DYNAMIC ADAPTATION OF SERVICE TO THE CONTEXT WITH VARIABILITY MODELS

Naoufel Kraiem[1,2], Omnia Saidani Neffati[2], Rim Samia Kaabi[2] and Zuhoor Al Khanjari[1]

[1]CSComputer Science Department University of Sultan Qaboos, Muscat, Oman.

[2]RIADI Lab, National School of Computer Science, University of Manouba, Manouba, Tunisia

Contact: naoufel@squ.edu.om, omnisaidani@gmail.com, rim.kabi@riadi.rnu.tn, zuhoor@squ.edu.om

***ABSTRACT*:** *Web services run in complex contexts where arising events may compromise the quality of the whole system. The dynamic nature of environments in which Web services are executed require that these ones must be reactive and adaptive in order to meet changing requirements. These changing requirements have to be taken into account in the Web service composition process. BPEL (Business Process Execution Language), the de-facto standard language used to define processes through the composition of Web services, doesn't support dynamic changes. Therefore, BPEL doesn't support the dynamic adaptation of Web service composition according to the context. In this paper, we propose solution in order to support the dynamic adaptation of service composition through BPEL. An evaluation demonstrates several benefits of our approach, both at design time and at runtime.*

Keywords: Web Service Composition and Orchestration, BPEL, BPMN, Context-awareness, Dynamic Adaption.

1. INTRODUCTION

Web Services are self-contained, modular, distributed and dynamic applications that can be described, published, located, or invoked over the network in order to create products, processes, and supply chains. These applications can be local, distributed, or Web based [1].

Web services can be combined in order to achieve specific functionalities. The process of assembling of these services together is called service composition [2].

The service composition can be defined as the process of constructing a complex service from atomic ones to achieve a specific task [3].

Web services are executed in dynamic environments in which several exceptional situations may arise continuously. We find out as cited in [4], that the dynamic nature of these environments requires that Web services must be reactive and adaptive in order to meet changing requirements. Therefore, these changing requirements have to be taken into account in the Web service composition process. Furthermore, a support for the Web service adaptation according to the context is required.

Among the most popular composition languages for Web services we note BPEL (Business Process Execution Language) [5] . BPEL is the de-facto standard language used to define processes through the composition of Web services [6]. Nevertheless, BPEL neither supports dynamic changes [7], nor does it offer a support for the dynamic adaptation of the orchestration logic according to the context [8], [9]. This fact is triggered by the static nature of BPEL process definitions which makes it difficult to adapt Web services at runtime [6].

In [10], the authors confirms the inability of BPEL engine to monitor running processes and thereafter to decide which Web service has to be replaced or omitted from a given composition in certain circumstances (e.g. when a service execution failure happens). BPEL needs to be redeployed in order to be adapted. The redeployment of the BPEL process implies the downtime of all the system since all services should be stopped until the modification process is done [11].

In this paper, we propose a methodological approach allowing to support the dynamic adaptation of service compositions according to the context based on BPEL.

The remainder of this paper is organized as follows: Section 2 introduces the background Section 3 presents an illustrative example. Section 4 presents the proposed approach. Section 5 provides architecture for a support tool. Section 6 introduces a case study in order to validate our approach. And finally, section 7 concludes the paper.

2. BACKGROUND

This section will focus on the definition of the basic concepts that are related to our approach.

2.1. Context and Context-awareness

The concepts of context and context-awareness are defined and used in several approaches [12,13.14].

According to [12], the context is all information that can be used to characterize an entity. An entity can be a person, a place, or a relevant object for the interaction between a user and an application, including the user and the application itself. In [15], Chihani has considered that the context-awareness is a particular type of formal logic on which theories and artificial intelligence algorithms (e.g. inference rules) can be applied in order to automate the deduction of new contextual knowledge and reasoning about the facts that represent the situation of the user. Context-aware service oriented systems refer to applications that use context information to provide appropriate services to the user [16].

2.2. Dynamic Adaptation

According to [17], the adaptation consists of making changes to a software or a computer system in order to perform its features and, if possible, to improve its performance in an environment of use.

Software adaptation can be seen as the ability for humans to reconfigure the software and then to restart it (static adaptation), or the ability of the software to reconfigure itself during execution (dynamic adaptation) [18]. According to [19], the dynamic adaptation of software behavior can be defined as the act of changing the behavior of some part of a software system as it executes, without stopping or restarting it.

3. Illustrative Example

In order to illustrate our work, we use the example of a hotel booking process Figure (1). The business rules are described as follows: the customers can make booking according to their preferences (e.g. Arrival and departure dates, room

type, service type). They can check the availability of rooms according to their arrival and departure dates. The booking is made and the payment is done on the basis of room's availability.

