Patterns for Modelling and Composing Flexible Workflows from Cloud Services

Imen Ben Fraj, Yousra BenDaly Hlaoui and Leila Jemni BenAyed
Research Laboratory in Technologies of Information and Communication and Electrical Engineering (LaTICE), Tunisia
Institute of Sciences and Techniques of Tunis, Tunis, Tunisia

Keywords: Cloud Service, Flexible Workflow, Flexibility Patterns, Functional Specification, BPMN Model, Behavioural Specification, State-chart Diagram.

Abstract: In this paper, we propose a Model-Driven Approach for the specification and the execution of cloud service flexible workflow applications. We define two flexibility patterns based on BPMN that deals with changes of resource requirements for workflow. The workflows are built on an abstract level, using a BPMN model for the specification of the cloud service workflow structure based on flexibility patterns, and the state-chart diagram for the specification of the cloud service workflow behaviour. The execution process is supervised by a control system which is responsible for making decisions on the execution of the workflow based on the behaviour defined by the state-chart diagram.

1 INTRODUCTION

1.1 Motivation

Nowadays, Cloud computing (Dillon and Chang, 2010) has emerged in the distributed computing community, it is a new delivery model for IT services based on Internet protocols. It has become an important topic in both industrial and academic areas. The basic goal of the cloud computing is to make a better use of distributed resources and be able to solve large scale computation problems. A cloud service is a web service that provides a set of well defined cloud interfaces and follows cloud specific conventions. These services constitute a powerful basis for modern and scientific applications development. In order to enable users to compose their applications without taking care of the lower level details, the concept of cloud workflow has emerged as a method for modeling complex and scientific application (Yubin, Zeye, Zewei and Ximing, 2013). The problem of building such applications requires composing and orchestrating appropriate services which are most vulnerable and it is frequently a delicate task. This is due to the very large number of available services and the different possibilities for constructing a flexible workflow from matching services. Flexible workflows (Yubin, Zeye, Zewei and Ximing, 2013) is a solution to fasten information system development in distributed and dynamical environment like the Cloud. One of the expected facilities of cloud is flexibility at different levels. Therefore, we propose in this paper a model driven approach for developing and executing flexible workflow applications composed of cloud services. Recently, several solutions have been proposed to model applications from cloud services such as works presented in (Amziani, Melliti and Tata, 2013; Fengyu, Ying, Zheng, Wei and Xilong, 2015). However, the proposed solutions need interaction with user and guidelines or rules in the design of the composed applications. In consequence, the resulting source code is neither re-usable nor it promotes dynamic adaptation facilities as it should. For applications composed of cloud services, we need an abstract view not only of the offered services but also of the resulting application. This abstraction allows in one hand the reuse of the elaborated application and on the other hand reduces the complexity of the composed applications. It has been proven from past experiences that using structured engineering methods makes easy the development process of any computing system and reduce the complexity when building large cloud applications. To reduce this complexity and allow the reuse of cloud service applications, we adopt a MDA approach. The MDA (Gronmo and Jaeger, 2005) approach developing starts with defining high-level models in BPMN

Fraj, I., BenDaly Hlaoui, Y. and Jemni BenAyed, L.
Patterns for Modelling and Composing Flexible Workflows from Cloud Services.
DOI: 10.5220/0006772903060313
In *Proceedings of the 20th International Conference on Enterprise Information Systems (ICEIS 2018)*, pages 306-313
ISBN: 978-989-758-298-1

(Allweyer, 2010), defines conversion rules from BPMN to target platform BPEL4WS (Shen, Grossmann, Yang, Stumptner, Schrefl and Reiter, 2007), and then use code generation to derive much of the implementation code for desired platform.

