Computers and Electrical Engineering 44 (2015) 200–217

Contents lists available at ScienceDirect

Computers and Electrical Engineering

journal homepage: www.elsevier.com/locate/compeleceng

AOMD approach for context-adaptable and conflict-free Web services composition ☆

Hanine Tout [a], Azzam Mourad [a,*], Chamseddine Talhi [b], Hadi Otrok [c]

[a] Department of Computer Science & Mathematics, Lebanese American University, Lebanon
[b] Department of Software Engineering & Information Technology, École de Technologie Supérieure, Canada
[c] Department of Computer Engineering, Khalifa University of Science, Technology & Research, United Arab Emirates

ARTICLE INFO

Article history:
Available online 22 April 2015

Keywords:
Web services composition
BPEL
Aspect-Oriented Programming
Aspect-oriented modelling
Adaptability
Formal verification

ABSTRACT

BPEL or Business Process Execution Language is so far the most important standard language for effective composition of Web services. However, like most available process orchestration engines, BPEL does not provide automated support for reacting according to many changes that are likely to arise in any Web services composition, like downtime services, modifications in the business logic or even new policies to govern the composition. Also low-level specification of these new changes, which would be integrated at runtime in the BPEL process, will be far from being used conveniently. Moreover, the complexity of interaction in composite Web services and the diversity of rules and policies can lead to critical behavioral conflicts. We propose in this paper AOMD, a novel aspect-oriented and model driven approach that defines new grammar to address both adaptability and behavioral conflicts problems, and offers extension for WS-BPEL meta-model for high level specification of aspects. Further, we formally verify our proposition and we present real life case study, examples and experimental results that demonstrate the feasibility and effectiveness of our work.

1. Introduction

Web services composition is emerging as a universal technology for integrating distributed and heterogeneous services over the internet in order to consolidate business applications across organizational boundaries [1]. BPEL or Business Process Execution Language [2] is so far the most important standard language for effective composition of Web services. Nonetheless, today's business environments are challenged by the need for continuous adaptation of business processes in order to meet with many changes that are likely to arise in any Web services composition after deployment, such as downtime services, modifications in the business logic or even new policies to govern the composition. Existing Web service-based process composition approaches such as BPEL need to be redeployed in order to be adapted. Process re-deployment generates downtime for systems since all services should stop until the modification process is done and even lead to possible loss of information about on-going transactions. The only changes possible at runtime are the bindings to partner links, yet not only they must be predefined at deployment-time but also cannot accommodate with all these types of changes. This problem has triggered active research efforts and achievements [3–8] that leverage Aspect-Oriented Programming (AOP) to address this limitation. In AOP, new behaviors, rules and security policies can be defined

☆ Reviews processed and recommended for publication to the Editor-in-Chief by Associate Editor Dr. H. Vahdat-Nejad.

* Corresponding author.

http://dx.doi.org/10.1016/j.compeleceng.2015.04.004
0045-7906/

independently as aspects [9], then weaved based on one of the weaving technologies such as AspectJ [10], and dynamically activated in the Web services composition at runtime.

Nevertheless, these approaches suffer from several limitations. First, [3,4,6,7] do not provide effective constructs to define context-aware aspects that can make the process adaptable with the state changes of predefined variables in the composition. Following these approaches, developers need to access the BPEL code and pass over the list of predefined messages to be aware of the existing variables, in order to formulate conditions of context-aware aspects. Such requirement entails some restrictions on the main purpose of using AOP, which is the separation of cross-cutting concerns, and limits the usefulness of that paradigm. Second, due to the complexity of interaction between composed Web services and diversity of rules and policies, conflicts are more likely to arise, a problem that [3,4,6,7] did not address. For instance, assuming that "Encryption" and "Logging" aspects are to be added after invoking particular Web service in the BPEL process; as defined in [11], the conflict problem arises in case the logging aspect is executed before the encryption aspect, where critical data will then be logged without being encrypted, leading to security flaw and rule violation in the system. Third, the specification of aspects in [3,4,6–8] is done at lower level making it way far to be used conveniently and friendly. Finally, other than performance analysis, none of these approaches has verified their proposition to make sure that all the specified aspects have been weaved correctly in the process as intended, the process is still operating correctly and remains flaw, conflict and deadlock free after the weaving.

We propose in this paper AOMD, a novel aspect-oriented and model driven approach that defines new grammar to address both adaptability and behavioral conflicts problems, and offers extension for WS-BPEL meta-model for high level specification of aspects. Further, we formally verify our proposition after weaving the aspects in the BPEL process. The verification can be done also at previous stage i.e., at the proposed model level, however this is out of the scope of this paper and we leave it for future work. We also present real life case study, examples and experimental results. All these along with the verification demonstrate the feasibility and effectiveness of our work.

This work offers the following contributions:

1. New Aspect-Oriented grammar to reach highly context-adaptable and conflict-free Web services composition: we first extend the Aspect-Oriented language, that we proposed previously [3], with new grammar in order to reach high adaptability by offering the ability to define context-aware aspects that allows the process to react dynamically with the state changes of predefined variables in the composition without the need to manually explore the original process code. Not only that, the new grammar is also able to solve the behavioral conflict that arises between different aspects applied at the same point in the business process of the Web services composition. The new grammar defines an order between them to ensure conflict-free composition (Section 3.1).
2. New aspect-oriented model to allow high level specification of aspects (i.e., behaviors, business rules and security policies) to be hardened in the BPEL process of the Web services composition: The model has not been presented previously. In our previous work [3], the specification of the aspects, that will be weaved in the BPEL process of Web services composition had to be done at lower level, making it way far to be used conveniently and friendly (Section 3.2).
3. Formal verification of the proposed approach: To the best of our knowledge, none of the relevant approaches that we present in the literature review leveraging AOP to BPEL has verified that all the specified aspects have been weaved correctly in the process as intended, the process is still operating correctly and remains flaw, conflict and deadlock free after the weaving (Sections 5 and 6.3).
4. Prototype of the proposed approach: We integrated it in a well-known and widely used commercial BPEL development environment, Eclipse. Please note that the prototype of the framework has passed through a complete process of testing and reviews by Eclipse BPEL project leader and members before getting accepted as part of their commercially used tool. To download and get additional information about the developed framework, please visit the following two links: http://www.azzammourad.org/#projects and http://www.eclipse.org/bpel/team.php. You can refer to Section 4 for more details (Section 4).

The rest of the paper is organized as follows. Section 2 is dedicated to discuss some related works. Section 3 is devoted for the proposed approach. Section 4 illustrates the framework design and implementation. Section 5 describes the verification objectives as well as the verification mechanism. Section 6 presents a full case study including experimental results. Finally, Section 7 concludes the paper and draws some future research directions.

2. Related work

In what follows, we discuss the existing approaches that leverage AOP to provide secure and adaptable Web services composition, and we expose their limitations.

AspectBPEL [3] is introduced as an Aspect-Oriented Programming language that is built on top of the current AOP techniques. This language is adapted to BPEL in order to allow the specification of BPEL security aspects. The approach proposed Tout et al. [4] is based on a synergy between XrML security license, Aspect-Oriented Programming (AOP), and BPEL. It offers the ability to define grants within licenses, associate them with the offered activities, and transform them automatically into

AspectBPEL aspects to include and update non-functional requirements such as license grants validation into a BPEL process, at runtime, and without affecting its business logic.

Hamdi et al. extended WSPL to allow context-based specification of policy rules [6]. They build a synergy between context and AOP concepts in order to integrate context-aware WSPL policies in BPEL processes of composite Web services. They provide an aspect-oriented policy tool, which allows service providers to integrate context-aware policies into the BPEL code. Yet these approaches [3,4,6] did not address the behavioral conflicts between the aspects, a critical problem that we were able to prevent in our work. Also, the specification of aspects is done at lower level, which makes them way far to be used conveniently and friendly. Finally, none of these works have verified their approaches.

AO4BPEL [7], is an aspect oriented extension for BPEL, which is able to offer modularity and adaptability to workflow processes. Yet, it has several limitations. First, it requires the use of a special orchestration engine to manage the resulting BPEL process, which cannot be executed on standard BPEL execution engines. Second, AO4BPEL does not address the behavioral conflict problem, and does not include verification of the proposed approach to prove that all the specified aspects have been weaved correctly in the process as intended, the process is still operating correctly and remains flaw, conflict and deadlock free after the weaving. As presented throughout this paper, we were able to overcome all these limitations. Finally, their approach imposes lot of overhead since it performs a check on each activity in the process to determine whether or not their aspect code is associated with it. However, our experimental results that have demonstrated that the overhead imposed by our approach is negligible.

