SOCA (2013) 7:217–230
DOI 10.1007/s11761-013-0127-5

ORIGINAL RESEARCH PAPER

XrML-RBLicensing approach adapted to the BPEL process of composite web services

Hanine Tout · Azzam Mourad · Hadi Otrok

Received: 30 July 2012 / Revised: 25 December 2012 / Accepted: 4 January 2013 / Published online: 2 February 2013

Abstract Web service orchestration represents an open and standards-based approach for connecting web services together leading to higher level of business processes. Business Process Execution Language (BPEL) engines are designed to handle this orchestration. However, web service compositions into BPEL suffer from several non-functional requirements such as security. To address this problem, we propose in this paper a novel approach that is based on a harmony between the licensing concept offered by eXtensible rights Markup Language (XrML), aspect-oriented programming (AOP), and web service compositions in BPEL. Our proposed approach, based on XrML, offers the ability to associate security licenses with activities offered by the composite web services. It allows to automatically generate BPEL aspects depending on the developed licenses, to separate between crosscutting concerns of the composed web services, and provides an easy way to include and update the non-functional requirements (e.g., security) into a BPEL process. It offers also the ability to validate the licenses, at runtime and without affecting the business logic of this model. To evaluate our approach, we have developed an inventory control system (ICS) sample that is composed of several web services. Case study and performance analysis are presented to demonstrate its feasibility as well.

Keywords Web services · XrML · BPEL · Security · AOP · RBL

1 Introduction

Services and information flow form the essential assets as well as the key to the growth and success for all businesses today. However, we need to ensure that these resources held on the system are secure especially when they are shared between several participants in a business interaction such as the case of a BPEL process that combines several web services.

Seeing that the use of the internet continues to grow, web services are assuming greater importance as the public face of business not only by making business applications accessible through the internet but also by providing a complete structure for a business process allowing different applications to collaborate together in order to deliver their services. Web services naturally have greater security risks than traditional applications; therefore, it is important for web services to be fully protected against these risks [11]. Many people using web services expect their confidential information to stay secure. Furthermore, with this high level of dependency upon the services provided by web services, it is essential that they are protected from any type of security breaches and threats. Hence, it is important to develop tools that meet the need of the enforcement of security requirements such as license verification, authenticity, authorization, and confidentiality into web services.

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

Electronic supplementary material The online version of this article (doi:10.1007/s11761-013-0127-5) contains supplementary material, which is available to authorized users.

H. Tout · A. Mourad (✉)
Department of Computer Science and Mathematics,
Lebanese American University, Beirut, Lebanon
e-mail: azzam.mourad@lau.edu.lb

H. Otrok
Department of ECE, Khalifa University of Science, Technology & Research, Abu Dhabi, UAE

Usually, security requirements are embedded statically into the design/code of the web services. Hence, a remarkable problem arises. Using such mechanism, any modification in the security measures requires an access to the design/-code and an update of this latter. In this context, several approaches have been proposed. They are more dynamic since they are based on policy languages. The following are instances of these languages: WS-Security [2], SAML [14], XACML [15], and XrML [6]. The WS-Security supports message integrity and confidentiality offering the ability to exchange signed encrypted messages in a web services environment. Security Assertion Markup Language (SAML) and XACML provide mechanisms for authentication and authorization in a similar environment. XrML is a licensing language that defines rights, associated with digital content and web services, within grants. These approaches can be useful when the problem pertains on updating a single web service.

Nevertheless, the problem remains at the BPEL process level handling the web service compositions. Therefore, it seems interesting to take advantage of the aforementioned languages to be applied at the process level. In our approach, we opted to adopt XrML that apt to meet the licensing requirements of the composite web services within the BPEL process. As a matter of fact, the current BPEL suffers from certain shortcomings: for instance, lack of modularity for modeling crosscutting concerns and non-support of runtime modification of the composite web services. In other words, the update, addition, or even removal of any partner web service may require stopping the entire process, making the needed updates into the web service(s), and then redeploying it. This leads to make all services unavailable during the update process.

Moreover, by invoking the BPEL process, all the involved partners (e.g., web services) get invoked. Hence, the security measures such as license verification will be executed at each web service call, that is, each invoke. This overhead of security measures checking will obviously affect the performance of the BPEL process.

We propose in this paper a new approach that aims to cope with the aforementioned problems. It takes advantage from the current XrML paradigm by adopting it into BPEL to cater for the needs of web services licensing. Using XrML, entities will be able to define grants within licenses and associate them with the offered activities. Our proposal exploits also the AOP concept that is based on aspects to provide modularity for modeling crosscutting concerns. It also offers the ability to generate automatically some AspectBPEL security aspects depending on the issued licenses. Built on top of the current AOP technologies, AspectBPEL (a language that we have developed in previous work) allows to describe and specify security BPEL aspects. These aspects not only provide modularity to describe separately security measures but also identify some selective join points where these measures will get dynamically activated. In other words, rather than being executed at each invoke, security features will be activated only at the stated join points in the BPEL process. Our approach provides as well a weaver that integrates the generated AspectBPEL aspects into the BPEL process at runtime.

The following are the main contributions of our approach:

1. Adopting XrML in BPEL which allows service providers to link their services with security licenses.
2. Generating automatically the BPEL aspects depending on the XrML licenses.
3. Separating between crosscutting concerns of the composed web services.
4. Enhancing the performance of the web service compositions by developing security and business logic into separate modules and centralizing the security at the BPEL level.
5. Offering an easy way to include and update the non-functional requirements such as security licenses into a BPEL process, at runtime and without affecting its business logic.
6. Allowing a real-time license validation at the BPEL process level at any moment during its running time.

To validate the utility and the flexibility of our approach, we have developed an inventory control system (ICS) that is composed of several web services. We have also elaborated a role-based license (RBL) model for the ICS and developed the web services to implement such system where security features will be enforced. Then, a BPEL process that handles the orchestration between these web services has been developed. Afterward, using our parser, we have generated the AspectBPEL aspects. The latter contains a call for the elaborated XrML-License checker that provides the ICS with the grants validation property depending on the XrML licenses. Finally, in the developed framework, the generated aspects have been dynamically weaved into the BPEL process at runtime using the AspectBPEL weaver tool that we have developed in a previous work. Case studies as well as experimental results are also presented to defend our proposal.

The remainder of this paper is organized as follows. In Sect. 2, we discuss the related work. Section 3 is devoted to the description of the inventory control system architecture. In Sect. 4, we present the RBL-ICS model of the inventory control system. Section 5 is elaborated to illustrate the proposed approach. In Sect. 6, we present the elaborated framework. In Sect. 7, we illustrate the implementation of our proposal in a case study. In Sect. 8, we present the results of the performance analysis that we did. Section 9 concludes the paper.

2 Related work

Recently, web services security has gained a lot of attention especially in the area of research. Standards and research papers have been proposed aiming to provide policies and techniques that enforce web services security. In this context, we explore the current standards for security policy description, their advantages, and limitations as well. We illustrate also different work that is related to the specification of security policies, based on the licensing concept and apt to be applied in the web services environment. Finally, we discuss the objectives, advantages, and shortcomings of the current initiatives related to the adoption of AOP in the same environment.

SAML [14] is a product of the Security Services Technical Committee of OASIS. It is an XML-based standard for communicating authentication and authorization data. It defines how to specify security credentials, which are represented as assertions. SAML can be used to manage secure sessions between organizations and can leverage several mechanisms such as basic password authentication, SSL and X. 509 certificates, etc. A security token is delivered to the requester after successful authentication. This security token allows granting certain permissions to the requester.

