ISSN:2229-6093

IJCTA

N Adadi et al, International Journal of Computer Technology & Applications,Vol 7(5),636-642

Formal Specification of Web Services Composition Using LOTOS

N. Adadi [1], M. Berrada[2], D. Chenouni [3]

IPI (Laboratoire d'Informatique et de Physique Interdisciplinaire) Laboratory, ENS, Sidi Mohamed ben Abdellah University, Fez, Morocco.

nouhaadadi@gmail.com[1], mohammed.berrada@gmail.com[2], d_chenouni@yahoo.fr[3]

Abstract

The composition of Web services, that is the combination of several services to obtain new features, becomes more and more popular and present a necessary stage for the realization of the collaboration inter-companies (B2B). To implement this collaboration a developer has to elaborate a specification which allows the modeling of the global behavior of the system, to verify formally this model to assure the quality of the system then pass to the implementation of the composed service. In this paper we present a summary of our proposed approach of web services composition which is separated into three tasks: specification using BPMN notation, implementation using BPEL language and formal verification, then we focus on the task of verification and we propose an approach of translating the BPMN specification of services composition to a formal specification LOTOS which allows, using verification tool like CADP, to apply the behavioral properties and validate the system of Web services composition.

1. Introduction

Nowadays many enterprises publish their applications functionalities on the Internet. This new generation of applications allows greater efficiency and availability for business. In fact, more and more applications make functionalities available using a web service format. However there are many services around the web, each one, taken alone, has a limited functionality. In many cases, a single service is not sufficient to respond to the user's request and often services should be combined through services composition to achieve a specific goal. In other words, from a user perspective, this composition will continue to be considered as a simple service, even though it is composed of several web services. Many researches focus on Web services composition [1, 2]. The process of service composition can be separated into three tasks: specification, implementation and verification. Many various languages such as BPMN (Business Process Modeling Notation) [3] and BPEL (Business Process Execution Language) [4] have been proposed to specify and implement Web services composition. However, their lack of well-defined formal semantics does not support formal verification. As a consequence, the validation of Web service composition remains a complicated task.

Formal verification establishes an essential asset to prove that the design of a system is correct and avoid such problems. During the elaboration of the specification of a composed web service, the developer creates a set of blocks performing the behaviors required to model the global functioning. However, there are subtleties in the specification that make the composed service will not necessarily have the expected behavior once implemented. The developer must therefore be able to ensure that the specification is correct before proceeding to the implementation. The tools of formal verification can answer this need but these tools check only specifications described in formal language which requires the translation of system's modeling to formal specification.

In this paper we present a summary of our proposed approach of web services composition, this approach is conceived for the specification, the formal verification and the implementation of composed web service, then we focus on the task of formal verification and we propose a new approach of translating the specification of services composition with BPMN notation to a formal

specification LOTOS [5] which allows, using verification tool, to apply the behavioral properties and validate the system of Web services composition.

The layout of this paper is as follows. The second section discusses various approaches used for verifying formally the composition of web services. The third section is devoted to the LOTOS language, its basics and operations. The fourth section present a summary of our proposed approach of web services composition. Finally, the fifth section presents an example of web service composition, in this section we use this example to transform the BPMN model of the system to formal specification LOTOS. The conclusion and future work are presented in section six.

2. Formal verification

Formal verification is the systematic process of verifying, through exhaustive algorithmic techniques, that an implementation is in accordance with its specification. Using formal verification, all possible execution paths are analyzed mathematically without requiring the preparation of test cases. The developer describes simply the properties according to the system functionalities whish he wants to prove and he leaves the formal verification tools explore exhaustively all possible execution paths on the mathematical representation.

2.1. Formal verification tools

Verification tools support in the inputs model to verify described in formal language and a set of behavioural properties defined by the developer based on the capabilities of the system he wants to test. In the output these tools return the result (true or false) is that the property is checked or not in the model and a set of proposed correction. To do this, the current tools utilize mathematical logic.

