process name= " Online_Purchase process>
<!-- Déclaration des partnerLinks -->
<PartnerLinks>
<PartnerLink name = "E-Commerce.A_OnlineBay"
partnerLinkType = " E-Commerce.A_OnlineBay_LT"
myRole = " A_OnlineBay_Role"
partnerRole = " A_OnlineBayCallabck_Role" />
<PartnerLink name = "Amazon.A_SelectItems"
partnerLinkType = " Amazon.A_SelectItems_LT"
partnerRole = "A_SelectItems _Role"
myRole = "A_SelectItems Callabck_Role" />
...
</PartnerLinks>
<!-- Declaration of variables -->
<variables>
<!—input and output for Travel process -->
<variable name="Action_ Travel"
messageType="StartState"/>
<variable name="ExternalState_Travel"
messageType="EndState"/>
<!— input and output for Pay process -->
<variable name="A_Pay"
messageType="StartState"/>
<variable name="ExternalState_Pay"
messageType="EndState"/>
<!— input and output for Reservation process -->
<variable name="A_ Reservation"
messageType="StartState"/>
<variable name="ExternalState_Reservation"
messageType="EndState"/>
...
</variables>
<!—Definition of the BPEL process main body -->
<sequence>
<receive partnerLink=" E-Commerce.A_OnlineBay "
portType=" E-Commerce.A_OnlineBay _PT"
operation=" A_OnlineBay "
Variable="Action_E-Commerce"
CreateInstance="Yes" />
<!-- Decision Name = D_Choose Item -->
<Flow>
<!-- Action = "A_SelectItem", Agent = Amazon-->
<sequence>
<assign>
<copy>
<from expression="A_selectItems"/>
<to variable="Action_Amazon" part="Message"/>
</copy>
</assign>
<invoke partnerLink="Amazon.A_SelectItems"
portType=" Amazon.A_SelectItems"_PT"
operation="A_SelectItems"
inputVariable="Action_Amazon" />
<receive partnerLink=" Amazon.A_SelectItems "
portType=" Amazon.A_SelectItems CallBack_PT"
```

```
operation=" A_SelectItems CallBack"
Variable="External State _ Amazon" />
</sequence>
</Flow>
<!-- Decision Name = D_Pay -->
<Flow> ... </Flow>
<!-- Decision Name = D_Deliver -->
<Flow> ... </Flow>
<invoke partnerLink=" E-Commerce.A_OnlineBay "
portType=" E-Commerce.A_OnlineBay CallBack_PT"
operation=" A_OnlineBay CallBack"
outputVariable=" ExternalState _E-Commerce" />
</sequence>
<!—Definition of Amazon process -->
<sequence>...
</sequence>
<!—Definition of Research process -->
<sequence>....
</sequence>
<!—Definition of Pay process  -->
<sequence>
<receive partnerLink=" Pay.A_Pay "
portType=" Pay.A_Pay _PT"
operation=" A_Pay "
Variable="Action_Pay "
CreateInstance="Yes" />
<!-- Decision Name = D_CallForPaymentDetail -->
<Flow>....
</Flow>
<!-- Decision Name = D_Invoice -->
<Flow> ... </Flow>
<invoke partnerLink=" Pay.A_Pay "
portType=" Pay.A_PayCallBack_PT"
operation=" A_PayCallBack"
outputVariable=" ExternalState _Pay" />
</sequence>
</process>
```

6. CONCLUSION

In this work, an approach for modeling and implementing web services composition is proposed.

The modeling phase insists on presenting the business processes of web services composition using BPM standards and the MARDS model. The implementation phase consist on generating the BPEL executable code from the specification.

Our approach of composition of web services is based on standardized and powerful languages, templates and technologies (MARDS model, BPMN Notation, BPEL language), therefore it can solve problems of the web services composition in different application domains and at all levels of complexity. As part of a case study, we chose the E-commerce sector and we have detailed each step of our proposed approach, to better explain, illustrate and help to understand this approach.

For the prospects, we are currently working on the automatic transformation of BPMN models into BPEL code. This work is important to facilitate the task of the developer.

REFERENCES

1. Ardagna D, Comuzzi M, Mussi E, Pernici B, Plebani P 2007 Paws: **A framework for executing adaptive web-service processes,** IEEE Software 24 39–46. https://doi.org/10.1109/MS.2007.174
2. Lécué F, Léger A, Delteil A 2008 DL, **Reasoning and AI Planning for Web Service Composition,** Web Intelligence 445–53. https://doi.org/10.1109/WIIAT.2008.344
3. L. Ying, 2010, **A Method of Automatic Web Services Composition Based on Directed Graph**, cmc International Conference on Communications and Mobile Computing 1 527-31. https://doi.org/10.1109/CMC.2010.91
4. Richard Soley**, Model Driven Architecture (MDA)**, Draft 3.2. Object Management Group, Inc., November 2000.
5. David Booth, Hugo Haas, Francis McCabe, Eric Newcomer, Michael Champion, Chris Ferris, and David Orchard. **Web services architecture.** http://www.w3.org/TR/ws-arch/, February 2004. W3C Technical Reports and Publications. pp 15.
6. T. Bellwood, L. Clement, D. Ehnebuske, A. Hately, M. Hondo, Y. L. Husband, K. Januszewski, S. Lee, B. McKee, J. Munter, et al. **Uddi version 3.0. Published specification, Oasis**, 2002. pp 16, 18, 23, 63 & 72.
7. Chand, Donald & Chircu, A.M.. (2012). **Business process modeling**. 10.4018/978-1-4666-0249-6.ch003.
8. A Aaroud, S. E. Labhalla, and B. Bounabat, **Modelling the handover function of global system for mobile communication**, The International Journal of Modelling and Simulation, ACTA Press, vol 25, n. 2, 2005. https://doi.org/10.2316/Journal.205.2005.2.205-4136
9. M. Berrada, B. Bounabat, and M. Harti, **Modeling and simulation of Multi-Agent reactif decisionnal systems using business process management concepts**, International Review on Computers and Software (IRECOS), vol. 2, n. 2, pp. 159-169, March 2007.
10. OASIS Standard. **Web services business process execution language**, version 2.0, April 2007.
11. BPM, Object Management Group, **Business Process Management Initiative**, 2007. http://www.bpmn.org/
12. B. Bounabat, **Méthode d'analyse et de conception orientée objet décisionnel. Application aux langages synchrones et aux systèmes répartis**, doctoraldiss, Cadi Ayyad University, Faculty of sciences, Marrakech, Morocco, 2000.
13. E. Christensen, F. Curbera, G. Meredith, and S. Weerawarana. **Web services description language (wsdl) 1.1.** http://www.w3.org/TR/wsdl, 2001. pp 15, 17, 72, 75 et 77.