We distinct four different payment methods: credit card payment, cash payment, bank transfer payment and check payment.

- *Check the room availability:* it aims at checking the room availability overlooked the arrival and the departure dates set by the customer.
- *Make booking:* it corresponds to the assignment of a room to the customer according to his preferences. This task is a composite task (sub-process) that is composed of the following tasks:
- *Provide the personal information:* It consists at giving the information related of the customer i.e. name, address, etc.).
- *Specify the arrival and the departure dates.*
- *Specify the room type.* The room type can be single, double, etc.
- *Specify the service type.* The customer has to precise the service type: full board, half board, etc.
- *Pay booking:* it corresponds to the payment of the booking fees. The payment can be done by credit card, by bank transfer, by check or cash.

We have chosen to model a given BPEL process through the BPMN diagram (Business Process Modeling Notation) [20], for the following reasons: BPMN tasks can express Web service operations while BPMN sub-processes can express composite service operations. Moreover, several approaches [21,22], have adopted BPMN model to represent the elements in a service composition since BPMN is suitable to express sequences and dependencies among Web services, on the one hand, and BPMN diagram can be transformed to an executable BPEL process through several dedicated tools, on the other hand. In this example, we will focus on the payment service. We assume that the customer has chosen to pay the reservation by the credit card. At runtime, a number of unexpected contextual changes can occur such as the lack of money needed to cover the booking fees, the expiration of the credit card, the network failure, the service unavailability due to fluctuations in available bandwidth and throughput rates, etc. These facts make impossible the achievement of the booking payment with the given mean (i.e. by credit card).

As consequence, the payment process may not finish correctly. Nevertheless, the booking process has not to be stopped until the customer does not wish to finishes it. For this, we need to propose a solution in order to adapt the hotel booking process to the unexpected changes that can occur at untime in order to satisfy without shutting down the system.

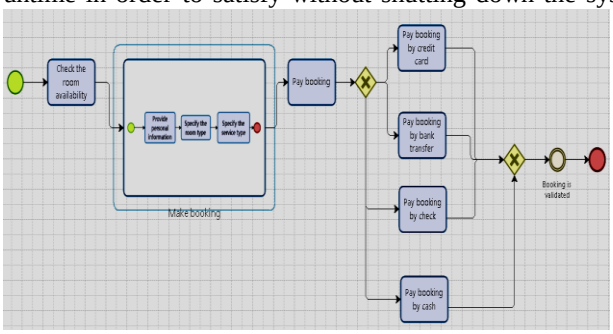

Figure 1: The hotel booking example

3. THE PROPOSED APPROACH

In order to adapt the hotel booking process to the unexpected changes that can occur at runtime, we present a methodological approach for the dynamic adaptation of service compositions according to the context. The proposed approach consists mainly of two steps: "the context management" step and "the adaptation actions" step.

The context management consists on the context modeling, capture and reasoning. The adaptation action consists of dealing with the context changes by applying a set of adaptation rules which are useful in order to adapt the BPEL process behavior according to the context changes. Details will be presented in section 4.1 and section 4.2.

4.1. The Context Management Step

This step is crucial especially in order to identify and to model the contextual information pertinent for services, to determine the current context.

4.1.1 A Context Model for Services

In order to use the contextual information in an adequate manner, we need to define a context model for the representation of the service context called: CM4S (Context Model for Services). The proposed model aims to represent the contextual factors related to the service adaptation. The different concepts for describing the contextual information and the relationships between them are described through the meta-model represented by Figure (2).

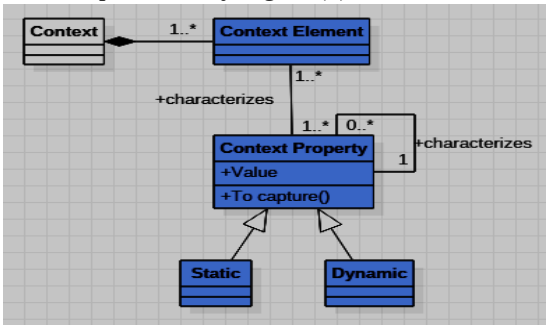

Figure 2: The context meta-model for services.

It should be noticed that only the elements with the gray color belong to the proposed context meta-model. The "Context" class is represented in order to show its relationship with the concepts of the proposed context meta-model. The concepts of the CM4S are described as follows:

- **Context element:** it represents a part of the knowledge related to the context. The service user context, the service context, the service execution environment are examples of context elements.
- **Context Property:** it is considered as an attribute that characterizes a context element. Indeed, a context element is featured by one or more property (ies) context. For example, the context element "User context" has the following context properties: profile and preferences (Figure 3). The context property can be itself featured by other context properties. For example, the "Profile" context property can be featured by the context properties: "age", "sex", "nationality", and "gender" Figure (3).