Braem et al. [12] has proposed a pointcut language that addresses all BPEL activities and allows selecting them as join points based on their properties specified in the BPEL code, while the authors extended later the approach with advanced "Stateful aspects" in order to denote the crosscutting concerns that depend on the past states of the program. The approach offers extensibility to WS-BPEL and the ability to track the state of the process during its execution [8]. However, one of the key issue is that during the weaving, many instrumentation pieces of code that are not part of the business logic are injected into the process to evaluate the protocol, the latter should be compared at runtime with the order of all states changes, both facts generate lot of run-time overhead as they confirmed in their paper. Per contra, in our work, any verification is done selectively at specific pointcuts in the BPEL process rather than unnecessary checks at each invokes. Also, in our proposition, only the condition that activates the aspect is checked rather than tracing the whole process state that includes tracking of all the events.

Differentially from the work presented by Patiniotakis et al. [13] that is devoted for BPMN, the approach we are proposing is for BPEL. The authors extend AO4BPMN with new elements like "replace" and "bypass" relation between advices and process models. Yet, they did not make it clear if there it is possible to define parallel advices for the same join point, while we made it clear through example that it is possible to define and weave more than one aspect at the same point in the Web services composition (e.g., Discount, Encryption and Logging). Now assuming that they support parallel advice, there will be inconsistency in their proposition when an advice wants to "replace" the join point, and another one tries to "bypass" it at the same time. However, as demonstrated in the paper, our approach is able to solve this with the new construct of E-AspectBPEL. Their approach enables also the detection of situations that require process adaptation. Likewise, we are able to achieve this, one can define aspects to monitor the process and react accordingly.

3. Proposed approach

We start this section by presenting both, the proposed approach schema and a comparison between this work and our previous achievement [3] to shed the light on the new contributions. The schema in Fig. 1 depicts the main components and contributions of our proposed approach.

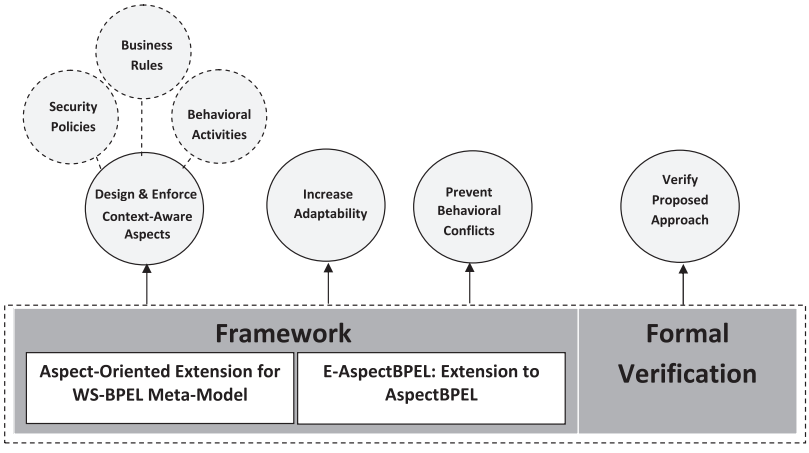


Fig. 1. Approach schema.

We first extend the Aspect-Oriented language (see Section 3.1), that we proposed previously [3], with new grammar in order to reach high adaptability by offering the ability to define context-aware aspects that allow the process to react dynamically according to the state changes of predefined variables in the composition accommodating with the changes in the business logic and even in the security policies that govern the composition. All without the need to manually explore the original process code. Not only that, the new grammar is also able to solve the behavioral conflict that arises between different aspects applied at the same point in the business process of the Web services composition. The new grammar defines an order between them to ensure conflict-free composition. Moreover, in our previous work, the specification of the aspects, that will be weaved in the BPEL process of Web services composition, had to be done at lower level making it way far to be used conveniently and friendly. However, in this work we offer Aspect-Oriented extension for WS-BPEL meta-model to allow the high level specification of aspects (see Section 3.2). Furthermore, opposed to our previous work that only included performance analysis of the composition after weaving the specified aspects; we add in this work a formal verification of the proposed approach proving that all the specified aspects have been weaved correctly in the process as intended, the process is still operating correctly and remains flaw, conflict and deadlock free after the weaving (see Sections 5 and 6.3). We keep both the implementation and the verification to separated sections.

3.1. E-AspectBPEL: AspectBPEL extension

In previous work, we have developed "AspectBPEL" a language built on top of the AOP paradigm and adapted to BPEL. AspectBPEL is an Aspect-Oriented language that allows the definition of BPEL security policies, behaviors and business rules and automatically weave them in the BPEL process. It uses notations close to those of the current AOP techniques. For more details, you may refer to [3]. In this section, we present E-AspectBPEL, our extension to AspectBPEL, that is able to reach higher adaptability and provide conflict-free composition.

Hereafter we present the syntactic constructs and their informal semantics. We added their equivalent in AspectJ between parentheses. Fig. 2 illustrates the grammar of the language (new constructs are highlighted).

3.1.1. Main constructs

The main construct of the language consists of `BPEL_Aspect`(aspect) that represents the aspect, `BPEL_Location_Behavior`(advice-pointcut) that represents the advice-pointcut within an aspect, `BPEL_Insertion_Point`(before, after, or around) to specify the point where the aspect will be injected with respect to the `BPEL_Location_Identifier`(pointcut) that identifies the exact joint point or sets of joint points in the process where the aspect will be activated and finally, `BPEL_Behavior_Code`(advice), which contains the new behavior code that will be weaved in the BPEL Process.

3.1.2. New constructs

3.1.2.1. Activation_Condition. Although one can specify conditions in the `BPEL_Behavior_Code` element using the regular condition statement in BPEL, yet it is not possible to access the BPEL variables directly from inside that element due to the fact that the BPEL process and the new aspects are created independently (separation of concerns concept of AOP). Thus the need for `Activation_Condition`, in which one can automatically specify the predefined variable without the need to manually access the original BPEL code and go over the messages definition to explore the variables. The expression of the activation condition consists of the BPEL variable and its value. For instance, let us consider the case where we want to add a new security aspect, in which a user has three chances to enter the authentication credentials (i.e. username and password), before being denied further access to the requested service of a BPEL process. The latter contains a BPEL context **Activation_Condition** variable called **FaultyTrials** that counts the failure login attempts in the process. Consequently, the aspect behavior (e.g., **BPEL_Behavior_Code**) that can block the user from accessing the service will not be activated unless **FaultyTrials** reaches 3. In addition, the **Activation_Condition** can be a set of variables logically combined using an Operator like "And" and "OR", rather than a single BPEL context variable. This construct offers the Web services composition, within BPEL, a higher level of awareness and adaptability to context changes.

3.1.2.2. BPEL_Aspect_Priority. This element is used in order to identify the order between the aspects upon their weaving in the BPEL process to avoid behavioral conflict situation. For instance, assume we have two aspects encryption and logging, and both share the same join point in the BPEL process. Let us consider that the data should not be accessible before being encrypted for security reasons. This requirement will be violated in case the logging aspect is triggered before the encryption [11]. Therefore, the order between these aspects plays a significant role. Based on the **BPEL_Aspect_Priority element**, the aspect having the lower priority will be weaved first in the Web services composition and dynamically activated last. In the case of multiple aspects that have the same priority, the order among them does not formulate a concern, therefore it is determined randomly. The idea of priority exists in aspects like AspectJ [14] and also exists in policy languages like XACML [15] where this notation is embedded in combining algorithms among the set of policies and rules to decide which one of them will be applied, which also make it reasonable to apply this priority-based mechanism between aspects.

```
BPEL_Aspect                 ::=  Aspect Aspect_Name
                                 BPEL_Aspect_Priority
                                 BPEL_Aspect_Body
BPEL_Aspect_Priority        ::=  Integer
BPEL_Aspect_Body            ::=  BeginAspect
                                 BPEL_Location_Behavior*
                                 EndAspect
BPEL_Location_Behavior      ::=  BPEL_Insertion_Point
                                 BPEL_Location_Identifier
                                 Activation_Condition*
                                 BPEL_Behavior_Code
BPEL_Insertion_Point        ::=  Before
                            |    After
                            |    Replace
BPEL_Location_Identifier    ::=  Assign <Signature>
                            |    Invoke <Signature>
                            |    Receive <Signature>
                            |    Reply <Signature>
                            |    Empty <Signature>
                            |    If <Signature>
                            |    Pick <Signature>
                            |    While <Signature>
                            |    Foreach <Signature>
                            |    RepeatUntil <Signature>
                            |    Wait <Signature>
                            |    Sequence <Signature>
                            |    Scope <Signature>
                            |    Flow <Signature>
                            |    Exit <Signature>
                            |    Throw <Signature>
                            |    Rethrow <Signature>
Activation_Condition        ::=  BPEL_Variable_Name
                                 Operator
                                 BPEL_Variable_Value
BPEL_Behavior_Code          ::=  BeginBehavior
                                 BPEL Code Statements
                                 EndBehavior
```

Fig. 2. Grammar of E-AspectBPEL.