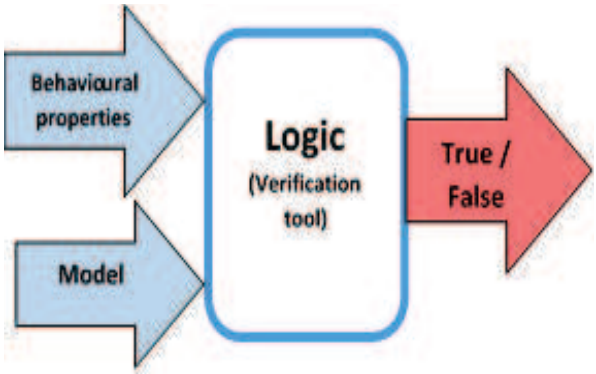

Figure 1. Classic formal verification structure

2.2. Formal verification of web services composition approaches

Many works realized on the specification and formal verification of the composition of web services using various languages and tools.

2.2.1. Using state transition systems. The state transition system STS is a well-known model in the field of formal specification, it is represented graphically in the form of directed graph, consisting of a set of states or nodes, a set of actions and a set of transitions between states labeled. The authors in [6] focuses on modeling of a service composed by an STS. Then translate this system to a Promela specification which is a formal language similar to C language. Once this specification is obtained, the tool of verification SPIN [7] allows to verify properties expressed in LTL (LinearTemporal Logic).

2.2.2. Using Petri Nets. As part of the Semantic Web, it is common to use the services described with DAML-S language [8]. The authors of [9] use this language in order to compose web services. Then they transform a DAML-S description of a Web service into a Petri net, which is a system formally defined [10] and presented by a directed, connected and bipartite graph. Finally, the use of a model as the Petri net, allows the use of a validation tool KarmaSIM for applying a simulation and validation services.

2.2.3. Using the process algebra. The process algebra is formal description language for specifying competitive systems we can cite CCS [11], LOTOS [5] and π-calculus [12]. In [13] the authors use the notation CRESS (Chisel Employing Systematic Representation Specification) for specification of web services composition. This notation is then converted into LOTOS specification, this allows then an analysis with a tool like LOLA. In [14] the authors use the UML activity diagram to model the composition of web services, this model is then transformed into LOTOS specification, it allows then the verification of behavioral properties using the CADP tool [15].

The Process algebra constitutes an approach among the newest and most useful ones for checking competitive systems, as the composed web service is a competitive system, we will use in our proposed approach the LOTOS language for specifying the processes generated from the composed web services, thus verifying this process by using an adequate verification tool like CADP.

3. An overview of LOTOS language

LOTOS (Language Of Temporal Ordering Specification) [5] is a formal description developed within ISO (International Standards Organization) for the specification of open distributed systems. LOTOS is based on temporal ordering of events and

process algebraic methods. It consists of two parts: a part for the description of data and operations, based on abstract data types, and a part for the description of concurrent processes, based on process calculus.

The two parts of LOTOS use various behaviour operators, These operators are summarized in Table 1 where G refer to a gate (channel of communication between processes), X to a variable, P to a process, S to a sort, V to a value and B a behaviour.

Table 1. LOTOS behaviour operators

Behaviour	Operator
inaction	stop
Action Prefix	G !V ?X:S ; B
Choice	B_1 [] B_2
Conditional	[E] -> B
Parallel composition	B_1 |[G_1,…,G_n]| B_2
Interleaving	B_1 ||| B_2
Successful termination	exit
Sequential composition	B_1 >> B_2
Process call	P [G_1,…,G_n] (V_1,…,V_m)

A more detailed introduction to LOTOS can be founding in [5]. In this paper, LOTOS was chosen for the specification of service composition workflows because of its ISO standardization, its high expressiveness its formalism and the existence of validation tools that support it such as CADP.