We distinguish two types of context properties; static context property which refers to a property whose value is always fixed (e.g. the context property "sex") and dynamic context property which refers to a property whose value can change(e.g. the context property (e.g. the context property "time").

4.1.2 An Ontology Based Model

Ontologies are widely accepted for modeling context information in many fields such as pervasive computing, service engineering and business process management. Ontology can be defined as a formal explicit specification of a shared conceptualization. A conceptualization is an abstract, simplified view of the world that we wish to represent for some purpose [23], [24].

We choose to use ontologies in order to instantiate the context meta-model already proposed in section 4.1.1 Figure (2).

Our choice is motivated by the following reasons: The context can be modeled as ontology since the use of ontologies allows not only the context modeling, but also the reasoning on the collected contextual data based on an inference engine [16]. Ontologies provide a formal and semantic description of context information in terms of objects, concepts, properties and relationships (Najar, S., 2014). According to [25], several advantages are involved in ontology based modeling. By providing a formal semantics to context data, it becomes possible to share and/or integrate context among different sources. Also, many available reasoning tools can be used both to check consistency of the set of relationships describing a context scenario, and to recognize higher level context information.

Thus, we propose an ontology-based model Figure (3) in order to show the contextual information. Furthermore, in order to provide an extensible, well-structured and easily understood ontology, we choose to use a multi-level ontology. In fact, the context is modeled according to two levels: (i) the general level (upper ontology) and (ii) the specific level (lower ontology).The upper ontology describes the basic context information which is quite generic and common to several areas while the lower ontology includes specific context information which is related to a particular domain.

According to the illustrative example, the presented lower ontology is specific to the booking area. We distinguish three categories for contextual elements:

- **The user context:** it represents the contextual information related to the service user. Examples of the user context are: profile, preferences, etc.
- **The Web service context**: it represents the contextual information related to the service itself such as the availability. We identify two general types of Web service contextual information: the first one is related to the business side of the service such as its goal (service of electronic payment for example) and its business rules. We qualify this type as "functional requirements". The second one is related to the "non-functional" considerations of Web services such as its availability, used security protocol, etc.
- **The environment context:** it represents the information that characterizes the service runtime environment such as the spatial and the temporal factors.

4.1.3 The Context Capture and Reasoning

The capture of the contextual information can be made through several methods such as access to the Information System (IS), access to the system calendar, and input by the user through an interface, etc.

A sensor can be defined as a hardware or software source which can generate contextual information [26]. We distinguish two main types of sensors, the physical sensors that represent hardware devices which are able to provide context data (e.g. GPS).and the virtual sensors that provide contextual information from software applications or services. Context reasoning is used in order to deduce new contextual information from existing one.

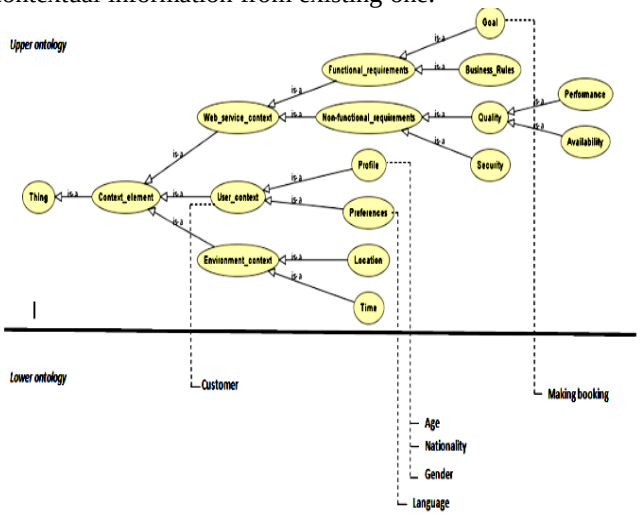

Figure 3: The context based-ontology model

This section introduces a methodology in order to express where and how a BPEL process can be adapted at runtime, according to the context. This methodology is represented using a BPMN diagram and shown in Figure (4). The methodology, that we propose, addresses the BPEL process dynamic adaptation through two axes. The first one is interested by the adaptation of the BPEL process before the detection of an execution failure. We qualify this type of

adaptation as "preventive adaptation". It aims to minimize the chances of the execution failure emergence.