1.2 Our Contribution

We propose a model driven approach to built flexible cloud service workflow using BPMN to specify the functional view of the flexible workflow, and the state-chart diagram of UML (Group, O.M., 2005). To describe the behavioural view of this workflow. We use our model driven approach to help a user to compose and develop her/his cloud service workflow applications in an abstract way using BPMN. Then the built model will be transformed into a specific cloud service workflow platform such as BPEL platform. In the modelling step, we guide the user to compose her/his workflow using a set of flexibility specified patterns that we have developed according to the cloud service characteristics. Hence, we propose an interactive approach for modelling and executing cloud service workflows. In addition, we propose a control system which controls the behaviour of the BPEL engine based on the properties of flexibility defined in a state-chart diagram using the ECA rules. Figure 1 presents the architectural view of the proposed approach. At the first step of the approach, the user specifies its cloud service workflow, by modelling a composition request using a BPMN model (functional view). Another specification of the cloud service workflow behaviour using a state-chart diagram (behavioural view) should be defined automatically. The provided request will be refined by the composition system to build the flexible workflow from available cloud services. Before being executed, two transformations should be produced, the first one is the transformation of the BPMN model into a running platform model like BPEL4WS, the second is the transformation of the state-chart diagram into a running platform using a control system.

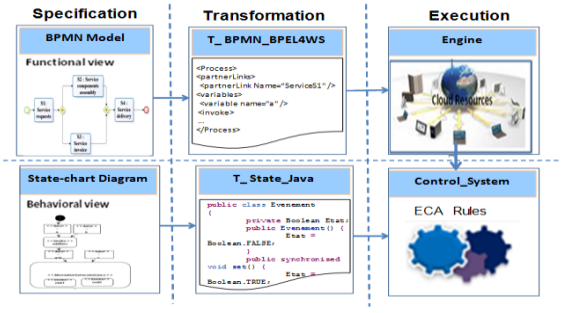

Figure 1: Architectural view of the approach.

1.3 Paper Reminder

This paper is organized as follows. Section 2 presents the related work. Section 3 defines the flexible workflow and the properties of flexibility for cloud service workflow. Section 4 details the proposed approach for modelling and running cloud services workflow. Section 5 presents the flexibility patterns. Section 6 details the behavioural specification through the state-chart diagram. Section 7 presents a transformation from BPMN model to BPEL4WS. Section 8 illustrates the specification process based on the proposed BPMN model and presents some experiments results. Finally, section 9 concludes the paper and proposes areas for further research.

2 RELATED WORKS

Several works and researches were carried out in the field of modelling of flexible workflow of cloud services like works presented in (Amziani, Melliti and Tata, 2013; Fengyu, Ying, Zheng, Wei and Xilong, 2015). In (Fengyu, Ying, Zheng, Wei and Xilong, 2015) the authors propose a flexible UML-based workflow model that is able to exhibit the architecture of workflow engine and to adapt to the changes of business process. They were based on UML diagrams to describe the flexible workflow. This approach would have been better if the composition were automatically elaborated since the number of available services is in increase with the existence of several forms and manners to compose such services. In (Amziani, Melliti and Tata, 2013), the authors, propose a formal model elasticity for service-based business processes (SBP). In this model, processes are defined as Petri nets. Then, elasticity operations (duplication and consolidation) are defined. Each service is represented by at least one place (the set of places of each service are related with an equivalence relation). The transitions represent calls, transfers between services according to the behavioural specification of the SBP.

The originality of our contribution, relatively to this work, is that first we save the user from the dynamic refinement and execution as we propose an MDA approach which separates the specific model from the independent model. Second, we facilitate the composition of flexible workflow as we guide the user to compose her/his workflow using flexibility patterns. Third, we consider the properties of flexibility in the specification of the workflow in a more natural way for the human user as we define the functional and the behavioural views.