3.2. Aspect-Oriented extension for WS-BPEL meta-model

We devise a new aspect-oriented model, as extension to the WS-BPEL meta-model, to offer high level specification of aspects. This model-driven solution alleviates the complexity of aspects specifications imposed by the current approaches [3,4,6–8], where aspects are hardcoded. In the proposed model, **BPEL_Aspect** is like a BPEL process that encompasses some activities to define certain behavior (i.e., logical behaviors, business rule or security policies). Therefore, we define it as a subclass of the BPEL **Process**. As shown in Fig. 3, the **BPEL_Aspect** inherits all the properties of a regular BPEL **Process**. However, it extends it with the new element *priority* needed to solve the conflict problem that may arise when weaving different aspects in the BPEL process.

As illustrated in Fig. 3, the **BPEL_Aspect** is also composed of one or more (1..*) *BPEL_Location_Behavior*. This latter is composed of the *BPEL_Insertion_Point*, *BPEL_Location_Identifier* and *BPEL_Behavior_Code*. The last element contains the BPEL activities that will be weaved in the BPEL process, thus it inherits its properties from the BPEL **Activity** class. The first two elements pinpoint the location in the BPEL process where the *BPEL_Behavior_Code* will be integrated. The *BPEL_Location_Identifier* element extends the **ExtensionActivity** element with the activityName and activityType elements. Both are mapped to an activity in the BPEL process to help identifying the *BPEL_Location_Identifier*. Moreover, it adds the *expression* element to formulate a context-aware model. The *expression* element is written in XPath and generated automatically based on the BPEL process variables. It defines the circumstances that should be met to dynamically activate the aspect behavior (as explained in the **Activation_Condition** element in the previous section).

4. Framework design and implementation

This section is devoted for technical details about the proposed framework implementation. Our prototype, in Fig. 4, has passed through a complete process of testing and reviews by Eclipse BPEL project leader and members before getting accepted as part of their commercially used tool, which demonstrates the feasibility and usefulness of our approach. It consists of the four modules: Modeler, Generator, Compiler and Weaver.

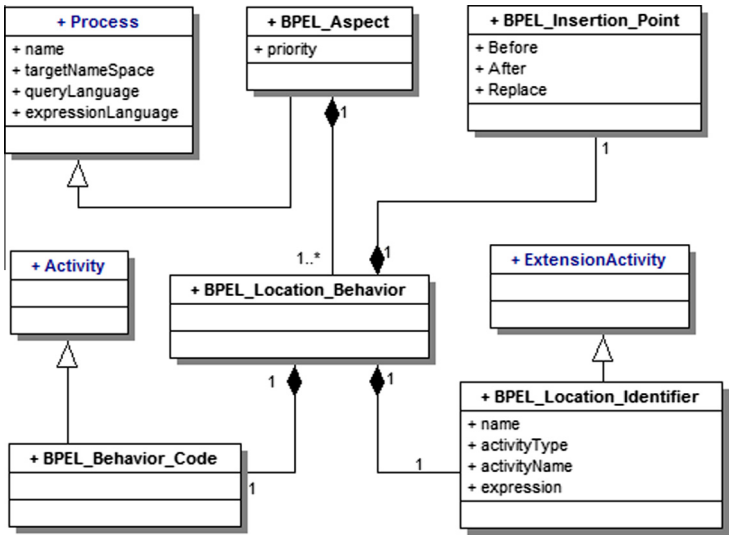
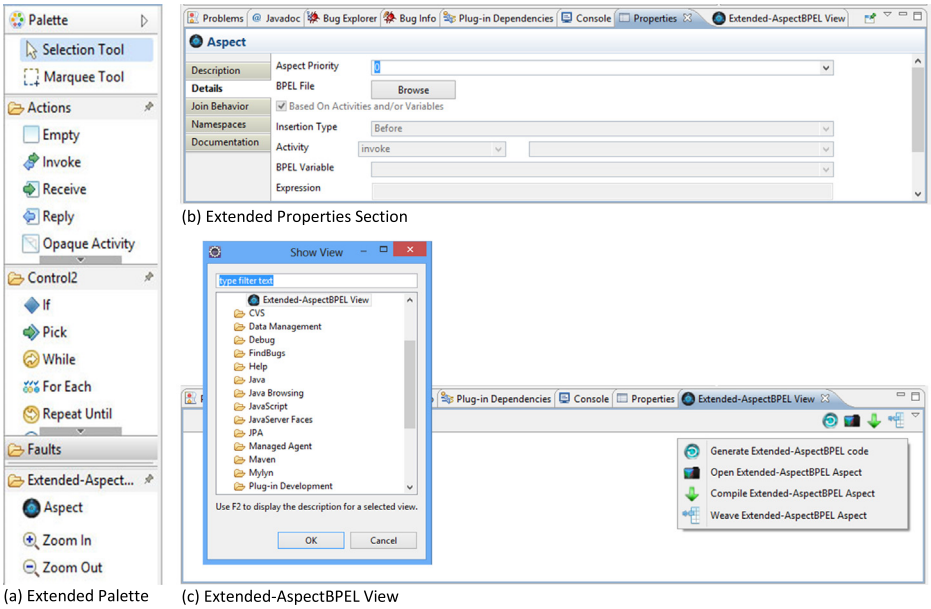


Fig. 3. WS-BPEL meta-model extension.



(a) Extended Palette with Aspect Element

(b) Extended Properties Section

(c) Extended-AspectBPEL View

Fig. 4. E-AspectBPEL Eclipse plug-ins.

4.1. Modeler

This modeler not only allows the specification of new behaviors, business rules and security policies, but also context-aware polices to reach high adaptability of Web services composition. It implements the aspect-oriented extension that we presented in Section 3.2 to offer high level specification of aspects by offering seamless, easy and visual solution to design the aspects requirements.

4.2. Generator

This generator automatically generates the code corresponding to specified aspects. It is implemented based on the DOM parser for XML, using the Java language. As depicted in Fig. 5, it takes the designated aspect model that must be consistent with the proposed E-AspectBPEL Meta-Model, extracts the E-AspectBPEL nodes and formulate the E-AspectBPEL code based

on our proposed language grammar. Precisely, the generator sets the name and the priority level values of the BPEL_Aspect based on the model. Afterwards, it extracts the BPEL_Insertion_Point element. Then, it fetches the activityName and the activityType combination to build up the BPEL_Location_Identifier element. Finally, it derives the BPEL activities nodes of the aspect module to compose its behavior. To recall these elements in the E-AspectBPEL language, you may refer to Sections 3.1 and 3.2.

4.3. Compiler

After writing the E-AspectBPEL language grammar using ANTLR (Another Tool for Language Recognition) V3.0 [16], we created the corresponding compiler, which performs a syntactic and semantic check of the generated E-AspecctBPEL code to make sure it is compatible with the new E-AspectBPEL language constructs, and also verifies the behavior code of these aspects against the WS-BPEL schema definition [17] to ensure its correctness.

4.4. Weaver

In previous achievement [3], we developed a similar tool that allows the automatic and dynamic integration of aspects in BPEL processes. However, in this work, we extend and upgrade this tool with three utilities. First, it supports the injection of several aspects in the BPEL process. Second, it includes a sorting mechanism, whereby it orders these aspects based on their priority level to avoid behavioral conflicts. It extracts the insertion point and the location identifier from the aspect code, which pinpoint to the exact spot in the BPEL process where the aspect should be injected. Third, it allows activating dynamically the aspect behavior based on the Activation_Condition element.

5. Formal verification

Other than performance analysis, none of the existing relevant works [3,4,6–8] has verified the proposed approach, which is a remarkable limitation. Per contra, in this section, we define the essential properties that should be verified after weaving new aspects in the Web services composition as well as the verification mechanism that we applied in order to verify our proposition.

5.1. Verification properties

First, we have to validate that the process behavior is consistent with the intended behavior after the injection of new aspects. For instance, if we look at the scenario in Section 6, integrating any new feature in the Web services composition should keep the original behavior of invoking the payment service after the reservation Web services get invoked, fully operational. Let us assume that new authentication aspect is weaved just before calling the payment service and not at the beginning of the process. This weaving can be the result of error in the weaving process itself or mistake in the aspect specification. In this case, an unauthenticated user will have the ability to call the reservation services, while the payment will never get invoked. For security purposes, such sequence of transactions should be atomic, and in case of illegitimate user, the process should roll back all previous transactions (i.e., reservations). However, BPEL does not support such flexibility. Therefore, making sure that such contradictory case will not occur will be one of the properties to be verified.

Second, we have to verify that the integration of these aspects did not cause a deadlock problem in the BPEL process and all the activities in the composition remains reachable. For example, after integrating the authentication aspect, the process has to preserve its ability to respond to any request made by the user despite its status (e.g., authenticated or not). Subsequently, it should be able to always reach its final state whereby it sends an error message in case the user is not authenticated, or another confirmation message otherwise.

Third, we have to make sure that the aspects have been weaved correctly and the process behaves as intended. For instance, adding authentication should guarantee that only legitimate users can invoke the offered services. Even in other situation, for example when injecting discount and logging aspects, the process should guarantee that checking for discount will not proceed logging. Otherwise, the process will end up with erroneous values. For instance, when the discount is applied, faulty data related to total fees will be logged.