Proposed by IBM, Microsoft, and Verisign, WS-Security [2] standard seeks to embed security within the SOAP messages. It addresses authentication, signatures, and encryption concerns. WS-Security describes how to exchange security tokens for authentication and authorization of SOAP messages. It is also a means for using XML to encrypt and digitally sign SOAP messages.

WS-XACML [15] or Web Service eXtensible Access Control Markup Language is an XML-based language to specify and exchange access control policies. It is offered by OASIS and designed to define authorization policies for principals using XML.

X-RBAC is an XML-based RBAC policy specification framework for enforcing access control in dynamic XML-based web services. It is proposed by Bhatti et al. [3]. The specification uses representations of users, roles, and permissions. The two main components of the proposed framework are the XML and the RBAC processors. The XML processor is implemented in Java using Java API for XML Processing (JAXP). Some modules have the duty to get the DOM instance of parsed XML documents and forward them to the RBAC Processor. The RBAC module is responsible for administration and enforcement of the policy according to the supplied policy information.

A web service architecture design for enforcing access control policies has been proposed by Agostino et al. [1]. Their proposal provides also an example of implementation based on the WS-Policy [16,19] as access control language. The main components of the proposed architecture are Policy Administration Point (PAP), Policy Evaluation Point (PEP), and Policy Decision Point (PDP). The PAP module is a policy repository that provides an administrative interface for inserting, updating, and deleting policies. The PEP module realizes the enforcement of the policies returned by the PAP module. The access request is granted if at least one policy is satisfied; the access is denied otherwise. A PDP module is the interface between the service and the enforcer module. It is responsible for taking final access control decisions based on the input from the PEP module.

The eXtensible rights Markup Language (XrML) [6] is an XML-based specification grammar proposed by ContentGuard. It offers the ability to specify and express grants within licenses associated with digital content, services, or any digital resource. It supports also service-centric models such as web services. By providing a standard language that is platform independent, it extends the usefulness of web services for service providers.

RBAC-WS-BPEL, proposed by Paci et al. [17], is a framework that offers the ability to define authorization policies and constraints for WS-BPEL business processes. WS-BPEL is a language for composing web services. To specify authorization policies, the authors used XACML. They introduce the Business Process Constraint Language (BPCL), which can be used to specify authorization constraints. The users are associated with roles as done in role-based access control (RBAC) models.

The aforementioned approaches support our claim about the need to have standard languages for the description and specification of security measures. Although the proposed solutions aim to enforce the web services security, they cause a security measures verification overhead at the web services side. Practically, they target the security features implementation and verification at the web services level. On the other hand, many security features require run-time verification. For instance, license validation which may often be modified and updated. In this context, when adopting one of the aforementioned solutions, another problem appears in compound systems such as BPEL process. Seeing that the latter is composed of several web services, any modification in the license will lead to stop the whole process during the update procedure. Consequently, all the services will become unavailable. Our approach relies on the dynamic injection of AOP aspects (representing the license grants validation) into BPEL processes and centralization of security controls at that level. This allows an easy update of the security measures when needed, without affecting the business logic of the BPEL process. In the sequel, we present the initiatives related to the adoption of AOP in the web services context.

AO4BPEL language has been introduced in [5] in order to improve the modularity and the dynamic adaptation of web

services composition. It extends the process-oriented composition language "BPEL" with an aspect-oriented mechanism in which aspects can be weaved into the composition process at runtime. In AO4BPEL: Join points are mapped to BPEL activities, Xpath language is used to represent point cuts and several types of advices (For instances, before, after and around) are supported. Chiara Di Francescomarino and Paolo Tonella [6] presented Co-AOP (Cooperative aspect oriented programming) for Executable Business Processes. It is an extension to AO4BPEL that aims to make the aspects Reusable. Co-AOP declares the explicit join points (EJPs) for BPEL processes. As an improvement to AO4BPEL, EJPs allow the base code to be aware of the aspect interfaces so that they enhance the aspect reusability by decoupling base code and aspects. They also force the communication to be parameterized between both the based and aspects code. The AO4BPEL offered the BPEL process the ability to modularize the web services composition as well as to adapt to future changes in such composition. However, their approach neglects the security aspect of BPEL and does not address the use of policy specification using standard languages.

3 BPEL process architecture of the inventory control system web services

This section explores the architecture of the inventory control system web services and their corresponding BPEL process.

3.1 Inventory control system overview

Inventory control system, a common web application, is considered as case study to illustrate the feasibility of our approach. Figure 1 depicts the interactions between the user and the inventory control system including the corresponding BPEL process and its composition of web services. As illustrated in the figure, the security features are deployed on the web services side (i.e., not in the BPEL process). This definitely assures that any change in these security features requires a modification in the corresponding web service.

Our inventory control system is consisted of a BPEL process, its composition of mainly four web services, and a graphical user interface that contains in the main page all services offered by our system: for instance, order new products from different suppliers, place an order, get stock reports, search for any sale invoice, generate invoice voucher, and send reminders for due bills. The first offered service allows the user to order new products from suppliers while the second lets the user purchase desired products encompassing the choice of shipment request. The get stock report allows the user to check the availability of the items. The user has also the ability to look for sales transactions and to generate invoice voucher for customers as well as to send reminders

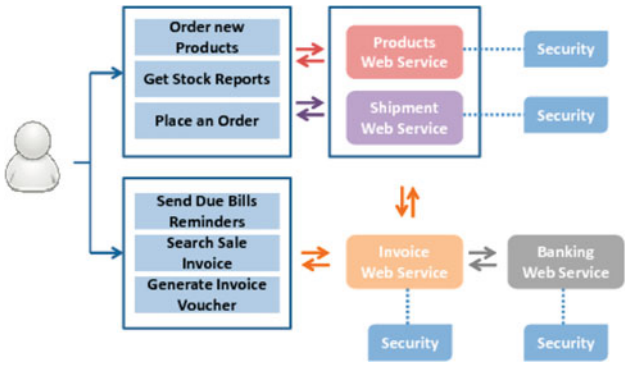

Fig. 1 ICS architecture

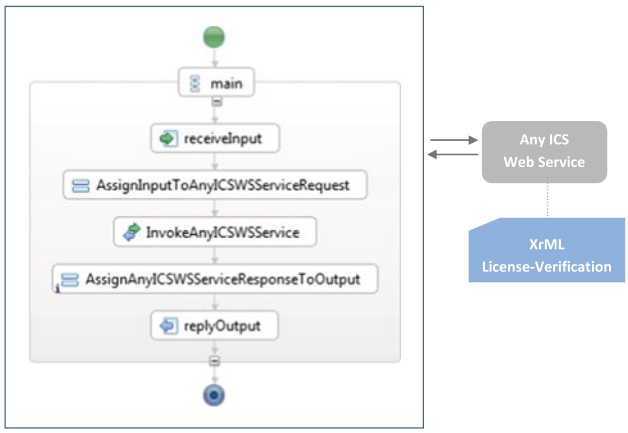

Fig. 2 ICS BPEL process

for due bills. The inventory control system enforces security requirements by invoking security services such as authentication and grants validation, respectively, with the user request to access any of the offered services. Namely, when a user demands one of the inventory control system services, the system fetches the database record where the ID and the password of each user are stored besides its personal information and ensures that he/she is a valid user.

3.2 BPEL process architecture

Figure 2 depicts a part of the architecture of the BPEL process of our inventory control system.

For space restriction, the figure only explores one service invoked from the BPEL process. We called this service *AnyICWSService*. It is offered by one of the ICS Web services. This latter may or may not run with security on the side. Once the user requests one of the services offered by the system, the process gets invoked. Then, this latter assigns the input to *AnyICWSService* Request message. Afterward, it calls the appropriate operation of the requested service and returns the needed information. Finally, the web service Response

message is assigned to the BPEL process Output variable and then forwarded back to the user.

