Security policy composition and Rollback Recovery for Composite Web Services

Author: P.Karthi, P.Ganesh Assistant Professor, Department of Computer Technology
SNMV College of Arts & Science, Malumachampatti, Coimbatore -50, India
P.Ganesh Assistant Professor,
Department of Computer Technology, Ayya Nadar Janaki Ammal College
E-mail: karthi86ucan@gmail.com

ABSTRACT- Service Oriented Architectures (SOA) promise a flexible approach to utilize distributed capabilities that may be located in independent trust domains. In this project we propose a service-oriented reliability model that dynamically calculates the reliability of composite web services with rollback recovery based on the real-time reliabilities of the atomic web services of the composition. Securing an SOA application is an important non functional requirement. However, specifying a security policy for a composite service is not easy because the policy should be consistent with the policies of the external services invoked in the composite process. Our model is a hybrid reliability model based on both path-based and state-based models. Many reliability models assume that failure or error arrival times are exponentially distributed. This is inappropriate for web services as error arrival times are dependent on the operating state including workload of servers where the web service resides. In real-world applications, where web services could contain quite a large number of atomic services, the calculus as well as the computing complexity increases greatly. We evaluate and classify of different SOA-platforms and security frameworks regarding secure cross-organizational service invocation. We propose to evaluate the performance implications of our security infrastructure to the overall service response time. We also propose a method for composing and validating policies in composite services according to rules that depend on the variable assignments and their properties. Our contribution is defining the process-independent policy composition rules and providing a method for semi automatically creating a security policy of the composite service.

Keywords

SOA application, Composite process, QoS of web services, Rollback recovery, WSDL document, BPEL process, Recovery block technique, Cross-organizational service

1. INTRODUCTION

SOA is convenient for satisfying functional requirements, but it is more difficult to satisfy the non functional requirements such as security. The security requirements are specified as security policies for the composite service, but there is no clear way to define the policies for the composite service. Currently, a developer needs to define the composite policies by hand by referring to the policies of the invoked services in the composite process. However, it is very hard to complete a policy composition without any inconsistencies, because the process definitions and security policies are complex and it is not clear how to compose policies to maintain consistency. In the next generation of systems, web services represent the most important computing paradigms for configuring applications. A lot of work has appeared on developing the middle-ware framework for these web services including the messaging structure, the typical composition of nodes, workflow architectures, orchestration, and choreography of these services. The service composition and selection are central activities in service-oriented computing, and the Quality of

Service QoS of a Service-Oriented Architecture SOA plays a key role to appropriately drive these activities. A key issue for the QoS analysis of SOA is parameter estimation. The properties of basic services are not easily made available from service providers. For the domain of secured services we propose a novel approach to automated composition of services based on their security policies. Given a community of services and a goal service, we reduce the problem of composing the goal from services in the community to a security problem where an intruder should intercept and redirect messages from both the community services and the goal in such a way that the goal service reaches its final state, defined as an insecure one.

The main objective is to minimize the maximum time that may be spent in saving the states of the system. The main objective is to develop service-Oriented reliability model enhance the QoS of web services' compositions. Creating a consistent composite policy from atomic policies applied to atomic services that consist of a composite service.

We also address the problem of checking that the composed service satisfies some security properties. For the validation of the synthetised service we can employ directly our cryptographic protocol validation tools. In a service federation, however, there might exist multiple models with different semantics or expressiveness for each federation member. In addition, access control decisions in classical models are based on identities and permissions assigned to them.

1.1 Service Level Agreements

To establish a federation with a new business partner, the manufacturer and the new supplier have to agree on IT-level and business process-level service parameters. This agreement affects the access control decision that is also based on environment attributes, such as time and access statistics.

1.2 Dynamic Adaptation of Security Policies

Once a federation is established, continuous changes relating to permissions assigned within a partner's domain might be necessary. For example changing personal, system evolution, and administrative, or ad-hoc delegation of access permissions might be a reason, but this information must not be mediated to federation members in order to prevent flooding and revocation of authorization information. We analyse existing security frameworks with respect to access control for cross-organisational composite web services. Although, these frameworks and platforms focus on entirely different security and SOA aspects, they can be categorized in different groups based on their application to protect a service in a federated environment. We propose a classification depending on the distribution of authentication and authorization information. For each category we present existing example frameworks along with a short description. Web service composition is about the combination of web services from different providers in order to create a more sophisticated, value-added web service. Most proposed composition languages follow the workflow paradigm, whereby the composition is defined as a workflow process that determines which web services participate in the composition.

BPEL allows the specification of interactions among the web services that participate in the composition according to various control-flow patterns. It provides three activities for web service interaction: <invoke> for invoking an operation on a partner web service, <reply> for sending a response to a client, and <receive> for blocking until a client request is received. These activities produce a message-based interaction between the composite web service and its partners, laying down the functional logic of a composite web service.