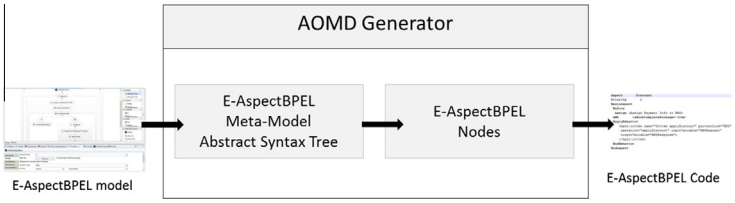


Fig. 5. Code generation scheme.

Finally, we have to prove that the conflict problem has been resolved. Taking the example of encryption and logging aspects, the solution has to assure that the data will not be logged unless it has been encrypted first, and this through invoking encryption before logging, and not vice versa. Same for the example of policies with combining algorithms First-Applicable, the solution has to assure that the order of checking the policies is preserved. A detailed explanation is illustrated in the case study, Section 6.3.

5.2. Formal verification mechanism

A well-known method to verify these objectives is through model checking [18–21]. In our approach, the BPEL is transformed to Petri nets and the objectives are expressed computational tree logic or in LTL [22] formulas, then the model checker verifies whether the objectives are met or not. Many approaches follow this mechanism [23,24] to analyze BPEL processes. In our approach, the BPEL is process is transformed to open workflow nets (oWFNs), which is a formal model for BPEL processes and eventually to PNML [25] using the BPEL2oWFN [26] tool. Petri net is a mathematically based technique for modeling and verifying software artifacts and it provides clear and precise formal semantics, an intuitive graphical notation, and many techniques and tools for their analysis. Recent researches indicate that Petri net is powerful enough to describe the behavioral features of service composition, and the analysis of Web services composition based on Petri nets has been studied by many recent approaches [23,27]. Then Petri nets are converted to KTZ through Tina-selt, a conversion package in TINA model checker [28] that we are using. KTZ is the binary format for Kripke transition system of TINA. In parallel, we map the objectives to LTL formulas in order to be checked by the model checker. A boolean result, that reflects whether the BPEL process fulfill the defined property, is conducted when the verification process is done.

6. Case study and experiments

To better illustrate our approach, we suggest a Travel Booking System (TBS) as a running example throughout the rest of the paper. TBS consists of a Flight Web Service that allows the user to book a ticket on a certain flight, an Hotel Web Service that enables the user to book a hotel room, a Car Web Service offering the ability to reserve car during the trip, and finally, a Banking Web Service to perform online payments for the reservation fees. The user may invoke one, two or even the three reservation Web services before getting to the online payment, therefore different scenarios might occur. However, to make the process figure fits within the page frame, we represent the offered services by "Any TBS Booking Service" as depicted in Fig. 6. Once a client invokes the TBS BPEL process, the latter assigns the request to the corresponding TBS Web service (i.e., Flight, Hotel or Car Web service) to invoke the relevant reservation service accordingly. Once done, it assigns the client payment information to the Banking Web service to process the payment of the booking fees. Finally, the TBS BPEL process assigns the appropriate notification back to the client. TBS imposes invoking the payment service whenever any of the booking Web services (i.e., Flight, Hotel and Car) get invoked to achieve an atomic sequence of transactions.

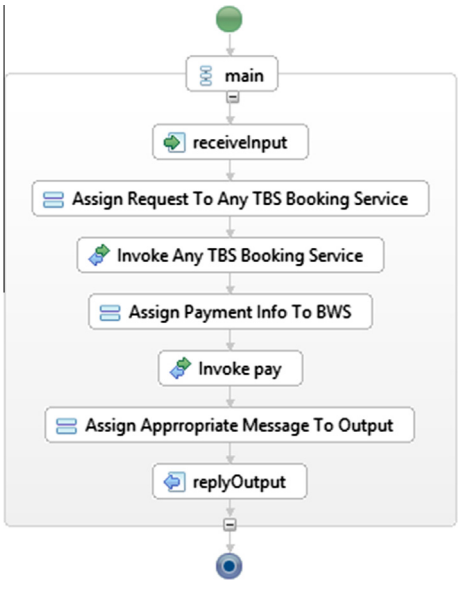


Fig. 6. Travel booking system.

6.1. E-AspectBPEL aspect specification

In this section, we show the specification security, behavioral, and context aware aspects to be hardened in the TBS BPEL process. Using our Modeler, the aspects behavior and their corresponding properties (i.e., priority level, location identifier, activation condition) can be defined in a seamless manner.

6.1.1. Authentication: Security Aspect

Behavior: It determines if the user is legitimate or not by verifying the credentials entered by the user. As shown in Fig. 7, if the authentication fails, the process assigns an appropriate error message to the BPEL output variable, forwards it back to the user and then stops. Otherwise the process continues its execution by invoking the appropriate Web service(s).

Properties: The properties section in Fig. 7 shows the Authentication aspect properties specification. *Priority level = 0, Insertion point = After and Location identifier = receive <receiveInput>*. Based on these properties, this aspect will be weaved right after receiving the request from the user (i.e., At the beginning of the BPEL process). The priority level is neglected since Authentication is the only aspect that will be injected at this joinpoint in the process having no effect on other aspects.

6.1.2. Encryption: Security Aspect

Behavior: It contains an invoke activity that calls the encrypt method in order to encrypt the assigned data.

Properties: As shown in Fig. 8, it has the following properties: *Priority level = 2, Insertion point = Before and Location identifier = assign <Assign Payment Info To BWS>*.

6.1.3. Logging: Behavioral Aspect

Behavior: It logs the exchanged data by invoking the log method.

Properties: The following are the associated properties as presented in Fig. 9: *Priority level = 3, Insertion point = Before and Location identifier = assign <Assign Payment Info To BWS>*.

6.1.4. Discount: Context-Aware Aspect

Behavior: A reduction is applied on the total booking fees by invoking applyDiscount method as shown in Fig. 10.

Properties: To benefit from this discount, the user should book a complete travel package. Thus, the activation of this aspect is conditioned by the variable *isBookCompletePackage*, in the BPEL process, that must have the value true. Subsequently, these are the properties: *Priority level = 1, Insertion point = Before and Location identifier = assign <Assign Payment Info To BWS> Activation Condition = $isBookCompletePackage = true*. It is worth to mention that our framework is responsible of parsing the original BPEL process, where the aspect will be applied, in order to extract the existing variables to formulate a context-aware activation condition of particular aspect.

The last three aspects share the same join point in the BPEL process. This may cause a conflict problem among them if they are not weaved in the appropriate order, and consequently lead to both security violation and erroneous data as discussed in Section 5.1. Therefore their priority levels are important.

6.2. E-AspectBPEL aspect integration

In order to integrate the pre-defined aspects in the Web services composition of the TBS process, the followings are the steps to be followed.

- **Generate E-AspectBPEL code from the design:** Having the aspects specified, their corresponding code is automatically generated using our E-AspectBPEL Generator. For space restrictions we show only one generated aspect code. Fig. 11 depicts the generated E-AspectBPEL code of the Discount aspect.
- **Compile E-AspectBPEL code:** The generated code is compiled to make sure that no element is missed in the aspect specification phase.
- **Weave E-AspectBPEL aspect:** Here comes the role of the Weaver. First, it orders the Discount, Logging and Encryption aspects based on their priority level to prevent the conflict between them. Then, it matches the location identifier of all the aspects with the BPEL activities. Based on the insertion types and the stated locations, these aspects are weaved in the BPEL process. Fig. 12 shows the BPEL process after the weaving.

6.3. Formal verification

In this section, we verify the properties that we discussed in Section 5.1. Fig. 13 shows the Petri net generated from the BPEL process of our TBS. In the Petri nets model, each activity in the process is mapped to a Petri net transition that is also followed by a place. For example, the receive activity is mapped to transition t1 and followed by p2 that represents the state of the system after calling this particular activity. The arrows represent the arcs in Petri net, which create a connection between the places and the transitions. In this model, the process calls any of the TBS booking services and the online payment, and assigns a confirmation message to the user.