3 CLOUD WORKFLOW FLEXIBILITY

The flexibility is the capability to implement changes of the requirements in the business process model and instances by changing only those parts of the business process model and instances that reflect the change. Flexibility (Nurcan, 2008) has been the focus of many researches (Regev and Wegmann, 2005; Rosemann and Recker, 2006; Saidani and Nurcan, 2006; Schmidt, 2005). There are many definitions of the flexibility in literature (Shi and Danies, 2003). It is defined in (Regev and Wegmann, 2005) as "the ability to yield to change without disappearing". Processes flexibility means fast reactivity to internal and external changes. It reflects the easiness to make evolve business process schemes (when required). Flexibility is also reflected by the ability that the support systems have to take into account business changes. Thus, flexibility refers to the executable ability to flexible process definition of workflow management system (WFMS). Flexibility of workflow means the dynamic generation and modification on definition of process instances during execution. Workflow have to provide means to suit the flexibility and adaptability requirements at any given time. Flexible workflow is a solution to fasten information system development in distributed and dynamical environment like the Cloud. The Flexibility of workflows allows users to maintain cloud-based workflows, depending on the requirements of the particular job or end customer.

3.1 Flexibility Properties

In this section, we present the properties of flexibility, it can be achieved by three ways in which the routing of cases along workflow tasks or cloud services can be changed (Van der Aalst, 2001):

- *Extend*. Each cloud service has a maximal capacity of treatment over that the QoS of the service decrease and we risk to have a breakdown, thus the solution is adding new tasks which (1) are executed in parallel, (2) offer new alternatives, or (3) are executed in-between existing tasks.
- *Replace.* An instance of cloud service could be unavailable, so a task is re-placed by another task or a subprocess (i.e., refinement), or a complete region is replaced by another.
- *Re-order*. A cloud service instance cannot follow the order defined in the workflow to reduce the execution time. Changing the order in which tasks are executed without adding new tasks, e.g., swapping tasks or making a process more or less parallel.

Thus, flexibility in cloud service workflows remains a solution for optimizing the cloud application performance and running costs of cloud services.

4 THE PROPOSED APPROACH

The purpose of our approach, is to allow the specification and the execution of flexible workflow applications composed from cloud services. The specification is based on semantic composition of cloud services and on flexibility properties that should be processed during the workflow run time. There are three objectives to achieve by proposing an architectural approach for developing distributing computing applications:

1. To consider the flexibility properties in the specification of the cloud service workflow,
2. To control the flexible workflow execution,
3. Ease the development of such applications.

To reach these objectives, our approach follows the model driven architecture by separating the platform development model from the platform specific model. Thus, we propose to specify the workflow model by a functional view using BPMN model (Allweyer, 2010). The model provides an abstract and understandable description of the cloud application which is obtained from the integration of cloud services in a workflow. Once a model is created, it will be transformed into the specific model to be implemented and executed. To resume, our approach is based on three steps, the first one is the specification of the cloud service workflow, the second is the meta-model transformation and the latter is the execution of the flexible workflow application. These steps will be detailed in the next sections.

4.1 Specification of the Cloud Service Flexible Workflows

The flexible workflow is modelled by a BPMN model based on different patterns of composition of workflows. Each activity represents the cloud service's operation that should provide the result. At the same time, the specification of the cloud service workflow behaviour is also defined using a state-chart model for processing the flexible properties. The composition is described in an abstract level using BPMN model (Allweyer, 2010) by assisting the user to compose her/his flexible workflow using the

appropriate pattern of flexibility. The workflow model identifies the resource from one depicted cloud service's operation to the next to build and compose the whole application. In this paper we emphasize upon the modelling of composed flexible workflows only from Cloud services and not from sub-workflows.

4.2 Meta-Model Transformation

This step is based on two transformations. The first, transforms the BPMN workflow model into a specific workflow model which will be executed using the BPEL4WS engine. While the second, translates the state-chart model describing the behaviour of each cloud service into a control system implemented language such as ADA (Woodruff and Van Arsdall, 1998).