4 RBL-ICS: role-based licensing model for an inventory control system web services

In this section, we introduce the RBL-ICS model of our inventory control system. Initially, we define the RBL-ICS model; subsequently, we present an excerpt of an XrML-based license specification for the inventory control system. We recall that XrML is an XML-based language for license specification.

4.1 RBL-ICS model definitions

This model includes several components: users, roles, grants, role hierarchies, user-role assignment, and role-grant assignment relations. Users are assigned to roles, and roles are assigned to grants. An RBL grant represents the ability to access a certain system service. A user is permitted to execute a service activity if he/she is assigned to a role that has the grant to perform that activity. On the other hand, RBL roles are structured in a hierarchy.

A: is the identifier of an activity (e.g., order new Products).
R: is the set of roles (e.g., Manager, Supervisor, Employee, and Customer).
U: is the set of potential users.
G: is the set of grants, (e.g., Execution of an activity).

4.1.1 Definition 1: RBL-ICS grant

Let ICS be our System. An *RBL-ICS* grant is a tuple (A_i, *Action*), where A_i is the identifier of an activity in ICS and *Action* identifies the type of the action that can be performed on activity A_i. For example, the tuple (*Order new Products*, *execute*) allows the authorized user to *execute* the "Order new Products" service provided by the inventory control system.

4.1.2 Definition 2: RBL-ICS role

An RBL-ICS role r is a set of attribute conditions $r = \{ac_i | ac_i = AttrName_i op AttrValue_i\}$, where $AttrName_i$ identifies a user attribute name, op is a comparison or a set of operators, and $AttrValue_i$ is a value, a set, or a range of attribute values. Note that the roles r and r' might be recognized by the same set of attribute names. However, it is a must that at least one of the values that the attributes of r and r' assume must be different. A user can be assigned to only one role while two users identified by the same attributes with the same values are assigned to the same role since we assume that a set of attribute conditions uniquely identifies a role (as illustrated in Table 1).

Table 1 RBL-ICS role hierarchy

R	
Manager	{Employment=Manager, ID=integer of 9 digits, Password=a string of at most 9 characters}
Employee	{Employment=Employee, ID=integer of 9 digits, Password=a string of at most 9 characters}
Supervisor	{Employment=Supervisor, ID=integer of 9 digits, Password=a string of at most 9 characters}
Customer	{Employment=Customer, ID=integer of 9 digits, Password=a string of at most 9 characters}

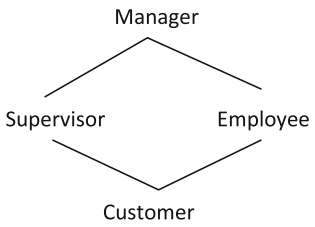

Fig. 3 RBL-ICS role hierarchy

4.1.3 Definition 3: RBL-ICS role hierarchy

Let R be a partially ordered set of roles. A role hierarchy defined over R is the graph of the partial-order relation between the roles in R. If $r, r_0 \in R$ and $r < r_0$, then we say r_0 dominates r. For instance, our inventory control system consists of four different roles. The highest role is the $Manager$ which has access to all the available services. The $Supervisor$ and $Employee$ come next in the hierarchy. They have less access rights than the $Manager$ but more access rights than the $Customer$ members. The role hierarchy of the inventory control system is depicted in Fig. 3.

4.1.4 Definition 4: RBL-ICS user-role assignment relation

Let U be the set of all potential users and R be a partial ordered set of roles. The *RBL-ICS* user assignment relation is the set of attributes $UA = \{(u, r) \in \mathrm{U} \times \mathrm{R} | \forall ac_i = AttrName_i op AttrValue_i \in r, \exists attr_j \in \mathrm{CredSet}\ (u)$[1] $| attr_j = AttrName_i \bigwedge ac_i$ is evaluated to "true" according to the value of $attr_j\}$. As for the online purchase system, the set of roles are R={ *Manager, Employee, Supervisor*, and *Customer*}. Assigning users to roles results in a set of attributes that defines the RBL-inventory control user assignment relation. For example, in our inventory control

[1] Credential Set.

system, the set of attribute conditions for the Manager role is r= { Type= "Manager," ID= a string of 9 characters, Password= a string of at most 9 characters }; thus, a credential set of the user u={Type= "Manager", ID="Emma," Password= "Empass"} will be evaluated as "true" and u is assigned to $Manager$.

4.1.5 Definition 5: RBL-ICS user-grant assignment

Let G be the set of grants of the activity A_1 supported by the system, and RG be the set of grant/role assignments. Thus, the RBL-ICS user-grant assignment relation is the set of attributes UG= {(u,g) ∈ U x G |∃ (u,r) ∈ UA | (r,g) ∈ RG}. For instance, a grant to order new items is assigned to $Manager$ by the RG relation. Thus, a user u can order new items only if he is assigned first to $Manager$.

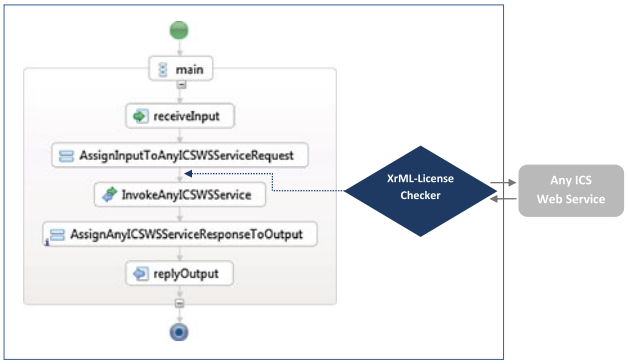

Fig. 4 License checker

5 WS-XrML-AspectBPEL approach

Our approach focuses on enforcing non-functional requirements (e.g., security) into web service compositions based on licensing concept.

In the context of policy specification, several XML-based usage grammar languages such as XACML [15], WS-Security [2], XrML [6], or SAML [14] have been developed for access control, integrity, confidentiality, digital rights management, authentication, and authorization. They show their suitability in scenarios where complex and composed security policies must be organized. They also offer the ability to specify security rules in XML-based documents(s). XrML is a licensing language used for expressing rights. It enables developers to establish the rights and conditions needed to access digital content and web services. On the other hand, AOP provides modularity for modeling crosscutting concerns between web services. Hence, to achieve our objective, we developed a framework based on a synergy between eXtensible rights Markup Language (XrML), aspect-oriented programming (AOP), and web service compositions.

Our proposition is constituted of three phases: (1) XrML-License Checker that validates licenses associated with the offered activities, (2) XrML-AspectBPEL Generator that builds automatically the AspectBPEL aspects according to specified standard description of XrML licenses, and (3) AspectBPEL Weaver that dynamically enforces security features into the BPEL process. The first phase exploits the license verification requirement. The second allows specifying security concerns in separate XrML components and generates automatically their corresponding aspects expressed using our elaborated language AspectBPEL, while the third phase offers a tool for dynamic weaving of the generated aspects in the BPEL process. In the sequel, we illustrate each phase.