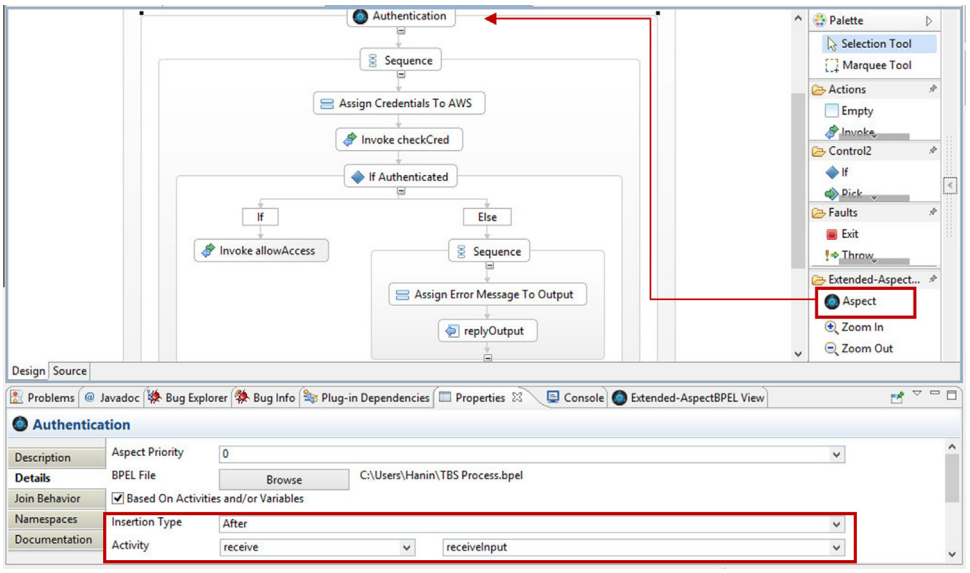


Fig. 7. Authentication aspect specification.

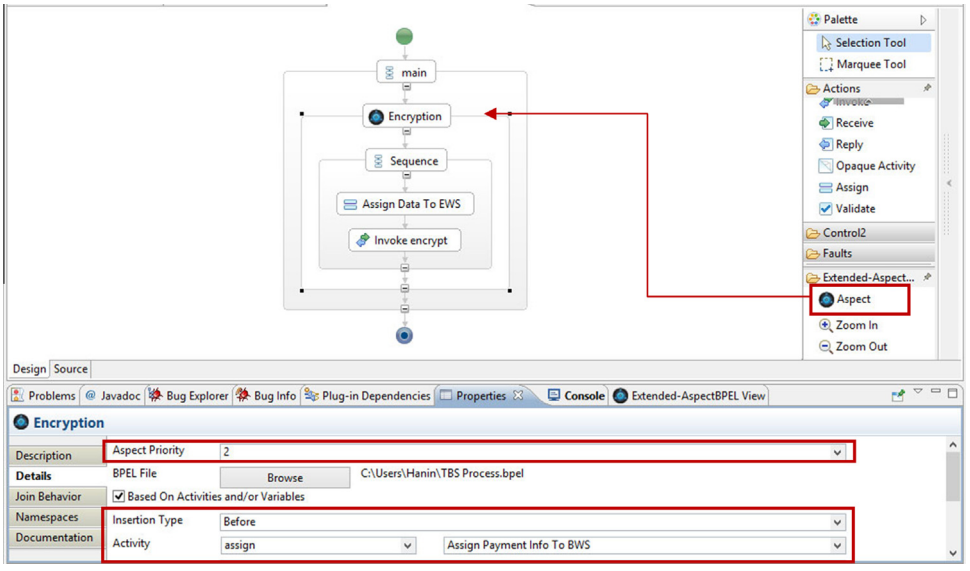


Fig. 8. Encryption aspect specification.

Fig. 14 illustrates the Petri net corresponding to the resulting BPEL process in Fig. 12. The Petri net model consists of 19 transitions and 20 places. Transitions 3, 4 and 5 correspond to the Authentication aspect so that if the user is not authenticated, the process sends an error message to the user and none of the Web services will get invoked. Otherwise, it grants the user access to any of the TBS services. Finally, transitions 11, 13 and 15 correspond to the Discount, Encryption and Logging aspects respectively. All aspects are accurately weaved in the process following the correct order and priority. This TBS process is free of any problem. On the other hand, as a contradictory example, we created a sample of a TBS BPEL process that reveals security violation, conflict and inconsistency problems, Fig. 15. Contrary to our process, in this example, the Authentication aspect appears at the end of the TBS so that any user, authorized or not, can access the system services. Such case may occur in case the reservation services need not to authenticate the client, while the latter has to be authenticated to process the payment of the booking fees, which is one of the common situations. Further, the Discount, Encryption and Logging aspects are not applied in the correct order. To start the verification, we generate the KTZ file corresponding to each Petri net model via the TINA tool and then, we map systematically the needed properties to LTL formulae. TBS-

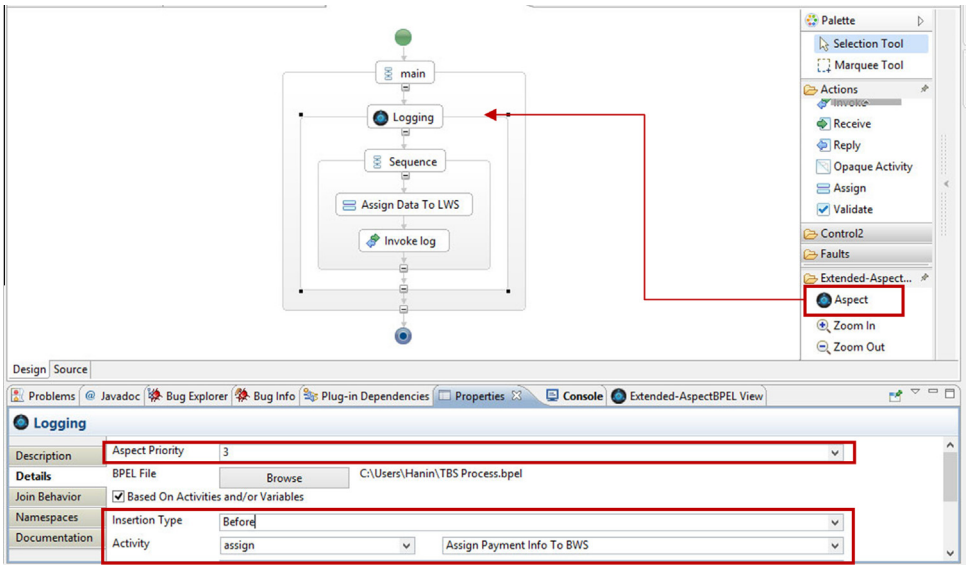


Fig. 9. Logging aspect specification.

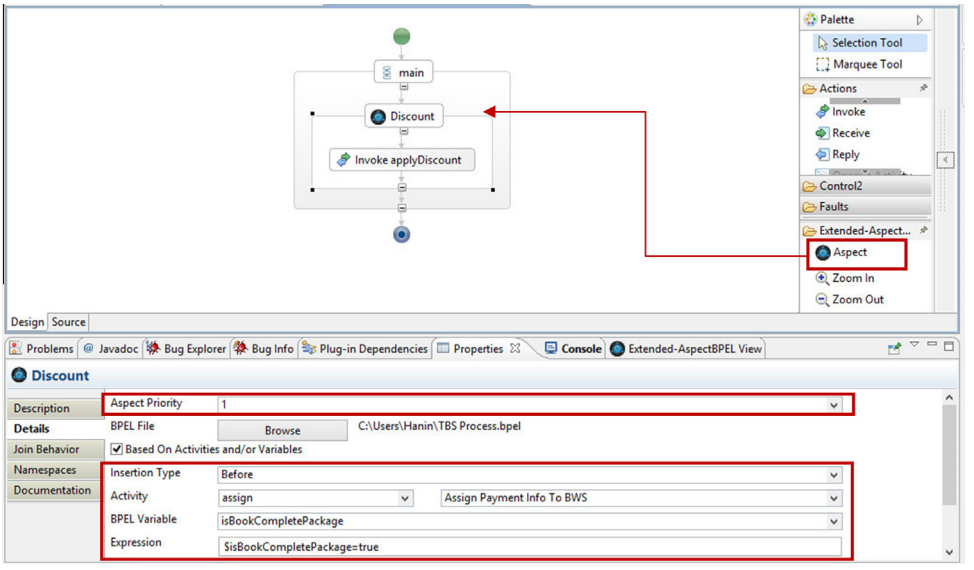


Fig. 10. Discount aspect specification.

```
Aspect       Discount
Priority       1
BeginAspect
 Before
  Assign <Assign Payment Info to BWS>
 AND       isBookCompletePackage='true'
 BeginBehavior
    <bpel:invoke name="Invoke applyDiscount" partnerLink="BWS"
     operation="applyDiscount" inputVariable="BWSRequest"
     outputVariable="BWSResponse">
    </bpel:invoke>
 EndBehavior
EndAspect
```

Fig. 11. Generated E-AspectBPEL code of the discount aspect.