4. Proposed approach of web services composition

The proposed approach is conceived for the specification, the formal verification and the implementation of composed web service. The figure 2 shows the steps involved in the proposed development processes to better understand how to proceed.

Once the requested services are selected by the directory we pass to the specification stage. At this level we propose a modeling based on MARDS model (Multi-Agent Reactive Decisional System) [16], and using the BPMN notation. The MARDS model, constitutes an approach among the newest and most useful ones for the composing and modeling of complex system such as the automated systems of production, the mobile systems and organizational system [17] [18]. We have used this system in our proposed approach because it allows to model the composition of services in a simple and powerful way, and in well-structured architecture. The BPMN notation, is a modeling language, it is more adapted to the domain of the Web services, legible and sufficiently precise and expressive to allow the generation of executable code from it. We have used this notation for modeling the processes generated from the composed web services on orchestration mode.

As it is better to detect errors as early as possible in the cycle of development, from the specification stage, the next step is the formal verification of our proposed model. The model of the system and behavioral properties described by the developer must be represented by a formal language so that they can be interpreted by formal verification tools which gives the result of verification. Our specification is described by the BPMN notation, but this language is often criticized for its lack of formality. One proposed solution is to transform the BPMN model into formal specification. Any formal specification language is susceptible to agree but we propose the use of the process algebra LOTOS which has the advantage of being supported by free formal verification tools such as CADP toolbox. Due to CADP, it is possible to validate automatically the behavioral properties. In case where errors are detected, the developer is responsible for correct and refine its model to arrive at a model proven correct. When the composition model is validated, the next step is the implementation of the system by generating BPEL 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.

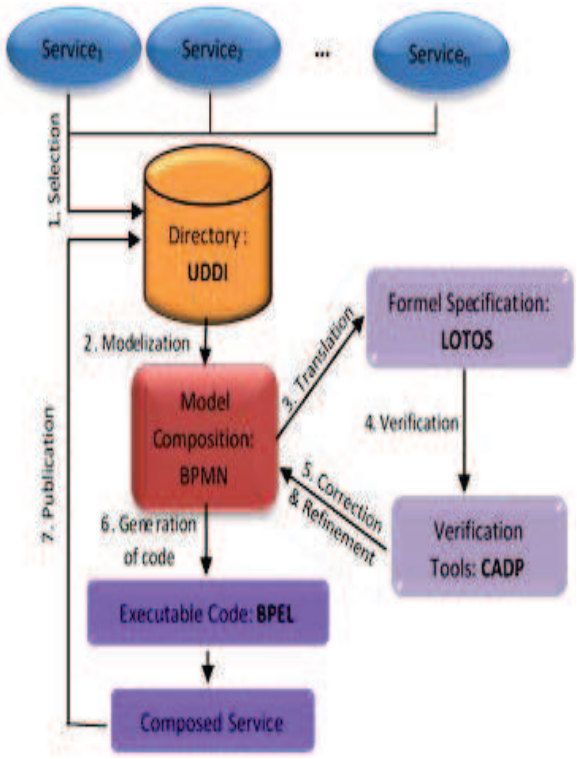

Figure 2. Proposed approach of web service composition

The phases of modeling of web services composition using BPMN and generation of code executable BPEL are already covered in previous works [19][20]. The step that remains to be processed is the formal verification, in this paper we propose to focus on this step and precisely the part of translating the BPMN model into formal specification LOTOS.

5. Example: Modeling and LOTOS specification of "travel organisation" composite web service

5.1. Example of Web Services Composition

As an illustrative example, we will consider in this work an online Travel Organization problem. This scenario is a typical web services composition problem. As far as creating the "*Travel Organization*" service, we can use five services ("*Airefare*", "*Car*", "*Hotel*", "*Payment_Detail*" and "*Bank*") that will internally execute the online Travel Organization, each one executes a task.