5.1 WS-XrML for WS-BPEL

Considering the workflow for digital content and services, it seems that the exchange of rights information between the workflow entities is needed. XrML is considered as the most advanced and mature language to specify rights within licenses. It offers the ability to specify the parties allowed to use the specified resources as well as their rights and the terms under which these rights can be exercised. Its constructs are precise and unambiguous. It exploits the advantages of the XML technology such as flexibility, extensibility, namespaces, aliases, and schemas. Moreover, it supports content-centric models such as e-book, and service-centric models as web services. It is a language that can be used by anyone owning or even distributing digital contents like software applications and services to associate licenses with these assets. Although the use of XrML is limited to digital works and services, it is a comprehensive rights language that provides an advanced syntax to describe both simple and complex business models. Hence, it provides an advantage over other policy languages which have a complex syntax. In this context, we opted to adopt it on the BPEL process model, handling a composition of web services, in order to validate grants offered with the requested services. As illustrated in Fig. 4, one of the contributions offered by our proposal is the decrease of the overhead caused by the security measures checking. In other words, rather than validating the licenses at the web services level, the proposed approach includes a license checker at the BPEL process side. It checks and validates the license grants before the invoke of the requested service.

Listing 1 depicts a simple XrML license. It contains a grant that conveys to a principal the right to use a particular service. Its structure is described as follows:

Listing 1 XrML License Snippet

```
[1] <license>
[2]  <grant>
[3]    <keyHolder>
[4]     <info>
[5]      <dsig:KeyValue>
[6]       <dsig:RSAKeyValue>
[7]        <dsig:Modulus>sdgs9gj...</dsig:
    Modulus>
[8]        <dsig:Exponent>YHj87h24jn...</dsig
    :Exponent>
[9]       </dsig:RSAKeyValue>
[10]     </dsig:KeyValue>
[11]    </info>
[12]   </keyHolder>
[13]   <!--Right-->
[14]   <use/>
[15]   <!--Resource-->
[16]   <serviceReference>
[17]    <wsdl>
[18]     <nonSecureIndirect URI="http://www.
    AnyWS.com/wsdlfile.xml"/>
[19]    </wsdl>
[20]    <service>anyws:WSService</service>
[21]    <portType>anyws:WSPortType
            </portType>
[22]   </serviceReference>
[23]   <!--Condition-->
[24]   <validityInterval>
[25]    <notAfter>2013-12-24T23:59:59</
    notAfter>
[26]   </validityInterval>
[27] </grant>
[28]</license>
```

- <**KeyHolder** >: Identifies the principal of the grant. He is described as the holder of a specific key. (Line 3 to Line 12)
- <**use** >: Represents the right granted to the stated principle. (Line 14)
- <**serviceReference** >: Encapsulates the information necessary to interact with a service. (Line 16 to Line 22)
- <**wsdl** >: Identifies a digital resource that specifies the location of a WSDL definitions element. (Line 17 to Line 19)
- <**service** >: Specifies the name of a particular WSDL service that is described in the WSDL definitions element. (Line 20)
- <**portType** >: Identifies the service's port to which the service reference refers. (Line 21)
- <**validityInterval** >: Represents the condition that must be met before the right can be exercised. (Line 24 to Line 26)

5.2 Dynamic approach for XrML deployment based on AspectBPEL

Considerable attention has been paid recently to address the web services security needs. Usually, web services security requirements are either embedded statically into their design/code, or enforced using policy languages. Either ways, the offered services will become unavailable during the update procedure of security rules or verification strategy. This will constitute a bigger problem in compound systems as of BPEL process. Precisely, any modification in the environment like updating a partner link (e.g., web service) requires process deactivation. In this case, not only the concerned web service activities will be inaccessible but all other web services activities of the BPEL process. Although BPEL allows dynamic binding through endpoint references, it may require stopping the running BPEL process before redeploying it since the updates should be done in the web service code itself. Moreover, BPEL does not support the dynamic adaptation of control flow and data flow in the process. For example, adding new partners, activities, and variables at runtime is not supported. Furthermore, when invoking a BPEL process, security measures get invoked at each web service of the process. This will render the process heavy and affect its performance.

In this context, AOP [8,9,12,13,18] can help to address these issues. It is one of the most prominent paradigms that investigate the use of aspects for the modularization of crosscutting concerns. The integration of non-functional requirements (e.g., security) is performed through the definition of aspects. An aspect is a module that contains pointcut designators specifying sets of join points. Each join point identifies one or a set of flow points in the concerned program which is in our case the BPEL process. Advices specifying how to alter the process behavior and when in the selected join points (i.e., before, after, or around) are activated at the flow points of interest. The aspect is ultimately woven (i.e., integrated) within the application code, at runtime (dynamic weaving) through one of the weaving technologies (e.g., AspectJ [12]). In addition to our experiments, many contributions [4,7,10,20] have proven the usefulness of integrating security features into software using AOP. In this context, we have elaborated in previous work a language called AspectBPEL. Using such technique in association with XrML would enforce the concept of separating concerns of composite web services.

We present in this paper a contribution between XrML and AOP to cope with the aforementioned issues. It is based on implementing security controls independently of the application logic by specifying the security licenses using XrML, then generating automatically the corresponding AspectBPEL aspects and weaving them into the BPEL process using our elaborated framework. The generated aspects offer the ability to activate dynamically the license checker (see description in Sect. 6.1.1) and only at the stated points in the BPEL process.

Some security features require run-time checking. For instance, license that has to be verified at runtime to check if the user has the privileges to perform the requested operation. This goal could be achieved by embedding these features in BPEL aspects and dynamic update can be performed as well. In addition, web service security is generally represented as a specification of policy rules that are written in a specific

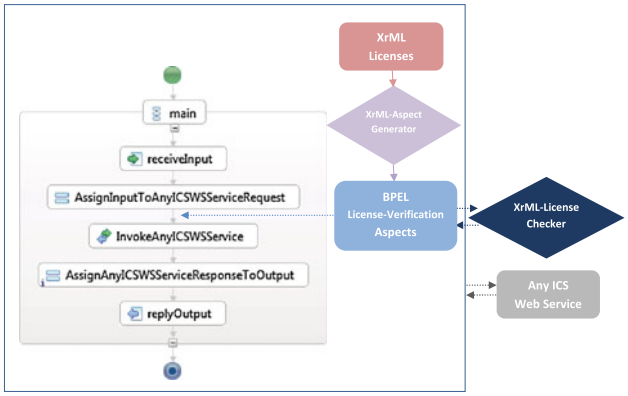

Fig. 5 Approach schema

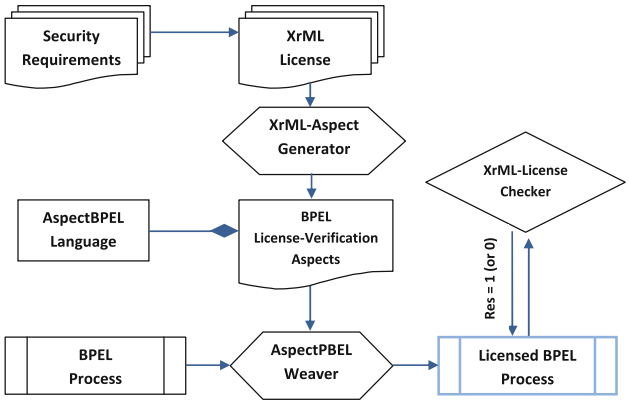

Fig. 6 Framework architecture

language. To enhance the dynamic verifications, it is enough to just link these rules to security aspects where pointcuts and join points are defined. Through these points, security rules can be injected in the BPEL process and could be updated as well, at any time, without the need to modify neither the business logic nor the web services code.

Figure 5 depicts our proposed approach. It illustrates the BPEL process where security features are embedded in AspectBPEL aspects generated automatically by the XrML-AspectBPEL Generator based on the XrML licenses. The comparison between Figs. 2 and 5 shows that the licensing verification feature is no longer part of the web services but rather developed independently within aspects modules, and hence any modification in the security licenses can be reflected in the generated AspectBPEL aspects that are weaved and activated dynamically in the BPEL process. These aspects may contain direct licenses verification to be integrated at some identified join points in the BPEL process, or it may contain an invoke to external web service(s) that handles the verification procedure. Examples of XrML security licenses and AspectBPEL aspects are presented in Sect. 7.