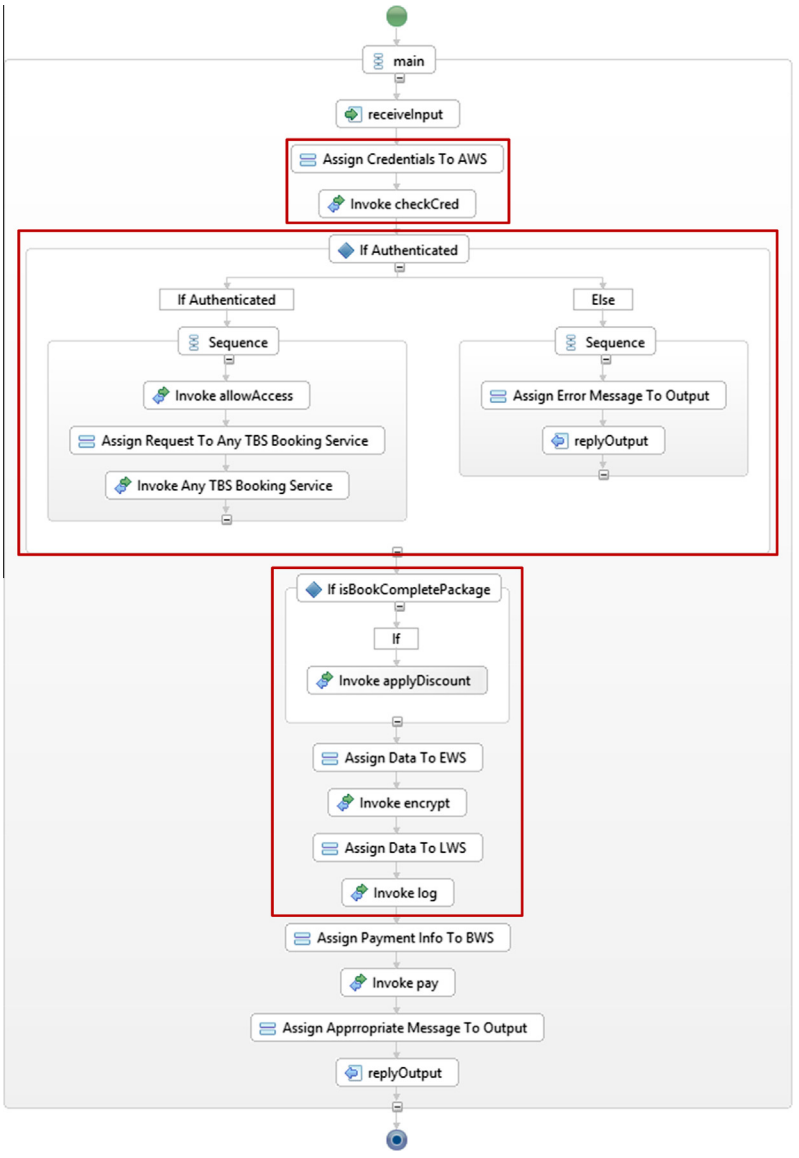


Fig. 12. Highly context-adaptable and conflict-free TBS BPEL process.

Original.ktz corresponds to the Petri nets model in Fig. 13, TBS-Correct.ktz is generated from the Petri nets in Fig. 14 and TBS-Flawed.ktz is conducted from the Petri nets in Fig. 15.

To help understanding the LTL formulae, we interpret the synopsis of its alphabet that is used in this case study. In unary operators, the symbol "◯" is used to define the **"Next"** state in the Petri nets model, "□" stands for **"Always"** and "◊" stands for **"Eventually"**. In binary operators, "∨" stands for the logical **"Or"**, "→" represents an **"Implication"** and "¬" for **"Negation"**. For more details about the Linear Temporal Logic, its syntax, semantics and its state/event-LTL form, you may refer to [22].

First, integrating any new behavior in the Web services composition should keep the original behavior operational and consistent with the intended behavior. For example, our TBS imposes that the Banking Web Service must get invoked in case any of the other Web services got invoked. However, after weaving new aspects, this rule may not hold as discussed previously. Such case is illustrated in Fig. 15, where the user will be able to invoke any of the reservation Web services. But if he/she is not authenticated, the payment service will never get invoked, breaking the declared rule. In order to verify that this is not the case in our model (Fig. 14), we used the LTL expressions in Table 1.

p4 and p10 in Table 1 map the places in the BPEL processes after calling any of the reservation Web services of the TBS, while **p6 and p18 correspond to the places after invoking the payment service**. Accordingly, the analysis of the LTL formulas with the model checker responses implies that in the original TBS BPEL process, in case any of the reservation

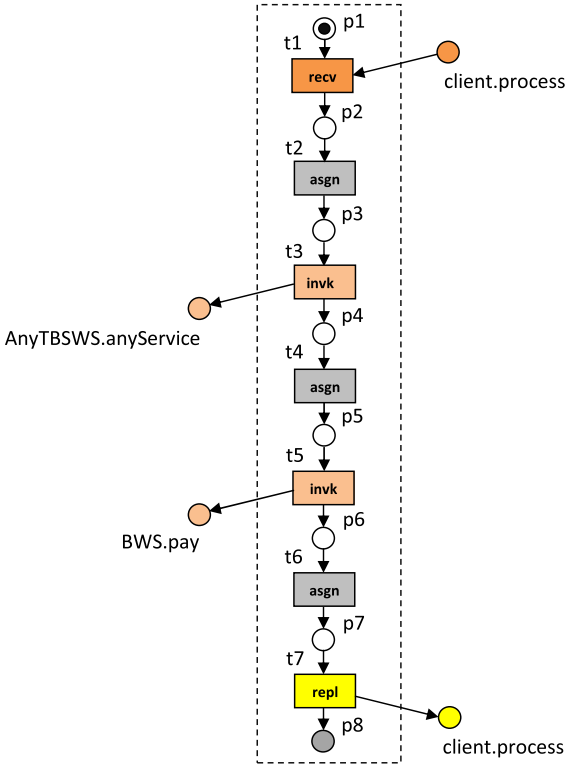


Fig. 13. Petri Nets of the original TBS BPEL process.

services is invoked, the payment will get invoked. This is due to the fact that **p4 will always lead to p6**. However, for the second LTL expression that examines whether invoking reservation will lead to the invoke of the payment service, the answer is **FALSE**. This means that sometimes the process will not call the pay service even when other services are invoked, which breaks the rule. This can be argued by the fact that whenever the user is not authenticated, another path (**p15** → **p17** → **p20**) will be taken by the TBS process and therefore, **p18** is never reached even though **p4** might have been attained. On the other hand, based on the **TRUE** response for the last LTL expression that corresponds to our model, **p18** is always reached whenever **p10** is. Consequently, our solution has no side effect on this behavior of the Web services composition.

Second, it is really important to make sure that weaving the security and business aspects did not cause a deadlock problem that blocks the execution of the composed services. Consequently, we have to verify that all the activities in the BPEL process are reachable. In this example, we will show that our approach assures that the process is able to respond to any user request. Table 2 shows the devised LTL expression used in order to achieve such verification as well as the response of the model checker. The LTL expression, in the table, states that **always p2 will eventually lead to p5, p10 or to p6, p8, and eventually achieve p20**. In other words, if it is the case of authenticated user, he/she will be able to invoke any of the offered services and get back a confirmation from the process. Otherwise, an error message will be sent to the user. According to the model checker, the response is **TRUE**, which assures that after weaving the aspects, our TBS process is able to react according to any user request regardless his/her status ruling out any possibility of a deadlock problem.

Third, to verify that the TBS BPEL process is conflict-free, we have to prove that the aspects are weaved in the correct order as follows: Discount, Encryption then Logging. Based on the first row in Table 3, invoking the **Discount** aspect mapped by **p12** is eventually followed by the **Encryption** invocation mapped to **p14** and the latter is also followed by the **Logging** aspect at **p16**, which reserve the correct order. On the other hand, this order is not respected in the Petri net model of Fig. 15 since the model checker returns **FALSE** when verifying the same property in the second formula.

Finally, we verify that the BPEL process behaves correctly based on the Authentication, Logging and Discount aspects. Hence, we confirm that only authenticated users can get access to the system and invoke the services offered by the Web services composition, and we verify that the process will not fall in erroneous status by logging faulty values. Table 4 illustrates the used LTL expressions. In the Petri net model of Fig. 14, **p5** represents the case of an **authenticated user** and **p6** the one of **non authenticated user**. **p10** represents the place in the process **after invoking any of the reservation Web services**. Whereas this place is mapped to **p4** in Fig. 15. In both models **p2** represents the place after receiving the user request. When checking if any user can access the system in the first formula, the model checker response was **FALSE**. In the second formula