5.2. Modeling of "Travel Organisation" composite web service

5.2.1. Modeling of Web Services Composition process. The modeling part was the target of other works already published [19] [20]. I present only the summary and the result of these works without getting into details. Our web services composition modeling approach is based on multi-agent reactive decisional system MARDS [16] which is a special type of multi-agent systems characterized by a set of operations, functions, and a well-structured hierarchical architecture that allows the composition of web services in a simple and powerful way. The application of the concepts of MARDS model on our example allows to have the following structure of the composition system by creating communication interfaces and new intermediate and main services.

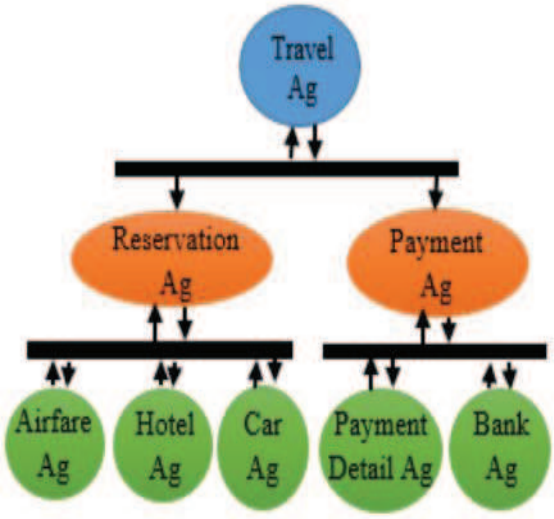

Figure 3. Structure of composition of the Web services

5.2.2. Business model of the services composition:

This structure of composition can be modeled using BPMN notation [3]. The Figure 4 displays the business model of the composition services structure. The action "*A_Online Travel"* received by "*Travel"* component generates two decisions {*D1_Reserve; D2_Pay*}. Each decision corresponds to a several sub-actions received by "*Reservation"* component {*D1_ Reserve; A_ Reserve*} and by *"Payment"* component {*D2_Pay, A_ Pay*}. Every sub-action received by any composite component will be realized and modeled as a sub-process.

The sub-action "*A_ Reserve"* received by the *"Reservation"* component generates in parallel three sub-decision {*D1_ReserveAirfare*; *D2_ ReserveHotel*; *D3_ ReserveCar*}. The first sub-decision "*D1_ ReserveAirfare"* generates the {*A_ ReserveAirfare*} action for "*Airfare*" basic component. The second sub-decision "*D2_ ReserveHotel"* generates the "*A_ ReserveHotel*" action for *"Hotel"* basic component. The third sub-decision "*D3_ ReserveCar"* generates the "*A_ ReserveCar*" action for *"Car"* basic component. The competition of the three sub-decisions corresponds to the sub-process of the "*A_ Reserve"* sub-action.

The sub-action "*A_ Pay"* received by the *"Payment"* component generates in sequence 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.