4.3 Execution of the Flexible Workflow Application

The workflow description document is sent to a workflow engine that produces implementation code for handling control-flow and data-flow. An execution engine, such as BPEL4WS engine, executes different workflow activities which are specified in a workflow execution document in the correct order and with their required input and output data. We extend BPEL construct to support flexibility patterns that we have defined. Before execution, the engine consider the preliminaries and requirements mentioned by the control system, the latter controls the execution process based on the behaviour described by the state-chart diagram. Thus, the real time meta-model transformation is achieved. We will present the different steps of the approach through an example (Figure 2) of an online computer shopping service (Amziani, Melliti and Tata, 2013) composed of four services. Figure 2 models the normal workflow functional behaviour, without considering the flexibility. However, we can meet some anomalies during this behaviour. These anomalies should be considered in the functional model to predict any abnormal behaviour.

In fact, the functional behavior of the workflow depends on each cloud service behavior. We resume these anomalies as follows:

- **Anomaly 1:** Each service has a maximal capacity of treatment over what the QoS of the service decrease and we risk to have a breakdown (Amziani, Melliti and Tata, 2013)

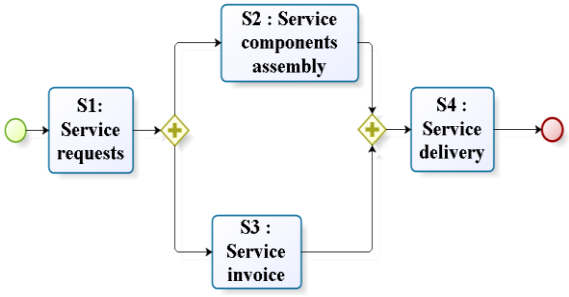

Figure 2: Workflow of an online shopping service.

- **Anomaly 2:** In case of quality satisfaction, we are with another copy of service, already created and useless (Amziani, Melliti and Tata, 2013).
- **Anomaly 3:** At any state of the execution process, we need a required resources, if it is not available, the process of the execution could be suspended and risks to fail.

The use of the flexibility properties is the solution for these anomalies. We should update the initial workflow model (e.g. the model of Fig. 2.) to consider each of these properties based on the behaviour parameter of the cloud service. Based on the properties of flexibility we define three modelling solutions:

- **Solution 1 (Duplication):** This solution consists in duplicating (Amziani, Melliti and Tata, 2013) the service which had an excess treatment of requests. This copy is created to treat the overloaded service.
- **Solution 2 (Delete):** If the number of the requests decreases, then it is necessary to delete the new service already created (Amziani, Melliti and Tata, 2013).
- **Solution 3 (Add Resource):** This solution consists in adding new resource when the previous resource is unavailable, after a delay of waiting, we should add the new resource, to avoid the suspension of the execution process.

These modelling solutions are described graphically by BPMN patterns, that we have developed to define and specify automatically these situations in the workflow model, in order to help the user to compose her/his flexible workflow. We add two patterns the *Duplicated Services Pattern*, and the *Alternative Services Pattern*, which should preserve the initial semantics of the built workflow. The first pattern is used to models the duplication and the delete solutions. The second one describes the add resource solution.

5 SPECIFICATION OF THE FUNCTIONAL VIEW OF THE CLOUD SERVICE FLEXIBLE WORKFLOWS USING BPMN

In the modelling step we predict the behaviour of each cloud service based on its running history. This history is represented by the QoS data which are basically; time, cost and reliability. As the integration of the flexibility patterns is systematic, we are brought about defining a set of functions which allow the selection of the right pattern to use in the workflow model. In the following we define formally these functions and predicates which depend on the history of the implied cloud service.