The second axe aims to adapt the BPEL process after the detection of an execution failure and determines a solution through the change of the current BPEL process behavior. We qualify this type of adaptation as "curative adaptation

4.2. 1. Preventive Adaptation

This mechanism is applied on each "switch" activity in a given BPEL process (Exclusive gateway in BPMN). The "switch" activity evaluates the state of the business process and, based on the condition, breaks the flow into one of the two or more mutually exclusive paths. We qualified the "switch" activity as a "decision point". A BPEL process can contain one or more decision point. According to the booking process presented in section 3, the decision point corresponds to the exclusive gateway related to the payment method choice. This gateway allows the customer to pay the booking fees by credit card, check, bank transfer or cash. The proposed alternatives as shown in the BPMN diagram Figure (3) are: "pay booking by credit card", "pay booking by check", "pay booking by bank transfer" and "pay booking cash". We assume that the offered Web services that participate in this BPEL process and which can accomplish these alternatives are respectively: "PayCard", "PayCheck", "PayBankTransfer" and "PayCash".

Depending on the current context zero, one or more alternatives may be suitable to be offered to the user who will choose, thereafter, only one alternative to make the booking payment. Hence, it is not necessary that all the Web services that are associated with these alternatives are adequate to the current context.

A context-based selection should be performed in order to provide to the user only the services that can be executed depending on such context. This fact allows to minimize the chance of the BPEL execution failure caused by the incoherence of some Web services with the current context, on the one hand, and reduces the time allocated to the execution of the BPEL process due by the providing of unusable Web services, on the other hand.

For example, in a given context in which the user credit card is expired, it is impossible to execute the service "PayCard". This Web service should not be, therefore, offered to the user from the beginning. Only the Web services "PayBankTransferCard", "PayCheck" and "PayCash can be offered to the user. Indeed, being informed about the expiration date of the credit card before the service running allows to know at an early stage if the service can be executed or not. Hence, this fact prevents the execution failures which can arise thereafter. Besides, it is beneficial in terms of time to avoid the proposal of unusable Web service in a given context.

In order to select the appropriate alternatives that can be offered to the user according to the context, we propose the following method which includes the following five steps:

(i) Identify each switch activity in BPEL (XOR gateway in BPMN) as a "decision point". This activity evaluates the state of the business process and based on the condition breaks the flow into one of the two or more mutually exclusive paths.

(i) Identify, for each decision point, the relevant contextual elements.

(ii) Capture the values of the contextual elements, already identified, for the current context.

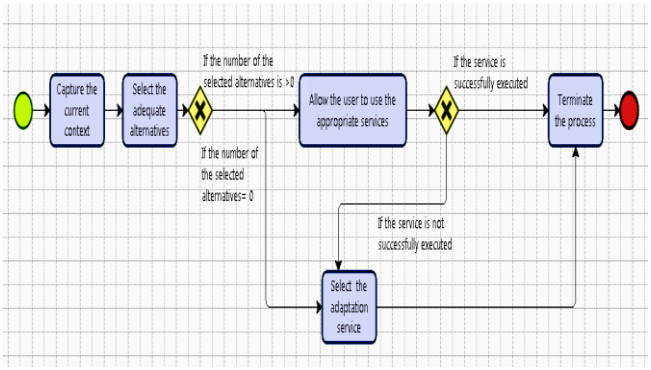

Figure 4: The adaptation methodology

Determine, for each proposed alternative, if its execution is possible with reference to the values of the current contextual elements or not.

(iii) Keep only the adequate alternatives and ignore the inadequate ones from the process schema.

Zero, one or several alternatives can be selected to be kept in the process schema. The aim of this adaptation is to adapt the BPEL process to the current context in order to prevent failure execution cases caused by the context negligence.

4.2.2. Curative Adaptation

The curative adaptation is useful when no alternative is adequate to the current context after the application of the "preventive adaptation". This fact implies that the current BPEL process will not be finished correctly since a Web service that participates to this process cannot be executed.

This category of adaptation proposes to use additional Web services (i.e. new functionalities offered by Web services) which are not included in the initial BPEL process. The use of the additional Web services leads to the modification of the initial process schema by the emergence of other alternatives initially absent. We qualify the additional Web services as "adaptation services". Their usefulness is to ensure the dynamic adaptation of the BPEL process at runtime, according to the current context. The adaptation services are Web services which are published by the service provider and stored at the UDDI repository [27]. They will be invoked in the situation of no proposed alternative within the initial process is suitable to the current context. The definition of the adaptation services is made at each decision point when one or more adaptation services can be associated with the decision point. In the case when only one adaptation service is associated with the decision point, this service adaptation will be directly invoked to be inserted into the current BPEL process. Else, in the case where several adaptation services are associated with the decision point, an association <context, adaptation service> (i.e. for each given context there is a corresponding adaptation service) will be created by the service provider, and the selection of the best adequate adaptation service is therefore based on this association. The execution of the already invoked service leads to a particular state in the process [20], e.g. the execution of the Web service "confirm registration" in the university registration process leads to the generation of the state "registration is confirmed". If the state generated after

the execution of the invoked service is the same generated at the end of the process, then the process will take end. Else if the state generated after the execution of the invoked service is the different to the generated one at the end of the process, then the process have to return again to the already defined decision point.