ISSN:2229-6093

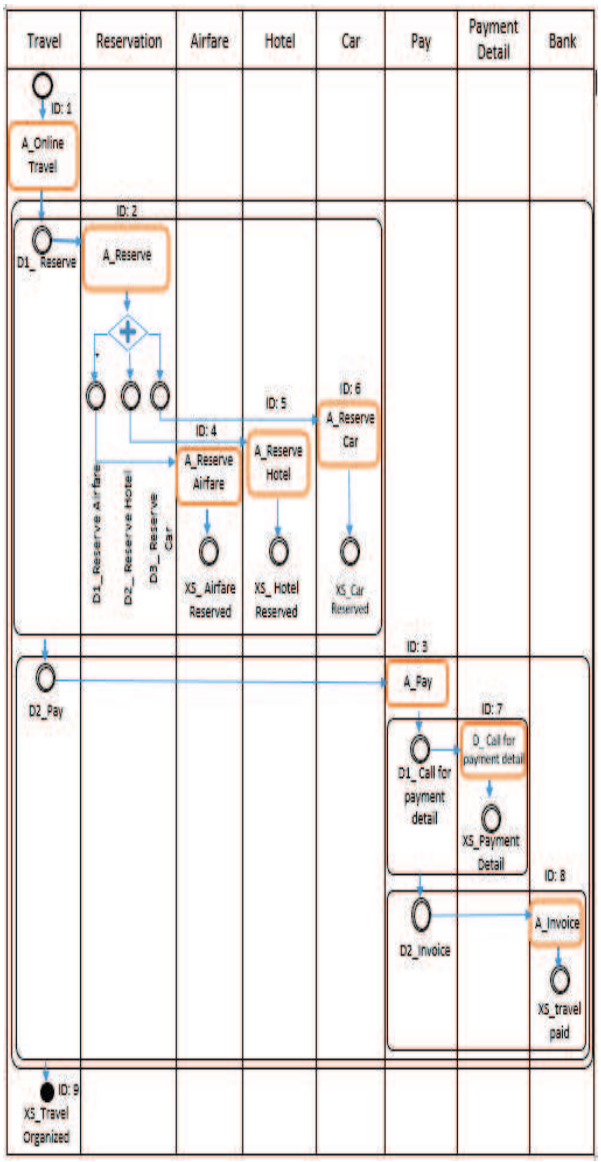

Figure 4. Business model of Services Composition example

5.2. Translation of BPMN modeling to LOTOS formal specification

To translate the BPMN notation depicted in Figure 4 into LOTOS we are going to follow these steps:

- Define a process for each step of the activity (including initial and final nodes). In our example the processes are ("*Init*", "*Travel_Organization*", "*Reservation*", "*Payment*", " *Reservation_Airefare*", " *Reservation_Car*", " *Reservation_Hotel*", "*Payment_Detail*", "*Bank*" and "*Final*"). Each process is defined by a set of behaviors.
- Assign an identifier (integer) to each of process. The identifiers (ID) are already specified in Figure 4 for a better understanding.
- Define the gates which are the channels of communication between processes. The peculiarity of our modeling with the SMARD model is that communication between services is done via the communication interfaces that receive and send actions and decisions, so we can consider these interfaces as processes (BUS0, BUS1 ... BUSn). The actions and decisions sent and received by the services and communication interfaces are considered LOTOS gates (SENDi, RECVi) when i between 0 and n. Indeed services processes can communicate with each other through these gates, thanks to BUS0, BUS1…BUSn processes.
- Define the operations between processes, in our example all service processes are executed concurrently using the ||| operator, which means that they are independent and they do not communicate directly with each other, but they use BUSi process. Note however that the |[SENDi, RECVi]| operator is used to synchronize the service processes with the BUSi process through the gates SENDi and RECVi, when i between 0 and n.
- Identify the control-flow patterns in the workflow in order to provide a definition (implementation) for each process.

The instantiation of the processes in LOTOS is provided in figure 5.

```
Specification Travel_Organization [SEND0, SEND1,
SEND2, RECV0, RECV1, RECV2]:noexit
Behaviour (
(Init [SEND0, RECV0] (0)
|||
Travel_Organization[SEND0, RECV0] (1)
|||
Reservation [SEND0, RECV0] (2)
|||
Payment [SEND0, RECV0] (3) )
[SEND0,RECV0]BUS0 [SEND0,RECV0]
(Reservation [SEND1, RECV1] (2)
|||
Reservation_Hairfare [SEND1, RECV1] (4)
|||
Reservation_Hotel [SEND1, RECV1] (5)
|||
Reservation_Car [SEND1, RECV1] (6) )
[SEND1,RECV1]BUS1 [SEND1,RECV1]
(Payment [SEND2, RECV2] (3)
|||
Payment _Detail [SEND2, RECV2] (7)
|||
Bank [SEND2, RECV2] (8)
|||
Final [SEND2, RECV2] (9) )
[SEND2,RECV2]BUS2 [SEND2,RECV2]
)
 Where (* Implementation of processes *)
)
```

Figure5. Processes instantiation in LOTOS

ISSN:2229-6093

Then we pass to define the implementation of processes.