5.1 Alternative Services Pattern

Formalization. When the cloud service of the workflow is unavailable, then we should replace its instance by another which is available, providing the same result and having the same semantic description. Let *response_time(s)* be a function, providing the time response of each operation's service. Another function is used, *max(s)* which present the maximum delay that the service could wait to be invoked. We define the predicate *alternative-service(s)* as follows:

Alternative-service(s): $S \rightarrow$ {true, false}

Alternative-service(s) = True if *response_time(s)*>max(response_time(s_i))
False else.

response_time : S→IR
s→t

max : $2^{IR} \rightarrow$ IR
(t1....ti) → t

with *response_time* ∈ QoS and *t* is the maximum of all the values. √i 1≤i≤n

Description. When the predicate Alternative-service is applied on the current activity of the cloud service, to be inserted in the workflow and it is evaluated as true, the activity is involved in the workflow using the pattern alternative-service.

Proposed Solution. In this solution the activity to insert is modeled as a composed super activity with a specified input data object and specified output data object (Figure 3). The super activity is stereotyped as *AlternativeServiceInstance* to indicate that its task may be accomplished by a set of alternative service's instances. These alternative service instances are described with sub-activities. The sub-activities shall be cloud service instances. It was up to decision mechanism of the workflow execution engine to choose which service instance in a such given workflow node is to be invoked and executed.

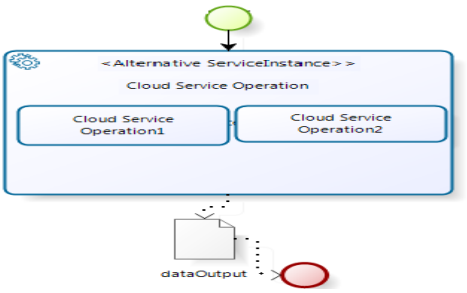

Figure 3: Alternative Service Pattern.

5.2 Duplicated Services Pattern

Formalization. When modeling workflows of cloud services, a specific matching based on semantic comparison could provide two or more different cloud services performing each of them the required operation. The Cloud registry could provide more than one operation able to produce the required result, the composition requires a specific pattern. Let *nb_req(s)* be a function, providing the number of queries of each service. Another function is used, *max_req(s)* which present the maximum number of queries that the service could be invoked. To capture this, we define a predicate *duplicated-service (s)* presented as follows:

duplicated-service (s): $S \rightarrow$ {true, false}

duplicated-service (s) = True if nb_req(*s*) > max -req (nb_req(*s*))
False else.

nb_req : S→IN
s→nb

max_req : $2^{IN} \rightarrow$ IN
(nb1....nbi) → nb

with *nb_req* ∈ QoS and *nb* is the maximum of all the values. √i 1≤i≤n.

Description. When *duplicated-service (s) = true*, we should duplicate the service *s* with another service s^d providing the same result of *s*.

Proposed Solution. Semantically, several services instances are invoked in parallel threads and will only wait for the adequate flow to finish. In Fig. 4, we present two same service's operations CloudService1Operation1 and CloudService1DuplicatedOperation1 providing the same output data DataOutput. In this solution the activity to insert is modeled as a composed super activity (Figure 4). The super activity is stereotyped as DuplicatedServiceInstance to indicate that its task

may be accomplished by one of two same service's instances if one of them is overloaded. If the first service is overloaded, we duplicate it to its copy (the second service), nonetheless, we just use the first service. Also, it was up to the decision mechanism of the workflow execution engine to choose which service instance in a such given workflow node is to be invoked and executed.

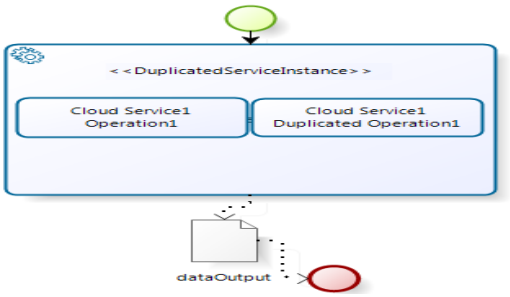

Figure 4: Duplicated Service Pattern.