5. ARCHITECTURE OF SUPPORT TOOL

In order to provide a potential implementation of the proposed approach, we propose the architecture shown in Figure (5). This architecture is composed of three modules:

- The Context Management Module
- The Adaptation Management Module
- The Adaptation Module

The role of each identified module and the interaction between them is detailed as follow:

1) Context Management Module: it is mainly responsible for the capture, the modeling, the interpretation and the storage of the contextual information (user context, service context, etc.).

This module consists of the following sub-modules:

(i)The Context Capture sub-module: it is responsible for the capture of the contextual information values and its interpretation.

(ii)The Context Storage sub-module: it is responsible for the storage of the contextual information. It consists of a context register.

2) The Adaptation Service Management Module: it is responsible for the storage and the management of the adaptation services which are invoked when required in order to adapt the running BPEL process.

3) The Adaptation Management Module: it is responsible for the adaptation process of the BPEL process by applying the adaptation rules. It is related to both the Context Management Module in order to extract the contextual information already stored and the Adaptation Service Management Module in order to select and to invoke the adaptation service.

This module consists of the following sub-modules:

(i)The Extraction Contextual Information sub module: it extracts the contextual information already stored in the context register.

(ii)The Adequate Service Selection sub-module: it is responsible for the selection of the adequate services to be allowed to the user according to the context.

(iii)The Adaptation Service Selection and Invocation sub-module: it is responsible for the selection and the invocation of the adaptation service.

The interaction between the different modules of the proposed architecture is as follows: The Context Capture sub-module captures changes in contextual information related to the Web service, to the user and to the environment (1). The captured contextual information will be stored, thereafter, in the context register (2). The Extraction Contextual Information sub-module extracts the contextual information already stored in the context register (3). The Adequate Service Selection sub-module selects the adequate services to be allowed to the user according to the context (4). If it is necessary (any service can be selected), the Adaptation Service Selection and Invocation sub-module selects and invokes the suitable adaptation service (5).

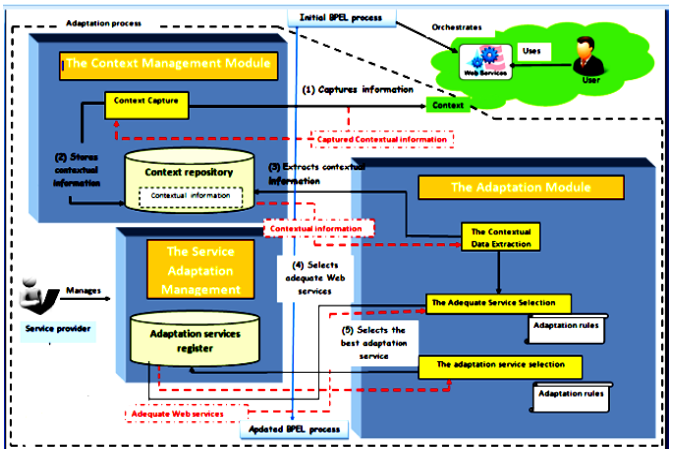

Figure 5: The support tool architecture

6. CASE STUDY

We propose to implement our approach on the hotel booking process (Section 3).

6.1. The Proposed Scenario

We take the following scenario: the customer consults the hotel website in which he wants to spend the holiday. He checks the room availability overlooked the arrival and the departure dates. Thereafter, he provides his personal information, confirms the arrival and the departure date and chooses a single room for his accommodation. Thereafter, he chooses a half board accommodation. Finally, the customer is redirected to the booking fees payment space in order to pay the booking fees. According to the initial process, the customer can pay by card, by bank transfer, by check or cash. By using our adaptation approach, the payment method choice is a decision point in which only the adequate Web services should be offered to the user.

6.2.2. Relevant Contextual Element Definition

In order to identify the adequate Web services on the basis of the proposed method explained in section 4.2.1, the exclusive gateway concerning the payment method choice Figure (1) is considered as a decision point. Table 1 presents the defined relevant contextual elements associated to the "payment method choice" decision point.

Table 1: Relevant contextual elements related to the "payment method choice".

Decision point	Relevant contextual elements
Payment method choice	Credit card possession
	Checkbook possession
	Cash possession
	Bank account possession
	Credit card status
	Preference to pay by credit card
	Preference to pay by check
	Preference to pay by cash
	Preference to pay by bank transfer
	Client type
	Current day
	Pay by credit card service availability
	Pay by check service availability
	Pay by cash service availability
	Pay by bank transfer
	Service availability

The description of the relevant contextual elements defined in Table 1 is as follow:

- **Credit card possession:** it indicates if the client has a credit card or not.