6 WS-XrML-AspectBPEL architecture and implementation

In this section, we describe the framework architecture and implementation. We illustrate also the aforementioned phases of our approach.

6.1 Framework description

Figure 6 illustrates the interaction between the components of the proposed framework. As depicted in the figure, security requirements are specified into separated components and expressed using XrML licenses. Afterward, our XrML-AspectBPEL Generator parses these licenses and generates their corresponding security BPEL aspects based on the proposed AspectBPEL language. The generated aspects are then conveyed to the AspectBPEL weaver with the selected BPEL process in order to produce a licensed BPEL process. This latter communicates with our XrML-License checker in order to validate the license grants at the process level. Based on the XrML-License checker response ("1" or "0"), the licensed BPEL process takes the decision to either grant or deny access to the resource in question.

6.1.1 XrML-License checker

To overcome the issue of license validation measures overhead at the web services level, we have developed an XrML-License checker. Typically, this latter get invoked at the BPEL process level and only at the stated join points of the generated aspects. It is developed in Java and contains a method that starts by parsing the license. Then, it checks whether the assigned role has the right to invoke the requested service, and finally, it returns an appropriate message. Using the XrML-License checker, our proposal provides also dynamism by offering the ability to validate the license grants at the BPEL process runtime without the need to stop its deployment once a license get updated.

6.1.2 XrML-AspectBPEL generator

To generate AspectBPEL aspects from XrML License, we have developed an XrML parser based on the DOM parser for XML, using Java language. The parser gets all the resources as well as their corresponding rights specified in each grant tag in the license. Subsequently, each combination serviceportType of the resource element is matched to an invoke activity in the BPEL process side. Then, a location identifier is generated for each of them and its corresponding behavioral code as well. Each location identifier represents a BPEL activity (i.e., invoke, assign, or any other activity). It identifies a/set of join point(s) where the behavioral code should be applied, whereas this latter contains the code (written in XPath) that will be weaved to the BPEL process before/after or replacing the previously stated location identifier. The list

of location identifiers is wrapped between a BeginAspect statement and an EndAspect statement. The behavioral code contains a call for the XrML-License Checker that handles the license verification. For each grant specified in the license, a loop runs to get the corresponding principal, resource, right, and condition values. The name of the invoke activities of the BPEL process can then be simply deductible from the resultant list of combinations service-portType. Practically, the port type and the operation attributes values of the invoke activity correspond to the portType and the service attributes values, respectively.

The generated aspect is written in AspectBPEL language. It calls dynamically the XrML-License Checker that validates the license grants. It includes also an If condition forming a break point that determines whether the process gets to continue its predefined path, or exits, depending on the checker response. The generated BPEL code encapsulated in the behavioral code is mainly divided into two parts. The first one assigns to the XrML-License Checker, the role corresponding to the login information entered by the user as well as the right and the resource that constitute the invoke activity in the BPEL process. The XrML-License Checker parses the license and returns 1 if the user has the right to access the requested operation. This is done by simply matching the assigned values with the data fetched by the elaborated parser taking into consideration that the specified condition should be satisfied. Otherwise, the returned value is 0, while the second part constituted of the If condition returns an error message to the client in case the checker response is 0 and consequently stops the process. Else, it allows the invoke of the requested service.

6.1.3 AspectBPEL weaver

After elaborating the AspectBPEL language, we have developed its corresponding framework and weaver. This latter allows the integration of the generated AspectBPEL aspect code into the specified BPEL process. It starts by compiling the AspectBPEL aspects, and then performing the needed weaving. First, it fetches the insertion point (i.e., before, after, or replace). Then, it takes the location behavior, which represents the activity type and activity name where the behavioral BPEL code will be inserted. We have integrated a BPEL parser based on the ode.bpel library in the AspectBPEL framework in order to find the defined join points in the BPEL process. Once found, the behavioral code will be inserted in the original code and a message indicating that the weaving was successfully done will appear in the output console.

6.2 Framework implementation

Our framework, which is illustrated in Fig. 7, offers a user-friendly environment to perform a binding between

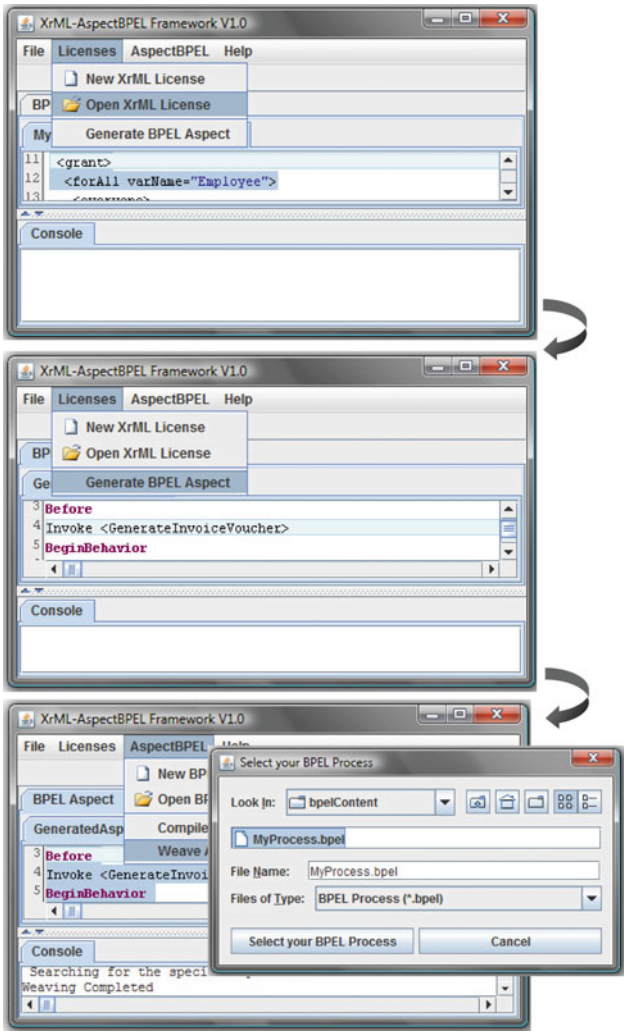

Fig. 7 XrML-AspectBPEL framework

an XrML license and a BPEL process. It allows automatically transforming the XrML license into AspectBPEL aspect and weaving it into the selected process. To perform the hardening process, the user should first select the "Licenses" menu. A new editor pane appears under the "Licenses" panel where the XrML license is developed or loaded. Then, the user has to select the "Generate Aspect" item to convert the XrML code into an AspectBPEL aspect. Once this latter is generated, the user should compile it by simply selecting "Compile Aspect" from the "AspectBPEL" menu. If a message appears indicating that the compilation was successful, the user can move to the final step in order to weave the generated aspect code into the process. He should go to the "AspectBPEL" menu and select the "Weave Aspect" item. On click, a dialog box appears, requesting the user to select to which BPEL process the AspectBPEL aspect code will be weaved. A "Weaving Completed" message in the console appears indicating that the weaving was successfully performed.

7 Case study: dynamic enforcement of the RBL-ICS model in the ICS

This section describes the implementation of the RBL-ICS model illustrating all the procedures as well as the mechanisms presented in our proposed approach for the dynamic enforcement of XrML-license verification feature in the ICS.