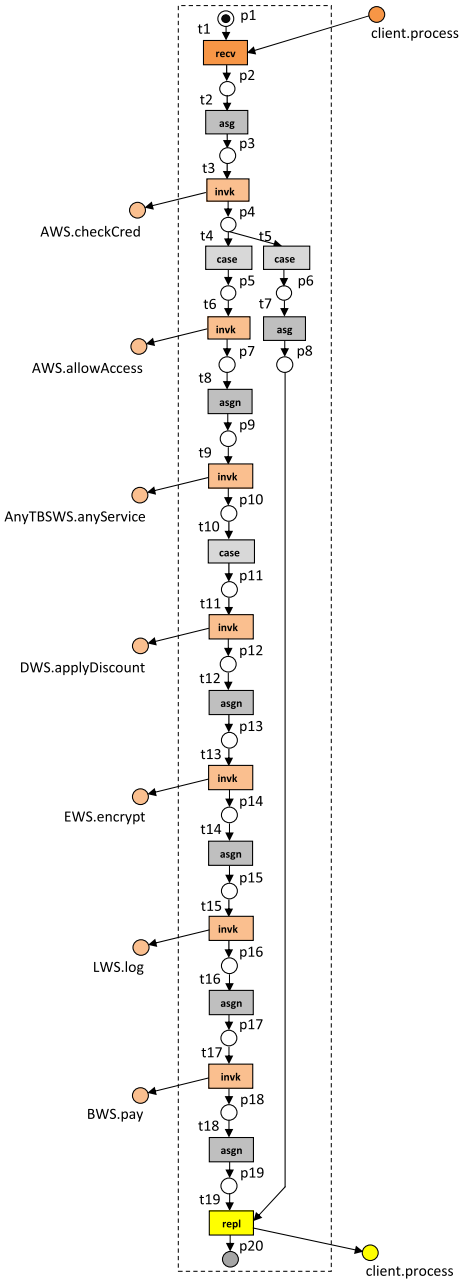


Fig. 14. Petri Nets of the correctly weaved TBS BPEL process.

examining whether authenticated users can access the system services, the response was **TRUE**. Same answer when applying similar test for non authenticated users in the third formula stating that in non of the cases, non valid users can invoke the services. Analyzing these results assure that in our approach only authenticated users can get access to TBS. On the other hand, the fourth formula in Table 4 shows that in the erroneous TBS BPEL process in Fig. 15, anyone is able to gain access.

Moreover, **p12** and **p16**, **p8** and **p6** in the LTL formulas on rows 5, 6, 7 and 8 represent the **"invoke" of the Discount and Logging** aspects in Figs. 14 and 15 respectively. Rows 5 and 6 assure that in our correctly weaved TBS BPEL process, always **p12** will precede **p16** and in none of the cases, the inverse is applied. In other words, always the discount will occur before the logging (in case the discount is applied) so that if the discount is applied, the correct value of the total fees will be recorded. However, this is not the case in the contradictory process based on the model checker responses for the formulas in rows 7 and 8.

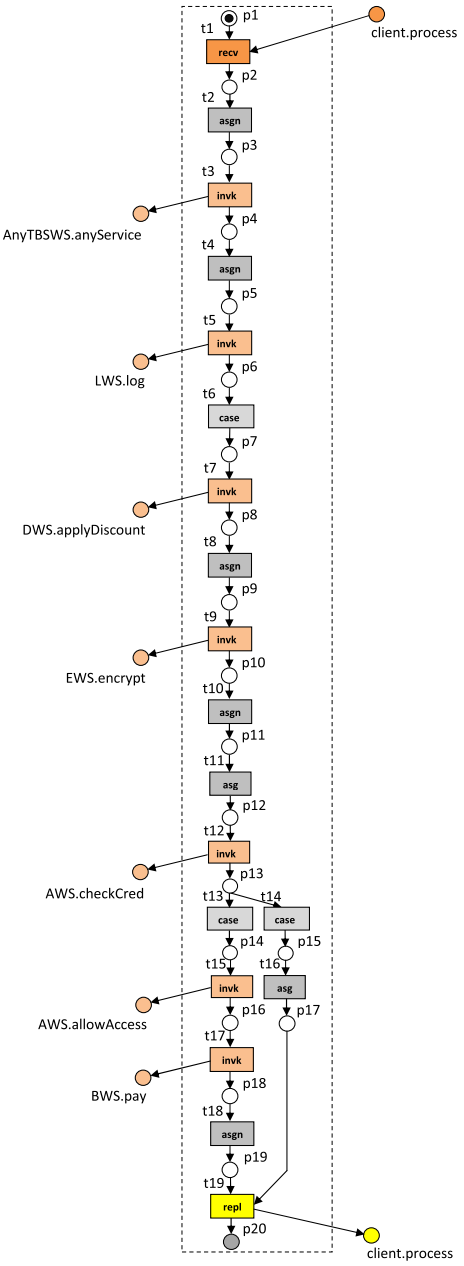


Fig. 15. Petri Nets of a weaved TBS BPEL process enclosing security, conflict and inconsistency problems.

Table 1
Process behavior consistency verification.

KTZ file	LTL expression	Model checker response
TBS-Original.ktz	$\Box(p4 \rightarrow \Diamond\, p6)$	TRUE
TBS-Flawed.ktz	$\Box(p4 \rightarrow \Diamond\, p18)$	FALSE
TBS-Correct.ktz	$\Box(p10 \rightarrow \Diamond\, p18)$	TRUE

6.4. Discussion and experimental results

As illustrated in Fig. 12, the Authentication aspect is injected at the beginning of the process to ensure that only legitimate users are able to access the TBS services. Moreover, the Discount, Encryption and Logging aspects are weaved in the correct

Table 2
Deadlock-free and reachability verification.

KTZ file	LTL expression	Model checker response
TBS-Correct.ktz	□ (p2 → ◊ ((p5 → ◊ p10) ∨ (p6 → ◊ p8)) → ◊ p20)	TRUE

Table 3
Conflict-free verification.

KTZ file	LTL expression	Model checker response
TBS-Correct.ktz	□ (p12 → ◊ p14 → ◊ p16)	TRUE
TBS-Flawed.ktz	□ (p8 → ◊ p10 → ◊ p6)	FALSE

Table 4
Weaving accuracy verification.

	KTZ file	LTL expression	Model checker response
1	TBS-Correct.ktz	□ (p2 → ◊ p10)	FALSE
2	TBS-Correct.ktz	□ (p5 → ◊ p10)	TRUE
3	TBS-Correct.ktz	□ ¬ (p6 → ◊ p10)	TRUE
4	TBS-Flawed.ktz	□ (p2 → ◊ p4)	TRUE
5	TBS-Correct.ktz	□ (p12 → ◊ p16)	TRUE
6	TBS-Correct.ktz	□ ¬ (p16 → ◊ p12)	TRUE
7	TBS-Flawed.ktz	□ (p6 → ◊ p8)	TRUE
8	TBS-Flawed.ktz	□ ¬ (p8 → ◊ p6)	TRUE

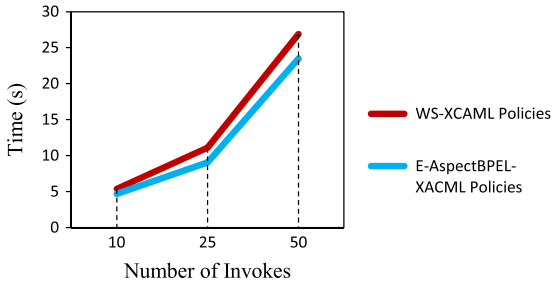


Fig. 16. Process execution time with XACML policies.

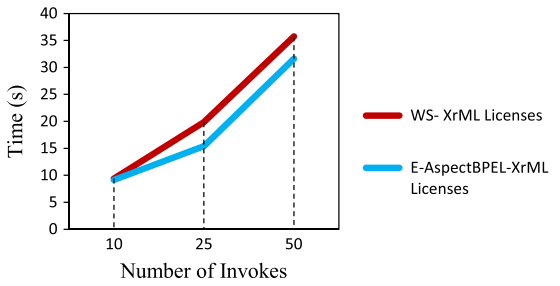


Fig. 17. Process execution time with XrML licenses.

order avoiding behavioral conflicts. The Discount aspect is weaved first, yet not activated unless the isBookCompletePackage variable value is equal to true. In other words, only users who have booked a complete package, that consists of a flight ticket, hotel room and a car can benefit from the discount promotion. This is reflected by the automatically generated If condition to cater with this requirement. It is worth mentioning that the Weaving process is atomic. In other words, in case a compilation error occurs in any of the related aspects (e.g., Discount, Encryption and Logging in this case), the weaving stops and prompts the user to make changes in the erroneous model and recompile it before proceeding.

Adding activities for checking policies' rules of multiple Web services to the single process orchestrating them may impose significant execution overhead to that process, particularly when composed together to form policies with large number of rules. Hence, to better demonstrate the effectiveness of our approach, we find it relevant to conduct performance analysis to measure the variation in the execution time between a BPEL process with policies enforced on the Web services side and a BPEL process with E-AspectBPEL policies enforced at the process level, which will give a good idea about the overhead of the proposed approach. In this regards, two experiments based on security licenses and policies have been performed. The security licenses are based on XrML language and engine [29], while the security policies are based on XACML language and Sun PDP engine [15]. We opted to use these standard languages because they are widely used in Web services.

The first impacting factor during the experiments is the size of the BPEL process. It is determined by the number of invokes to the distributed Web services, which is increased progressively until reaching 50 invokes. It refers to the number of services that are called in the BPEL process. The size of a policy/aspect, which is determined by the number of pointcuts and number of rules, has also an overhead effect on the total time of execution at each run. Modifying (i.e. increasing or decreasing) the number of policy rules will affect equally both approaches. Hence, we have taken a constant number of rules in all policies in order to verify the overhead in the different studied contexts. The experimental results in Figs. 16 and 17 show that the overhead imposed by our approach is less since the policies verification is done selectively at specific pointcuts in the BPEL process rather than unnecessary checks at each invokes, which is the case of policies enforced on the Web services side. In case of E-AspectBPEL policies at the process level, an additional impacting factor should be taken into consideration, which is the time taken to compile and weave the policy. Nevertheless, it has no runtime effect since it is performed offline.