Back to the initial workflow (Figure 2), several requests to its composing services might be generated, suppose that the workflow receives 100 requests. If there is not any flexibility adopted, the response time of the workflow will be equal to: 62s (processing time of a request) × 100 (number of requests) = 6 200s (Amziani, Melliti and Tata, 2013). The solution is to duplicate some services of the workflow, but not any services, we should not create a constraint on the consumption of resources. So back, to the initial workflow, the service *S2* has the maximum time and resource consuming. Thus we can have an overload on this service when it has a lot of calls, to avoid that, we model a flexible workflow (Fig. 5) based on the *Duplicated Service Pattern*. This pattern contains two services, the first is the service *S2*, and the second is the duplication of the same service named *S2-D* which will be invoked when it is overloaded. When we have an overload we duplicate the service as the number of requests, for this example, we will have 100 copies of *S2* in parallel. Thus the response time for processing the requests will be equal to 62s.

When we have an unload, the number of the requests decreases, then it is not necessary to duplicate the service, we just invoke the initial service for this example *S2*. The choice of which service instance, should be invoked and executed is leaded to the mechanism of the workflow execution engine. In this case, by applying to the *DuplicatedService Pattern*, we can solve the two first anomalies mentioned before.

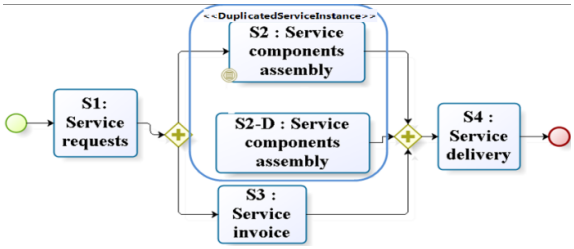

Figure 5: Flexible workflow.

To solve the last anomaly, the solution consists in adding a new resource when the previous resource is unavailable, after a delay of waiting, we should add the new resource, to avoid the suspension of the running of the workflow. To fulfil, we apply to the *Alternative Service Pattern*, which define a number of resources available for the same service (Figure 6). Back to the example, any services of the four services can be replaced by this pattern, because at any time, any service could not dispose of the required resources. To simplify the representation, we choice the service *S3* as the service which lacks of resources. The response time of the service *S3* is 10s, the maximum response time is 5s (here response_time(S3) > *max*(*response_time*)). Thus, we add another service named *S3-R* which provide the same resources for the *S3*.

Figure 6: Adding resource for service S3.

6 SPECIFICATION OF THE BEHAVIORAL VIEW OF THE CLOUD SERVICE FLEXIBLE WORKFLOWS USING SATE-CHART DIAGRAM

In order to control the execution of the flexible workflows, we should define the behaviour of each cloud service involved into the workflow. The definition of an adequate behaviour for dynamic modifications in different situations could avoid any eventual suspension of the workflow running process.

The state-chart diagram models the behaviour to recognize which services of flexibility patterns should be involved, and is it also supervised by a control system implemented by ADA (Woodruff and Van Arsdall, 1998). It models as well the actions that the control system should perform during the running time. During its processing time, a cloud service workflow could be in one of the three states: *waiting*, *running* and *terminated*. The two first states are composed of sub-states as shown in Fig. 7. (With E1: event overload, C1: nb_req> max_req, A1: Add a new instance of the cloud service, E2: event unavailable resource, C2: response_time> max_response_time, A2: Add a new resource for the cloud service, E3: event unload, C3: nb-req< nb-max, A3: delete the duplicated instance of the cloud service)

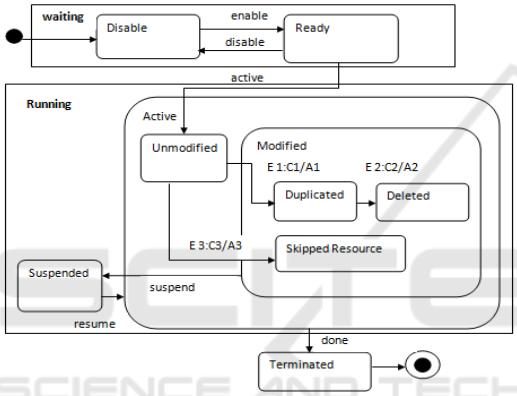

Figure 7: State-chart diagram.