7.1 RBL-ICS XrML specification

Listing 2 outlines a synopsis of the XrML-based license for the ICS. This listing includes only the grants (Line 7 to Line 29) offered to the employee. Other grants are omitted here due to space constraints. However, they are set similarly. Each grant contains the role (Line 8 to Line 17) to which the privileges are certified. These roles are structured in a way that the root is denoted by Manager. Followed by 2 sub-roles, Supervisor and Employee, the Manager is granted all the rights to be performed on the associated resources. The Supervisor and the Employee roles share the Customer sub-role and are assigned, respectively, to varName="Supervisor" and varName="Employee," While the Customer is assigned to varName="Customer." Each grant encloses the right (Line 19), the associated resource (Line 22 to Line 24), and the condition (Line 25 to Line 28) that must be met before the right can be exercised, designated to each role. For instance, the varName="Supervisor" includes the generating invoice that is valid during certain time interval.

7.2 AspectBPEL aspects realizing the RBL-ICS

In what follows, we describe the generated BPEL security aspect realizing the aforementioned XrML license of the RBL-ICS model. Please note that the syntax and constructs of the aspects are specified in AspectBPEL language. At the beginning, our ApsectBPEL framework compiles the aspects. It returns a failure message in case any error was found. Otherwise, a message indicating that the compilation was successful is returned. Afterward, it weaves the generated aspect to the selected BPEL process. Our approach has been tested, and we were successfully able to verify licenses associated with the offered services at the BPEL process level during its running time.

Listings 3 and 4 (for space limitations, AspectBPEL aspect code is divided into 2 parts, (a) and (b)) illustrate an excerpt of the generated AspectBPEL aspect for the verification of license associated with ICS services and realizing the XrML license of Listing 2. For space restriction, the listings depict only the AspectBPEL code that validates the grants associated with the GenerateInvoiceVoucher service. The others are set in similar way. As described in the aforementioned RBL-ICS model (described in Sect. 4), each user is assigned a role and its corresponding grant(s). The RBL model reflected by the XrML license has been generated as AspectBPEL aspect code. This latter integrates the grants verification before (Line 3) the execution of any invoke activity whose port type and operation were specified in the XrML license (Line 4). It begins by assigning the role, right, and resource subjects of the user request (Line 7 to Line 34). Subsequently, the XrML-License Checker parses the license and checks if access should be granted or denied. As illustrated in the listing, the response of the checker is wrapped inside an If condition (Line 38 to Line 54) that checks the returned value. If it is equal to 1, the BPEL process continues its execution by invoking the appropriate web service for the requested service. Otherwise, it returns an error message (Line 40 to Line 55) and the BPEL process exits.

Listing 2 Excerpt of XrML-based license for ICS

```
[1]<license licenseId="ICSLicense" xmlns="
    http://www.xrml.org/schema/2001/11/
    xrml2core" ...>
[2] <title>ICS License</title>
[3] <!--Grants set for the Manager-->
[4] <!--Grants set for the Supervisor-->
[5] <!--Grants set for the Customer-->
[6]  ...
[7] <grant>
[8]  <forAll varName="Employee">
[9]   <everyone>
[10]     <trustedIssuer><dsig:KeyInfo><dsig:
    KeyValue>
[11]       <dsig:RSAKeyValue>
[12]        <dsig:Modulus>sdgs9gj...</dsig:
    Modulus>
[13]        <dsig:Exponent>YHj87h24jn...</
    dsig:Exponent>
[14]       </dsig:RSAKeyValue>
[15]     </dsig:KeyValue></dsig:KeyInfo></
    trustedIssuer>
[16]   </everyone>
[17]  </forAll>
[18]  <principal varRef = "Employee"/>
[19]  <use/> <!--Right-->
[20]  <serviceReference> <!--Resource-->
[21]   <wsdl> <nonSecureIndirect URI="http://
    localhost:8080/ode/processes/ICS/
    ICSwsdlfile.xml"/> </wsdl>
[22]   <service>GenerateInvoiceVoucher</
    service>
[23]   <portType>ns:ICS</portType>
[24]  </serviceReference>
[25]  <validityInterval> <!--Condition-->
[26]   <notBefore>2011-11-01T00:00:00</
    notBefore>
[27]   <notAfter>2013-02-01T00:00:00</
    notAfter>
[28]  </validityInterval>
[29] </grant>
[30] <issuer>
[31]  <dsig:Signature><dsig:KeyInfo><dsig:
    KeyValue>
[32]     <dsig:RSAKeyValue>
[33]      <dsig:Modulus>sdgs9gj?</dsig:
    Modulus>
[34]      <dsig:Exponent>YHj87h24jn?</dsig:
    Exponent>
[35]     </dsig:RSAKeyValue>
[36]  </dsig:KeyValue></dsig:KeyInfo></dsig:
    Signature>
[37]  <details>
[38]   <timeOfIssue>2011-10-26T16:20:01</
    timeOfIssue>
[39]  </details>
[40] </issuer>
[41]</license>
```

Listing 3 Excerpt of Generated AspectBPEL Aspect for License Grants Validation (a)

```
[1]Aspect GeneratedGrantsValidationAspect
[2]BeginAspect
[3]Before
[4]Invoke <GenerateInvoiceVoucher>
[5]BeginBehavior
[6] <bpel:sequence>
[7]  <bpel:assign validate="no" name="
    MyAssign">
[8]   <bpel:copy> <bpel:from> <bpel:literal>
[9]    <impl:checkAccess xmlns:impl="http://
    t320.open.ac.uk">
[10]    <impl:Principal>impl:Principal</impl:
    Principal>
[11]    <impl:Action>impl:Action</impl:Action
    >
[12]    <impl:Resource>impl:Resource</impl:
    Resource>
[13]   </impl:checkAccess>
[14]  </bpel:literal> </bpel:from>
[15]  <bpel:to variable="PartnerLinkRequest"
    part="parameters">
[16]  </bpel:to>
[17] </bpel:copy>
[18] <bpel:copy>
[19]  <bpel:from part="payload" variable="
    input"> <bpel:query> <![CDATA[tns:input
    ]]></bpel:query> </bpel:from>
[20]  <bpel:to part="parameters" variable="
    PartnerLinkRequest"> <bpel:query> <![
    CDATA[ns:Principal]]> </bpel:query> </
    bpel:to>
[21] </bpel:copy>
[22] <bpel:copy>
[23]  <bpel:from> <![CDATA["GenerateInvoice
    "]]> </bpel:from><bpel:to part="
    parameters" variable="PartnerLinkRequest
    ">
[24]   <bpel:query> <![CDATA[ns:Right]]> </
    bpel:query> </bpel:to>
[25] </bpel:copy>
[26] <bpel:copy>
[27]  <bpel:from> <![CDATA["ICS"]]> </bpel:
    from> <bpel:to part="parameters" variable
    ="PartnerLinkRequest">
[28]   <bpel:query> <![CDATA[ns:Service]]> </
    bpel:query> </bpel:to>
[29] </bpel:copy>
[30] <bpel:copy>
```

7.3 Discussion and experimental results

Our approach produces a licensed BPEL process as illustrated in Fig. 9 (due to the big size of the BPEL process, we show only one service, which we call AnyICSWSService). To reach that goal, it starts first by generating automatically the license grants validation aspect in Listing 3 from the XrML license in Listing 2, and then weaving them into the BPEL process of the inventory control system presented in Fig. 2. The resulted BPEL process not only provides dynamic license verification feature for the inventory control system but also decreases the overhead of security measures verification at the web services level. The licensed BPEL process begins by receiving the client role formulated from his login information. It assigns the user's role besides the requested service to the XrML-License checker. Once this latter gets called, it parses the license and checks whether the client is granted to invoke the requested service. If the access is granted, the process proceeds and invokes the inventory control system web service AnyICSWS and returns the user requested service. Otherwise, the process assigns the web service response message to the BPEL output variable, forward it back to the user, and stops.