- **Checkbook possession:** it indicates if the client has a checkbook or not.
- **Cash possession:** it indicates if the client has cash or not.
- **Bank account possession:** it indicates if the customer has a bank account or not.
- **Credit card status:** it indicates if the credit card is valid or expired.
- **Preference to pay by credit card:** that indicates if the customer prefers to pay the booking fees by his credit card or by another mean.
- **Preference to pay by check:** it indicates if the customer prefers to pay the booking fees by check or by another mean.
- **Preference to pay by bank transfer:** it indicates if the customer prefers to pay the booking fees by bank transfer or by another mean.
- **Preference to pay by cash:** it indicates if the customer prefers to pay the booking fees by cash or by another mean.
- **Customer type:** it indicates whether the customer is accustomed to make bookings at this hotel or not. If the number of bookings per year is more than five, then the client is described as "regular" otherwise he is described as "normal".
- **Current day:** it indicates the current date.
- **Pay by credit card service availability:** it indicates if the payment by credit card service is available or not.
- **Pay by check service availability:** it indicates if the payment by check service is available or not.
- **Pay by cash service availability:** it indicates if the payment by cash service is available or not.
- **Pay by bank transfer service availability:** it indicates if the payment by bank transfer is available or not.

The selection of the adequate alternative requires a given context. A context representation can be defined as a conjunction (AND) of contextual assertions which may be negative by a negation operator (NOT) (Angles, R., 2014). The required contexts related to the proposed alternatives are described as follow:

- **Pay booking by credit card=** ((Credit card possession= yes) AND (Credit card status= valid)AND (Preference to pay by credit card=yes) AND (Payment by credit card service availability=yes)

This assertion implies that in order to use the "PayCard" Web service, the customer should have a valid credit card and the use of the card has not to be out of his preferences. Besides, the "PayCard" Web service should be available at the current time.

- **Pay booking by check=** (Check book possession=yes) AND (Preference to pay by check=yes) AND (Payment by check service availability=yes)

This assertion implies that in order to use the "PayCeck" Web service, the customer should have a check book, and the use of the check book has not to be out of his preferences. Finally, the "PayCeck" Web service should be available at the current time.

- **Pay booking by bank transfer=** (Bank account possession=yes) AND (Preference to pay by bank transfer=yes) AND (Payment by bank transfer service availability=yes) AND NOT ((day = Saturday) OR (day= Sunday))

This assertion implies that in order to use the "PayBankTransfer" Web service, the customer should have a bank account, and the use of the bank transfer has not to be out of his preferences. Besides, the "PayBankTransfer" Web service should be available at the current time. Also, it should not be a Saturday or Sunday day since most banks are closed these days.

- **Pay booking by cash=(**Cash possession=yes) AND (Preference to pay by cash=yes) AND (Payment by cash service availability=yes)

This assertion implies that in order to use the "PayCash" Web service, the customer should have cash, and the use of the bank transfer has not to be out of his preferences. Finally, the "PayCash" Web service should be available at the current time.

6.3. The Relevant Contextual Element Capture

We assume that, in the current context, the capture of the relevant contextual elements values shows the following elements (see Table 2). These values can be captured explicitly or implicitly. We give just a preview.

Table 2: Relevant contextual elements values.

Relevant contextual element	Value	Capture method
Credit card possession	Yes	Explicit (provided by the user)
Credit card status	Valid	Implicit (by the access to the calendar)
...	...	...

6.4. Adequate Web Services Definition

We assume that the current context allows using only the credit card in order to pay the booking fees. Hence, only the "PayCard" Web service will be offered to the user in this context. This "preventive adaptation" modifies the initial booking process Figure (6).

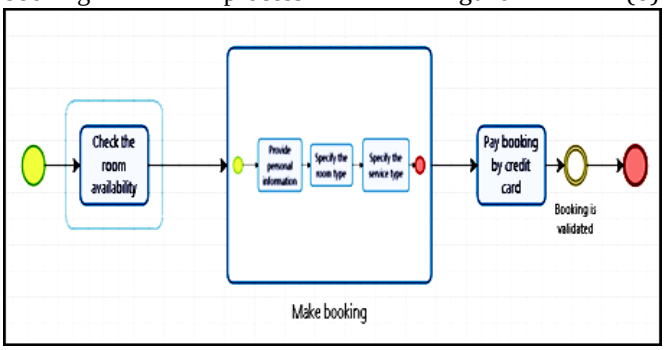

Figure 6: The new process schema after the "preventive adaptation"

6.5. Adequate Web Services Definition

As mentioned in section 4.2, each decision point is associated with one or more adaptation services.