7 TRANSFORMATION OF BPMN ALTERNATIVESERVICEINSTANCE PATTERN TO BPEL4WS

BPEL4WS language with its standard constructs does not support BPMN *AlternativeServicesInstance* pattern. This later models the semantics that states that the two operations are provided by two different services but they require the same resource. To fulfill this need, we have extended the BPEL4WS constructs by *alternativeservices* construct as BPEL4WS is an open source and an open system. To be supported by the BPEL4WS execution engine, we have defined for the *alternativeservices* construct a JAVA code which is added to the BPEL4WS execution engine JAVA code. The mapping pattern between *Alternative-services* pattern and BPEL4WS is presented next:

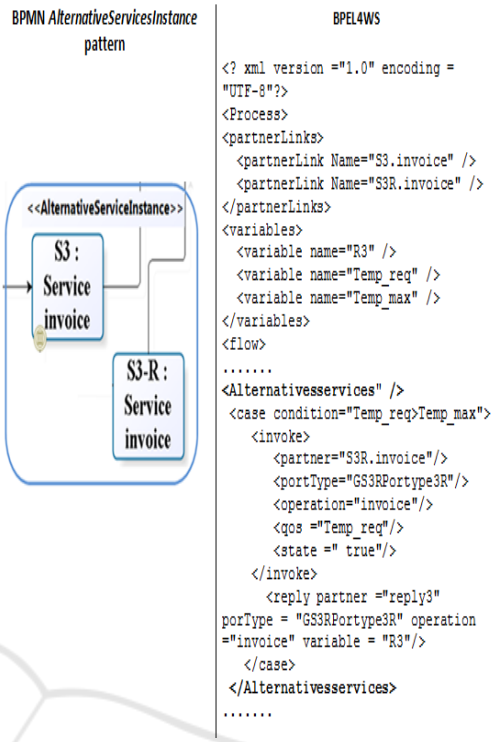

Figure 8: Transformation of BPMN alternative service Instance to BPEL4WS.

To execute this construct, the BPEL4WS execution engine will select the operation having the higher quality of service in term of cloud service response time or cloud service availability. Applying on the BPEL model of Fig.8 the transformation patterns defined between BPMN and BPEL4WS, we generate a BPEL4WS model.

8 EVALUATION OF THE FLEXIBILITY PATTERNS

In our experiment, we used an invocation scenario that represents a calls arrival on the workflow. This scenario applies for both *Alternative Service Pattern* and *Duplicated Service Pattern*. For each pattern, we generate a graph which represents all the possible evolution of the workflow in term of duplicating, deleting and adding resources. The analysis of this graph shows the use of the Alternative Service Patten (Figure 9 (a).), while the Duplicated Service pattern is applied in (Figure 9 (b).). Thus, based on these flexibility patterns, our approach helps the user to avoid any eventual suspension of the running workflow process and it reduces the cost of the running time.

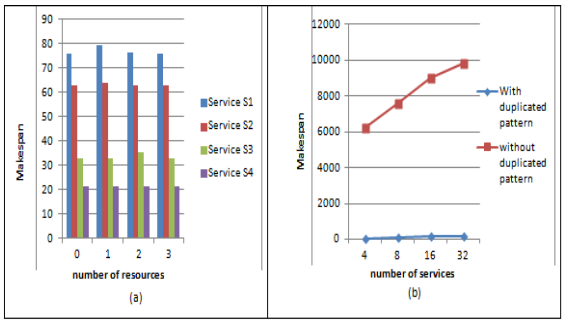

Figure 9: The evolution of resources and services using flexibility patterns.