After performing an extensive testing, we were able to demonstrate the utility and the feasibility of our proposition. First, we got the ability to successfully integrate the RBL-ICS security features in the corresponding original BPEL process code. Then, we applied some modifications on the license which have been effectively and dynamically reflected in its corresponding generated aspects. Finally, we were capable to ensure that the original functionalities of the ICS have not been altered after the modification has been applied dynamically onto the BPEL process.

Listing 4 Excerpt of Generated AspectBPEL Aspect for License Grants Validation (b)

```
[31]    <bpel:from> <![CDATA["ICS"]]> </bpel:
    from> <bpel:to part="parameters" variable
    ="PartnerLinkRequest">
[32]     <bpel:query>
<![CDATA[ns:PortType]]>
</bpel:query> </bpel:to>
[33]   </bpel:copy>
[34]  </bpel:assign>
[35]  <bpel:invoke name="MyInvoke"
    partnerLink="PartnerLink" operation="
    checkAccess" portType="ns:Evaluator"
[36]   inputVariable="PartnerLinkRequest"
    outputVariable="PartnerLinkResponse">
[37] </bpel:invoke>
[38] <bpel:if name="If" xmlns:http="urn:http:
    namespace">
[39]   <bpel:condition>
<![CDATA[$PartnerLinkResponse.parameters=
    "0"]]></bpel:condition>
[40]   <bpel:sequence name="Sequence">
[41]    <bpel:assign validate="no" name="
    AnotherAssign">
[42]     <bpel:copy>
[43]      <bpel:from> <bpel:literal> <tns:
    MyProcessResponse xmlns:tns="http://
    myprocess.localhost"> <tns:result>tns:
    result</tns:result>
[44]       </tns:MyProcessResponse> </bpel:
    literal>
[45]      </bpel:from> <bpel:to variable="
    output" part="payload"> </bpel:to>
[46]     </bpel:copy> <bpel:copy>
[47]      <bpel:from> <![CDATA["Access Denied
    "]]> </bpel:from> <bpel:to part="payload"
     variable="output"> <bpel:query> <![CDATA
    [tns:result]]>
[48]       </bpel:query>
[49]     </bpel:to>
[50]    </bpel:copy>
[51]   </bpel:assign>
[52]   <bpel:reply name="Reply" partnerLink="
    client" operation="process" variable="
    output"></bpel:reply>
[53]  </bpel:sequence>
[54] </bpel:if>
[55] </bpel:sequence>
[56]EndBehavior
[57]EndAspect
```

8 Performance analysis

This section summarizes the results of performance analysis conducted on the BPEL process running in three different scenarios: A BPEL process without license verification (LV), a BPEL process with LV (license verification) at the web services level, and a BPEL process that runs with AspectBPEL aspect including a call to our XrML-License checker. Practically, in the first scenario, anyone can invoke the BPEL process activities since it does not contain license grants validation. In the second one, the license checking property is integrated in the web services code, while in the third one, this property is centralized at the BPEL level that calls our XrML-License checker before invoking the requested service.

Table 2 shows how the size of the .bpel and .wsdl files of the inventory control system varies from one scenario to another. It shows that the size of the process (i.e., number of lines of the process code) with AspectBPEL license verification is the biggest, ascribe that to the AspectBPEL aspect code integrated in it. However, in the rest of this analysis, it will be obvious that this overload does not have a negative side effect on the BPEL performance. The first chart of Fig. 8 shows the variation in the execution time (time taken to execute/invoke the services) of the BPEL process in the aforementioned scenarios with respect to the number of invokes it contains. This latter refers to the number of services that are called in the ICS BPEL process. The stated values were measured using the Visual Studio profiling tool. As any profiler, this latter allows to measure the behavior of a program as it executes (for instance, the duration of function calls). As the results vary from one call to another, we decided to calculate their average reducing the margin of deviation. As shown in this chart, the non-licensed BPEL process (BPEL without LV) enclosing 10 invokes runs in an average of 7,776 ms and reaches 68,875 ms for 100 invoke activities. Conversely, the BPEL process of 10 invokes with the license verification on the web services side (BPEL with WS-LV) takes up to 9393 ms and comes to 85,188 ms when enclosing 100 invoke activities. Nevertheless, the BPEL process with AspectBPEL-LV runs in an average of 9,136 ms for 10 web services invoke and takes up to 71,562 ms for 100 web services invoke. In the second scenario, the license is embedded within the business logic of the web service; this will lead to invoke it at each service call (even if the service is not included in the license of the web service). However, in the last scenario, we can specify where to integrate the security (license verification) code at the BPEL level. So the code will be inserted and called only at the stated join point decreasing the overhead of verification at the web services level. Therefore, the execution time in the third case (overhead at the web services level was decreased) is significantly less than the one in the second case. The chart reflects this by showing that the line illustrating the execution time of the BPEL process with AspectBPEL license verification is almost in congruence with the one of the non-licensed process. This proves also that the overload of the AspectBPEL code is not affecting the performance of the BPEL process.

Table 2 Size of the ICSProcess BPEL and WSDL File

	Without LV	WebService LV	AspectBPEL-LV
ICSProcess.bpel	170 Lines	259 Lines	268 Lines
ICSProcess.wsdl	69 Lines	85 Lines	74 Lines

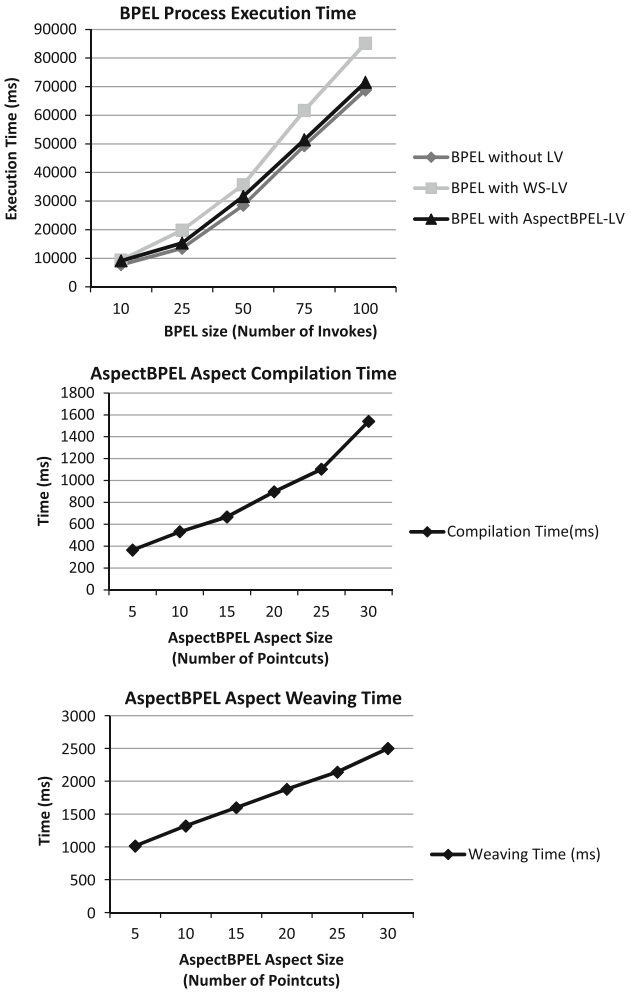

Fig. 8 Performance analysis

The performance analysis could not be accomplished without measuring the compilation and the weaving time of the AspectBPEL-LV aspect (i.e., time taken to compile and integrate the AspectBPEL aspect code in the process). Therefore, using the Eclipse Test and Performance Tools Platform (TPTP), we have performed this test, and the results are shown in the second and the third chart of Fig. 8. The compilation and the weaving time are in ms