7. Conclusion and future work

In Web services composition, many changes are likely to arise after deployment. For instances, partner services may go down or get updated, and even new policies to govern the composition might be added. To accommodate with these changes, today's business environments are challenged by the need for continuous adaptation of business processes. Yet, like other existing Web service-based process composition approaches, BPEL does not provide automated support for this end. In this context, many existing approaches leverage Aspect-Oriented Programming to support BPEL with the needed adaptability. However, our approach improves the related literature in several aspects. First, we offer new Aspect-Oriented grammar that on one hand offers higher adaptability and on the other hand prevent behavioral conflicts between different aspects in the Web services composition. Second, we offer Aspect-Oriented extension for WS-BPEL to allow high level specification of aspects. Finally, we formally verify our proposed approach. Real life case study, illustrative examples, verification and experimental results, the integration of the approach in well-known and widely used commercial BPEL development environment, Eclipse, all demonstrate the feasibility and effectiveness of this work. As for future work, we will investigate novel model and algorithms to verify and prevent conflicts between aspects at the specification level, a priori to their integration into the BPEL process.

Acknowledgments

This work is supported by CNRS Lebanon, Lebanese American University (LAU), NSERC Canada and Khalifa University of Science, Technology & Research (KUSTAR).

References

[1] Mizouni Rabeb, Serhani Mohamed Adel, Dssouli Rachida, Benharref Abdelghani, Taleb Ikbal. Performance evaluation of mobile web services. In: 2011 Ninth IEEE European conference on web services (ECOWS). IEEE; 2011.
[2] WSBPEL TC. Web services business process execution language version 2.0 <http://docs.oasis-open.org/wsbpel/2.0/wsbpel-v2.0.html> [accessed 2015.01.07].
[3] Mourad Azzam, Ayoubi Sara, Yahyaoui Hamdi, Otrok Hadi. A novel Aspect-Oriented BPEL framework for the dynamic enforcement of web services security. Int J Web Grid Serv 2011;8(4):361–85.
[4] Tout Hanine, Mourad Azzam, Otrok Hadi. XrML-RBLicensing approach adapted to the BPEL process of composite web services. Service Oriented Comput Appl 2013;7(3):217–30.
[5] Tout Hanine, Mourad Azzam, Yahyaoui Hamdi, Talhi Chamseddine, Otrok Hadi. Towards a BPEL model-driven approach for web services security. In: 2012 Tenth annual international conference on privacy, security and trust (PST); 2012. p. 120–7.
[6] Yahyaoui Hamdi, Mourad Azzam, Almulla Mohamed, Yao Lina, Sheng Quan Z. A synergy between context-aware policies and AOP to achieve highly adaptable web services. Service Oriented Comput Appl 2012;6(4):379–92.
[7] Charfi Anis, Mezini Mira. AO4BPEL: an aspect-oriented extension to BPEL. World Wide Web 2007;10(3):309–44.
[8] Braem Mathieu, Gheysels Dimitri. History-based aspect weaving for WS-BPEL using Padus. In: ECOWS, Citeseer; 2007. p. 159–67
[9] Mourad Azzam, Laverdière Marc-André, Debbabi Mourad. An aspect-oriented approach for the systematic security hardening of code. Comput Security 2008;27(3):101–14.
[10] Golbeck Ryan M, Selby Peter, Kiczales Gregor. Late binding of AspectJ advice. In: Objects, models, components, patterns. Springer; 2010. p. 73–191.
[11] Durr Pascal, Bergmans Lodewijk, Aksit Mehmet. Static and dynamic detection of behavioral conflicts between aspects. Runtime Verification; 2007. p. 38–50.
[12] Braem Mathieu, Verlaenen Kris, Joncheere Niels, Vanderperren Wim, Van Der Straeten Ragnhild, Truyen Eddy, et al. Isolating process-level concerns using Padus. Springer; 2006.

[13] Patiniotakis Ioannis, Papageorgiou Nikos, Verginadis Yiannis, Apostolou Dimitris, Mentzas Gregoris. An aspect oriented approach for implementing situational driven adaptation of BPMN2.0 workflows. In: Business process management workshops. Springer; 2013. p. 414–25.
[14] Kiczales Gregor, Hilsdale Erik, Hugunin Jim, Kersten Mik, Palm Jeffrey, Griswold William G. An overview of AspectJ. In: ECOOP 2001-object-oriented programming. Springer Berlin Heidelberg; 2001. p. 327–54.
[15] XACML TC. OASIS eXtensible Access Control Markup Language (XACML) Version 2.0 <https://www.oasis-open.org/committees/tc_home.php?wg_abbrev=xacml> [accessed 2015.01.07].
[16] Terence Parr. The ANother Tool for Language Recognition <http://www.antlr.org/index.html> [accessed 2015.01.07].
[17] OASIS. Schema for executable process for WS-BPEL 2.0 <http://docs.oasis-open.org/wsbpel/2.0/CS01/process/executable/ws-bpel_executable.xsd> [accessed 2015.01.07].
[18] Mizouni Rabeb, Tahar Sofiène, Curzon Paul. Hybrid verification integrating HOL theorem proving with MDG model checking. Microelectron J 2006;37(11):1200–7.
[19] Kova Melissa, Bentahar Jamal, Maamar Zakaria, Yahyaoui Hamdi. A formal verification approach of conversations in composite web services using NuSMV. In: Proceedings of the 2009 conference on new trends in software methodologies, tools and techniques: proceedings of the eighth SoMeT_09. IOS Press; 2009. p. 245–61.
[20] Bentahar Jamal, Yahyaoui Hamdi, Kova Melissa, Maamar Zakaria. Symbolic model checking composite web services using operational and control behaviors. Expert Syst Appl 2013;40(2):508–22.
[21] Kholy Warda El, Bentahar Jamal, Menshawy Mohamed El, Qu Hongyang, Dssouli Rachida. Modeling and verifying choreographed multi-agent-based web service compositions regulated by commitment protocols. Expert Syst Appl 2014;41(16):7478–94.
[22] Bae Kyungmin, Meseguer José. State/event-based LTL model checking under parametric generalized fairness. In: Computer aided verification. Springer; 2011.
[23] Xiong PengCheng, Fan YuShun, Zhou MengChu. A Petri net approach to analysis and composition of web services. IEEE Trans Syst Man Cybernet. Part A: Syst Humans 2010;40(2):376–87.
[24] Lohmann Niels, Massuthe Peter, Stahl Christian, Weinberg Daniela. Analyzing interacting BPEL processes. Springer; 2006.
[25] Kindler Ekkart. The Petri Net Markup Language and ISO/IEC 15909-2: concepts, status, and future directions. In: Entwurf komplexer Automatisierungssysteme, vol. 9; 2006. p. 35–55.
[26] Lohmann Niels, Gierds Christian, Znamirowski Martin. BPEL2oWFN <http://www.gnu.org/software/bpel2owfn/> [accessed 2015.01.07].
[27] Cardinale Yudith, Haddad Joyce El, Manouvrier Maude, Rukoz Marta. Web service composition based on Petri nets: review and contribution. In: Resource discovery. Springer; 2013. p. 83–122.
[28] Berthomieu Bernard, Popova-Zeugmann Louchka. Time Petri nets: theory, tools and applications <http://www2.informatik.hu-berlin.de/~popova/tutorial.html> [accessed 2015.01.07].
[29] TechNet. XrML <http://technet.microsoft.com/en-us/library/cc747717(v=ws.10).aspx> [accessed 2015.01.07].

Hanine Tout received her M.Sc. degree in Computer Science from the Lebanese American University. The topics of her research activities are Web services security, BPEL, Aspect-Oriented Programming, and model checking.

Azzam Mourad is an Assistant Professor of Computer Science at the Lebanese American University. He holds a Ph.D. in ECE from Concordia University and M.Sc. degree in Computer Science from Laval University. He is currently working on information security, Web services, vehicular networks, and formal semantics. He is serving as TPC and reviewer of several prestigious conferences and journals.

Chamseddine Talhi is an Associate Professor in the Department of software engineering and information technologies at ''Ecole de technologie superieure'', Montreal, Canada. He is leading a research group that investigates Smartphone and embedded systems security. Recently, he is involved in Cloud security and secure sharing of embedded systems. Chamseddine holds a Ph.D. in Computer Science from Laval University, Quebec, Canada.

Hadi Otrok holds an Associate Professor position in the Department of ECE at Khalifa University. He received his Ph.D. in ECE from Concordia University. He works on network and computer security, game theory and mechanism design. He chaired several security-related conferences. Moreover, he is a TPC member of several prestigious conferences and reviewer of several IEEE and Elsevier journals.