The Init process (*Id:0*) merely starts the Travel_Organization process (*Id:1*). As a consequence, it uses the sequence pattern before exiting, as defined in figure 6.

```
Process Init [SEND0,RECV0] (ID:Int) : exit:=
Sequence [SEND0,RECV0] (Id,1)
>>exit

Where (*Definition of Sequence process*)

endproc
```

Figure 6. LOTOS specification for Init process

The Travel_Organization process waits for a RUN message from Init before starting. After that, it realizes an sequence between Reservation (*Id:2*)and Payment process (*Id:3*), as defined in figure 7.

```
Process Travel_Organisation [SEND0,RECV0] (ID:Int) : exit=
RECV0 ! Id ! 0 ! RUN ! Void;
Sequence [SEND0,RECV0] (Id,2)
>> Sequence [SEND0,RECV0] (Id,3)

>>exit

Where (*Definition of Sequence process*)
endproc
```

Figure 7. LOTOS specification for Travel_Organization process

The Reservation process waits for a RUN message from the Travel_Organization process before starting concurrently the Reservation_Hairfare (*Id:4*) and Reservation_Hotel (*Id:5*) and Reservation_Car (*Id:6*) processes, thus realizing a parallel split pattern. The corresponding specification is provided in figure 8.

```
Process Reservation [RECV0, SEND1,RECV1] (ID:Int) : exit=
RECV0 ! Id ! 1 ! RUN ! Void;
PrallelSplit [SEND1,RECV1] (Id,insert(6,insert(5,insert(4,{}))))
>>exit

Where (*Definition of ParallelSplit process*)
endproc
```

Figure 8. LOTOS specification for Reservation process

The Payment process waits for a RUN message from the Travel_Organization process before starting sequentially the Payment_Detail (*Id:7*), Bank (*Id:8*) and Final (*Id:9*) processes. The corresponding specification is provided in figure 9.

```
Process Payment [RECV0, SEND2,RECV2] (ID:Int) : exit=
RECV0 ! Id ! 1 ! RUN ! Void;
Sequence [SEND2,RECV2] (Id,7)
>> Sequence [SEND2,RECV2] (Id,8)

>> Sequence [SEND2,RECV2] (Id,9)
>>exit

Where (*Definition of Sequence process*)
endproc
```

Figure 9. LOTOS specification for Payment process

The Final process waits for a RUN message from the Bank process before exiting.

```
Process Final [SEND2,RECV2] (ID:Int) : exit:=
RECV2 ! Id ! 8 ! RUN ! Void;
>>exit
endproc
```

Figure 10. LOTOS specification for Final process

In Sequence process, an activity identified by id_dst should be executed after the completion of the activity identified by id in the workflow, We say that both activities are then executed sequentially. The LOTOS specification is provided in figure 11.

```
process Sequence [SEND , RECV] (Id:Int ,Id_dst : Int ):
exit :=

SEND ! Id_dst !Id !RUN ! void; exit

endproc
```

Figure 11. LOTOS translation for sequence pattern

Finally, the ParallelSplit process, The identifiers of the activities (Ids_dst) to be executed in parallel are passed in parameters to the process as a set of integers (IntSet). The process needs to iterate over this set and send a RUN message to each activity identified in the set. However, recursion is the only way to realize cyclical behavior in LOTOS. As a consequence, the ParallelSplit process is calling itself recursively and removing already processed *Ids* from the set in order to iterate over it.

```
process ParallelSplit [SEND , RECV] (Id:Int , Ids_dst : IntSet ) : exit :=
[ empty (Ids_dst )] -> exit
[]
[not (empty (Ids_dst ))] ->
(let DestInt = pick( Ids_dst ) in
SEND ! Dest !Id !RUN ! void;
ParallelSplit[SEND , RECV ](Id ,remove (Dest , Ids_dst )))
endproc
```

Figure 12. LOTOS translation for parallel split pattern

6. Conclusion

In this paper we presented a summary of our web services composition approach including the specification the formal verification and the generation of executable code. We have already realized in other works the stages of specification and implementation [19] [20], and in this paper very importing parts of verification that is the translation of a BPMN model to a LOTOS program, this critical part allows later use of the CADP verification tool that takes charge of applying the properties of the model and give the result of verification.