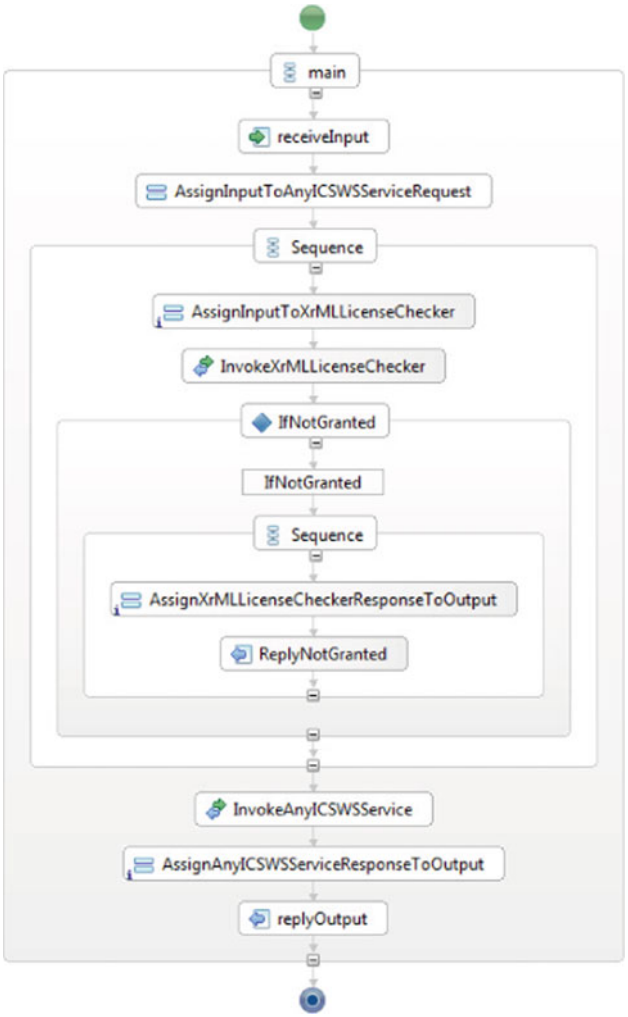

Fig. 9 ICS licensed BPEL process

while the aspect size is measured in terms of the number of pointcuts presented in it. As shown in these charts, it takes only 1540 ms to compile an AspectBPEL aspect of 30 pointcuts and just 2,500 ms to weave it in the BPEL process.

All above results not only reflect how fast the compilation and the weaving functions can be but also they prove that a BPEL process with AspectBPEL license verification still by far faster than a BPEL process with license verification implemented on the web services side. For example, assuming that we have a BPEL process enclosing 25 invoke activities that needs an AspectBPEL aspect of 25 pointcuts. That means, different security behaviors (i.e., license verification) for each invoke activity. The total running time of this process is equal to the compilation time plus the weaving time of the AspectBPEL aspect plus the runtime of the process. In this example, it is = 1103 + 2140 + 15356 = 18,599 ms $<$19,833 ms which is the running time of the BPEL having the license verification property implemented at the web services level.

9 Conclusion

We presented in this paper a new approach that is based on a synergy between the licensing concept offered by XrML, aspect-oriented programming (AOP), and BPEL process of composite web services. It exploits the XrML paradigm and the aspect concept offered by AOP to cater for the needs of the web services composition into a BPEL process. It offers the ability to define grants within licenses, associate them with the offered activities, and transform them automatically into AspectBPEL aspects. These aspects offer a dynamic procedure to include and update the non-functional requirements such as license grants validation into a BPEL process, at runtime, and without affecting its business logic. They allow also a real-time license validation at the BPEL process level. Moreover, through the modularity offered by the aspect concept, business work flow and security concerns can be developed separately as different components which allow the integration and the modification of the web services composition of the BPEL process at run time without the necessity to stop the process.

References

1. Ardagna CA, Damiani E, De Capitani di Vimercati S, Samarati P (2006) A web service architecture for enforcing access control policies. Electron Notes Theor Comput Sci 142:47–62
2. Atkinson B et al. Web services security (WS-Security). http://www.oasis-open.org/committees/tc_home.php?wg_abbrev=wss
3. Bhatti R, Joshi J, Bertino E, Ghafoor A (2003) Access control in dynamic XML-based web-services with X-RBAC. In: Proceedings of the international conference on web services (ICWS03), pp 243–249
4. Bodkin R (2004) Enterprise security aspects. In: Proceedings of the AOSD 04 workshop on AOSD technology for application-level security (AOSD04:AOSDSEC)
5. Charfi A, Mezini M (2004) Aspect-oriented web service composition with AO4BPEL. In ECOWS04
6. ContentGuard. XrML The digital rights language for trusted content and services. http://www.xrml.org/
7. DeWin B (2004) Engineering application level security through aspect oriented software development. PhD thesis, Katholieke Universiteit Leuven
8. Evermann J (2007) A meta-level specification and profile for AspectJ in UML. J Object Technol 6(7):27–49
9. Fuentes L, Sanchez P (2006) Elaborating UML 2.0 profiles for AO design. In: Proceedings of the international workshop on aspect-oriented modeling
10. Huang M, Wang C, Zhang L (2004) Toward a reusable and generic security aspect library. In: Proceedings of the AOSD 04 workshop on AOSD technology for application level security (AOSD04:AOSDSEC)
11. Ken North Computing. XML and web services: message processing vulnerabilities. http://www.webservicessummit.com/Articles/MessagingThreats.htm
12. Kiczales G, Hilsdale E, Hugunin J, Kersten M, Palm J, Griswold WG (2001) An overview of AspectJ. In: Proceedings of the 15th european conference on object-oriented programming (ECOOP01), pp 327–353, London, UK. Springer

13. Kiczales G, Lamping J, Menhdhekar A, Maeda Ch, Lopes C, Loingtier J-M, Irwin J (1997) Aspect-oriented programming. In: Akÿsit M, Matsuoka S (eds) In: Proceedings european conference on object-oriented programming, vol. 1241, pp. 220–242. Springer, Berlin
14. Lockhart B et al. OASIS security services TC (SAML). http://www.oasis-open.org/committees/tc_home.php?wg_abbrev=security
15. Moses T: OASIS eXtensible access control markup language(XACML), OASIS standard 2.0. http://www.oasis-open.org/committees/xacml/
16. Nolan P (2004) Understand WS-Policy processing. IBM Corporation, Technical report
17. Paci F, Bertino E, Crampton J (2008) An access-control framework for WS-BPEL. Int J Web Serv Res 5(3):20–43
18. Pavlich-Mariscal J, Michel L, Demurjian S (2007) Enhancing UML to model custom security aspects. In: Proceedings of the 11th international workshop on aspect-oriented modeling AOM@AOSD07
19. Schlimmer J (2004) Web services policy framework (WS-Policy). http://www-128.ibm.com/developerworks/webservices/library/specification/ws-polfram/
20. Shah V (2003) An aspect-oriented security assurance solution, Technical Report AFRL-IF-RS-TR-2003-254, Cigital Labs
21. Tonella P, Di Francescomarino C (2009) Cooperative aspect oriented programming for executable business processes. In: Proceedings of The the 2009 ICSE workshop on principles of engineering service oriented systems., Vancouver, Canada