We assume that the service provider associates the following adaptation services to the "payment method choice" decision point:

- "KeepingReservation2Days" adaptation service: allows keeping the booking process for two additional days.
- "KeepingReservation4Days" adaptation service: allows keeping the booking process for additional four days. It is used especially when it is a weekend period for example

and keeping the reservation process only for two days is not enough.

- "ValidateReservation" adaptation service: allows validating the booking process as if the customer paid the booking fees. This adaptation service can be used in rare cases where a customer is considered "faithful customer" (i.e. who's the number of booking made during the year exceeds five times [28]. This service does not replace the booking payment service but rather to validate the booking in order to satisfy a "faithful customer" and terminates the process as well. The specification of the payment procedure in this case is left to the enterprise business expert. For example, when a new booking process is launched by the "regular customer", the fees payment of this booking is made [27].

The already defined adaptation services and their required contexts (defined by the business expert of the enterprise) are presented in Figure (7).

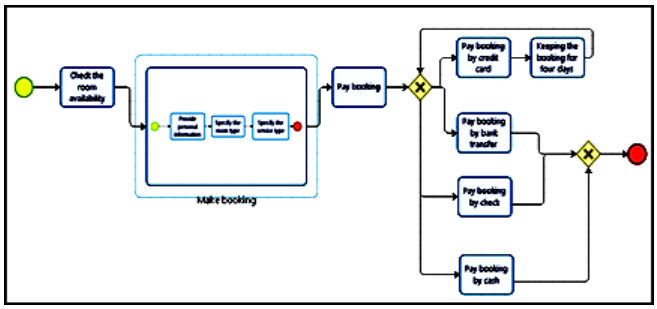

Figure 7: The adapted process after the "curative adaptation.

	Context 1		Context 2		Context 3		Context 4	
Alternative	(Current date= Saturday) OR (Current date= Sunday)	Customer type=regular	NOT ((Current date= Saturday) OR (Current date= Sunday))	Customer type=regular	(Current date= Saturday) OR (Current date= Sunday)	Customer type=faithful	NOT ((Current date= Saturday) OR (Current date= Sunday))	Customer type=faithful
KeepingReservation2Days	No		Yes		No		No	
KeepingReservation4Days	Yes		No		No		No	
ValidateReservation	No		No		Yes		Yes	

Figure 8: The adaptation services and their required contexts.

Currently, we do not have need to use an adaptation service from the three proposed ones since the definition of adequate alternatives step implies that the "PayCard" service is adequate to the current context and it can be used by the customer.

6.6. Context Change and Adaptation Service Selection

When the user enters the credit card number to make the booking payment, an alert notifies him that his credit card amount is not sufficient to cover the accommodation fees. Hence, the payment cannot be made with this credit card. The alert is an event that will trigger either the proposal of another alternative (i.e. the use of another allowed Web service if it exists) or the looking for a suitable adaptation service in the case of the absence of another alternative. In our case, there are no other alternatives offered to the customer since the only possible way is "Pay booking by credit card" (see section 6.4). Then, the call for an adaptation service remains the only possibility in order to adapt the BPEL process to the new context.

According to the new context, the adaptation service "KeepingReservation4Days" will be selected and invoked. The adapted process is modelling using BPMN diagram and shown in Figure (8).

7. Conclusion and Future Work

In this paper, we have presented a solution to support the dynamic adaptation of service compositions. The "preventive adaptation" is carried out before the execution of given services that participate to the process. This adaptation is based on the selection of adequate services depending to the current context. The "curative adaptation" is carried out at the service runtime and it is based on the invocation of the "adaptation services". Our solution is validated by a hotel booking case study and illustrated by support tool architecture.

Actually, we are working on the implementation of the proposed support tool architecture. Also, we will look to extend our approach in order to handle with the service composition issue in unknown contexts.

REFERENCES

[1] "No Title," *IBM*, 2015. .

[2] A. Salinas and P. Ferragud, 2014 "Document downloaded from : This paper must be cited as : Dynamic Adaptation of Service Compositions with Variability Models," *Journal of Systems and Software* .pp. 24–47,.

[3] C. Szyperski, "Component Software and the Way Ahead," *Found. Component-Based Syst.*, pp. 1–20, 2000.

[4] C. Zeginis and D. Plexousakis, "Web service adaptation: State of the art and research challenges," *Self*, pp. 1–66, 2010.

[5] OASIS Web Services Business Process Execution Language (WSBPEL), D. Jordan, and A. Alves, "Web Services Business Process Execution Language Version 2 . 0," *Language (Baltim).*, vol. 11, no. April, pp. 1–264, 2007.

[6] D. Domingos and C. Cândido, "Flexibility in Cross-organizational WS-BPEL Business Processes," *Procedia Technol.*, vol. 9, pp. 584–595, 2013.