References

[1] S.Pan and Q.Mao, "Case Study on Web Service Composition Based on Multi-Agent System", Journal of Software, Vol. 8, n. 4, April 2013.

[2] M. Bakhouya and J. Gaber. Service composition approaches for ubiquitous and pervasive computing environments: A survey. Agent Systems in Electronic Business, Ed. Eldon Li and Soe-Tsyr Yuan, IGI Global, (978-1-59904-588-7):323–350, 2007.

[3] BPMI.org. (2006). Business Process Modeling Notation Specification. OMG Final Adopted Specification

[4] OASIS Standard. Web services business process execution language version 2.0, April 2007.

[5] T. Bolognesi and E. Brinksma. Introduction to the iso specification language lotos. The Formal Description Technique LOTOS, pages 23–73, 1989.

[6] X. Fu, T. Bultan, and J. Su. Analysis of interacting BPEL web services. In Proc. Of the13th International World Wide Web Conference (WWW'04), USA, 2004.

[7] Gerard J. Holzmann. The SPIN MODEL CHECKER - Primer and Reference Manual. Addison-Wesley, Pearson Education, september 2003.

[8] David Martin, Mark Burstein, Grit Denker, Jerry Hobbs, Lalana Kagal, Ora Lassila, and Katia Sycara. DAML-S (and OWL-S) 0.9 draft release, 2003.

[9] Srini Narayanan and Sheila McIlraith. Analysis and simulation of web services. Comput Networks, 42(5):675–693, 2003.

[10] J. Peterson. Petri Net Theory and the Modeling of Systems. Prentice Hall, Englewood Cliffs, 1981. p 26.

[11] R. Milner. Communication and concurrency. International Series in Computer Science, 1989. p 30– 48.

[12] R. Milner, J. Parrow, and D. Walker. A calculus of mobile processes i & ii. Information and Computation, 100(1):1–77, 1992. p 23-30.

[13] Kenneth J. Turner. Formalising web services. In Proc. of Formal Techniques for Networked and Distributed Systems, Taiwan, 2005, p 473–488.

[14] C. Dumez, Approche dirigée par les modèles pour la spécification, la vérification formelle et la mise en œuvre de services Web composés, 2010.

[15] J. C. Fernandez, H. Garavel and A. Kerbrat, Cadp-a protocol validation and verification toolbox. In Proc. of International Conference on Computer Aided Verification, 1996.

[16]M.Berrada, B.Bounabat, , "Modeling and Simulation of Multi-Agent Reactif Decisionnal Systems using Business Process Management Concepts" , IRECOS Vol.2 N.2, pp. 159-169, March 2007.

[17]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.

[18]A. Aaroud, S. E. Labhalla, and B. Bounabat, "Design of GSM handover using MARDS model", Proceedings of International Conference for Information Technologies and Application (ICITA), 2004, Harbin, China.

[19]N. Adadi, M. Berrada, D. Chenouni, B.BOUNABAT "Multi-Agent Architecture for Business Modeling of Web Services Composition based on WS2JADE Framework", IRECOS ,Vol 9,no 10, 2014.

[20]N.ADADI, M.BERREDA, D.CHENOUNI, B.BOUNABAT " Modeling and Simulation of Web Services Composition Based on MARDS Model". In Proc. of International Conference on Intelligent Systems Theories and Applications (SITA) Rabat, Morocco, 2015.