9 CONCLUSION

We have proposed an MDA approach for the specification and the execution cloud workflow applications from cloud services. We have detailed a set of steps for integrating and matching, systematically, cloud services in a workflow. In addition, we have defined two patterns based on the properties of flexibility and predicates which is used in the execution process to depict the right cloud service to involve in the workflow. The approach shows also how the modelling proposed constructs are applied to model and represent a flexible workflow from cloud services by extending the BPEL4WS constructs. This process was illustrated under the example of an online computer shopping (Amziani, Melliti and Tata, 2013). As a proposal for further work, we propose to verify the semantic preservation of the different transformations.

REFERENCES

Allweyer, T., 2010. *BPMN 2.0: Introduction to the Standard for Business Process Modeling*.

Amziani M., Melliti, T., Tata, S., 2013. Formal Modeling and Evaluation of Service-Based Business Process Elasticity in the Cloud. Chapter *On the Move to Meaningful Internet Systems*: OTM 2013 Conferences. Volume 8185 of the series Lecture Notes in Computer Science, pp 21-38.

Dillon, T., Wu, C., Chang, E., 2010. Cloud Computing: Issues and Challenges. In *24th IEEE International Conference on Advanced Information Networking and Applications*, pp 27-33.

Fengyu, Y., Ying, C., Zheng, H., Wei, Z., Xilong, D., 2015. Design and Implement of a Flexible Workflow Model Based on UML Modeling Technology. *Applied Mechanics and Materials* Vols. 738-739, pp. 304-310.

Gronmo, R., Jaeger, M. C., 2005. Model-Driven Semantic Web Service Composition. In. Proc. APSEC'05, Dec, pp. 79-86.

Group, O.M., 2005. Uml 2.0 superstructure specification. Technical report.

Hu, J. M., Zhang, S. S., Yu, X.Y., 2002. A Workflow Model Based on ECA Rules and Activity Decomposition. *Journal of Software*, 13(4), pp 761-767.

Nurcan, S., 2008. A Survey on the Flexibility Requirements Related to Business Processes and Modeling Artifacts. In *Proceedings of the 41st Hawaii International Conference on System Sciences*.

Regev, G., Wegmann, A., 2005. A Regulation-Based View on Business Process and Supporting System Flexibility, *Proc. of the CAiSE'05 Workshop*, pp. 91- 98.

Rosemann, M., Recker, J., 2006. Context-aware Process Design Exploring the Extrinsic Drivers for Process Flexibility, BPMDS, Luxembourg.

Saidani, O., Nurcan, S., 2006. A Role-Based Approach for Modelling Flexible Business Processes, *The 7th Workshop on Business Process Modelling, Development, and Support* (BPMDS'06), (in association with CAISE'06), Springer Verlag (pub), Luxembourg, 2006.

Schmidt, R., 2005. Flexible Support of InterOrganizational Business Processes Using Web Services. *Proceedings of the CAiSE'05 Workshop*, pp. 51-58.

Shen, J., Grossmann, G., Yang, Y., Stumptner, M., Schrefl, M., Reiter, T., 2007. Analysis of Business Process Integration in Web Service Context. *Future Generation Computing System*, vol. 23, no. 3, pp.283–294.

Shi, D., Danies, R.L., 2003. A survey of Manufacturing Flexibility: Implications For E-Business Flexibility. *IBM Systems Journal* 42(3), p.414-427.

Van der Aalst, W. M. P., 2001. How to handle dynamic change and capture management information? An approach based on generic workflow models. In *Proceedings of the Fourth IECIS International Conference on Cooperative Information Systems*. Woodruff, J. P., Van Arsdall, P. J., 1998. A Large Distributed Control System Using Ada in Fusion Research. In *Proceedings of the 1998 annual* ACM SIGAda international conference on Ada, pp 121-131.

Yubin, G., Zeye, C., Zewei, L., Ximing, L., 2013. Design and Implementation of a Flexible Workflow Management System. *Journal of Software*, Vol 8, No 12, pp. 3060-3065.