[7] A. Charfi and M. Mezini, *AO4BPEL: An aspect-oriented extension to BPEL*, vol. 10, no. 3. 2007.

[8] K. Boukadi, C. Ghedira, Z. Maamar, D. Benslimane, and L. Vincent, "Context-aware data and IT services collaboration in E-business," *Lect. Notes Comput. Sci. (including Subser. Lect. Notes Artif. Intell. Lect. Notes Bioinformatics)*, vol. 5740 LNCS, pp. 91–115, 2009.

[9] "World â€™ s largest Science , Technology & Medicine Open Access book publisher A Study of the Porosity of Activated Carbons Using the Scanning Electron Microscope," 2010.

[10] K. Boumhamdi and Z. Jarir, "A Flexible Approach to Compose Web Services in Dynamic Environment," *Int.*

J., vol. 1, no. 2, pp. 157–163, 2010.

[11] H. Tout, A. Mourad, C. Talhi, and H. Otrok, "AOMD approach for context-adaptable and conflict-free Web services composition," *Comput. Electr. Eng.*, vol. 44, pp. 200–217, 2015.

[12] F. Computing and E. Group, "Understanding and Using Context."

[13] S. N. Han, G. M. Lee, and N. Crespi, "Semantic context-aware service composition for building automation system," *IEEE Trans. Ind. Informatics*, vol. 10, no. 1, pp. 252–261, 2014.

[14] J. D. Kim, J. Son, and D. K. Baik, "CA 5W1H Onto: Ontological context-aware model based on 5W1H," *Int. J. Distrib. Sens. Networks*, vol. 2012, 2012.

[15] B. Chihani, E. Bertin, F. Jeanne, N. Crespi, B. Chihani, E. Bertin, F. Jeanne, and N. C. Context-aware, "Context-aware systems : a case study To cite this version :," 2012.

[16] H. Guermah, T. Fissaa, H. Hafiddi, M. Nassar, and A. Kriouile, "A semantic approach for service adaptation in context-aware environment," *Procedia Comput. Sci.*, vol. 34, pp. 587–592, 2014.

[17] A. Celentano, "Adaptation in context-aware pervasive information systems : the SECAS project," 2006.

[18] F. Akkawi, A. Bader, D. Fletcher, K. Akkawi, M. Ayyash, and K. Alzoubi, "Software adaptation: A conscious design for oblivious programmers," *IEEE Aerosp. Conf. Proc.*, 2007.

[19] J. Keeney, "Completely Unanticipated Dynamic Adaptation of Software Completely Unanticipated Dynamic Adaptation of Software," no. January 2005, 2014.

[20] "Documents Associated With Business Process Model And Notation™ (BPMN™) Version 2.0," *BPMN*, 2011. [Online]. Available: http://www.omg.org/spec/BPMN/2.0/. [Accessed: 01-Dec-2015].

[21] C. Ayora, V. Torres, V. Pelechano, and G. H. Alférez, "Applying CVL to business process variability management," *Proc. Var. You Work. Var. Model. Made Useful Everyone - VARY '12*, no. February 2015, pp. 26–31, 2012.

[22] R. Angles, P. Ramadour, C. Cauvet, and S. Rodier, "Adaptation dynamique de processus m??tier: Application au circuit du medicament ?? l???AP-HM," *Ing. des Syst. d'Information*, vol. 19, no. 2, pp. 35–60, 2014.

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

[24] R. Studer, V. R. Benjamins, and D. Fensel, "Knowledge Engineering: Principles and Methods," *Artif. Intell.*, 1998.

[25] C. Bettini, O. Brdiczka, K. Henricksen, J. Indulska, D. Nicklas, A. Ranganathan, and D. Riboni, "A survey of context modelling and reasoning techniques," *Pervasive Mob. Comput.*, vol. 6, no. 2, pp. 161–180, 2010.

[26] J. Indulska and P. Sutton, "Location management in pervasive systems," *Conf. Res. Pract. Inf. Technol. Ser. Vol. 34*, p. 143, 2003.

[27] J. Alston and D. Hess, "UDDI (Universal Description , Discovery , and Integration) and Web Services : Perspective Summary registry of Web services data and metadata , APIs , and interfaces for accessing the services . Note Universal Description , Discovery and Integration (UDDI) is a registry , not a repository . The UDDI registry only stores the interfaces to Web services , but not the Web services themselves ; the implementations are available at another location . A repository , on the other hand , would store both the interfaces and the UDDI (Universal Description , Discovery , and Integration) and Web Services : Perspective List Of Figures," no. November, pp. 1–28, 2002.

[28] Y. Jamoussi, "Enhancing Satisfaction of Actor's Requirements in Web Service Composition: A Guided Negotiation Based Approach," *J. Softw. Eng.*, vol. 9, pp. 429–450, 2015.