2. RELATED WORK

Architecting service-oriented systems is a complex design activity. It involves making trade-offs among a number of interdependent design decisions, which are drawn from a range of concerns by various software stakeholders. In order to achieve effective and efficient SOC design we believe a careful study of architectural styles that can form the reference architecture is important. Hence, this paper provides a study of architectural styles for the reference architecture of SOC-based software systems. The author proposes a classification scheme for the architecture styles. These architectural styles are extracted from existing research projects and industry practices based on our classification scheme. For all those identified styles, author presents an evolution trend driven by engineering principles for Internet-scale systems [1]. A recent survey on Web services adoption, for example, shows that quality requirements such as system security, scalability, reliability, flexibility, and performance have become the most important criteria for a company to choose Web services solutions.

Web services represent an alternative basis for the rapid development of application systems. Much of the research on composition and orchestration of web services centers around functional sufficiency and performance. Non-functional characteristics like reliability and security play an important role in the selection of web services by system architects. This paper provides a basis for measuring reliability of an application system that is assembled using web services. Assembling an application using web services requires that the underlying business process be clearly specified, appropriate web services be selected for each task, and the set of selected services integrated into a cohesive application. Several factors need to be considered when selecting appropriate web services to support a specific task. At the outset, the requirements of the task must be met by the functionality provided by the web service [2]

The author observed that both the composite and constituent Web services often constrain the sequences of invoking their operations and, therefore, author proposes using a finite state machine to model the permitted invocation sequences of Web service operations. We assign each state of execution an aggregated reliability to measure the probability that the given state will lead to successful execution in the context where each Web service may fail with some probability. We show that the computation of aggregated reliabilities is equivalent to eigenvector computation and adopt the power method to efficiently derive aggregated reliabilities. In orchestrating a composite Web service, the author proposes two strategies to select Web services that are likely to successfully complete the execution of a given sequence of operations [3].

The ability to automatically compose security policies created by multiple organizations is fundamental to the development of scalable security systems. The diversity of policies leads to conflicts and the need to resolve priorities between rules. In this paper author explore the concept of defeasible policy composition, wherein policies are represented in defeasible logic and composition is based on rules for non-monotonic inference. This enables policy writers to assert rules tentatively; when policies are composed the policy with the firmest position takes precedence. In addition, the structure of our policies allows for composition to occur using a single operator; this allows for entirely automated composition [4].

Automatic composition of web services is a challenging task. Many works have considered simplified automata models that abstract away from the structure of messages exchanged by the services. For the domain of secured services (using e.g. digital signing or time stamping) the author proposes a novel approach to automated composition of services based on their security policies. Given a community of services and a goal service, we reduce the problem of composing the goal from services in the community to a security problem where an intruder should intercept and redirect messages from the service community and the goal in such a way that the goal service reaches its final state, considered as insecure.

The approach amounts to collecting the constraints on messages, parameters and control flow from the components services and the goal service requirements. A constraint solver checks the feasibility of the composition, possibly adapting the message structure, while preserving the semantics, and displays the service composition as a message sequence chart. Moreover the resulting composed service can be verified automatically for ensuring that it cannot be subject to active attacks from intruders [5].

After the modelling stage, the fault tolerance and execution error at the web service level were introduced by using an adaption of the recovery block technique. The broker is supposed to have a built-in acceptance testing mechanism that examines the results feedback from a particular web service. If the returned results (the feedback results) can be deemed acceptable, it proceeds with the computation to the next web service. If they are considered to be unacceptable, they can involve rollback and another component web service to carry out the same function previously required from the faulty component web service. This can be done by saving the states and creating checkpoints [6].

Although a broker with such an acceptance testing mechanism increases the trust in the composite web services, it still constitutes a single point of failure. A main issue for the QoS analysis of SOA is the parameter estimation of the atomic web services and their combination. Concerning the atomic web services, we study in this section, many reliability models assume that the failure or error arrival times are exponentially distributed. This is inappropriate for web services as the error arrival times will be dependent on the operating state including workload of the server where the web service resides. Previous work [7] indicates that the probability of errors occurring in computer system is higher where the workload is higher [8].

To improve the overall reliability of composite web service, one can use redundancy [9] and [10]. Here, by redundancy we mean that the same kind of functionality is available in the web service provided by different service providers [11]. So in this case, we can still provide the same functionality even when some of the web services are not available due to a failure or other reasons.

Also, the redundancy of web services often has a high diversity. Web services delivered by different service providers are often developed individually. The diversity makes it much less likely that the same failure would hit all redundant web services.

The author presented an access control model and techniques for specifying and enforcing access control rules for Web service compositions. They introduced composite roles and principles and specified access control policies using pure-past linear temporal logic. The author proposed an aspect-oriented approach to specify security policies for Web service compositions. They implement a set of aspect in AO4BPEL that is an aspect-oriented extension to BPEL [12].

Finally, Existing method defines the security policy composition rule that is independent of composite processes. Existing method also addressed problem of ours is not concrete definitions of policy consistency. Existing approach is not a static composition of policy representations. Existing model for inserting rollback points in process control type programs. Existing check pointing technique is not suitable for systems where the parameters of the program are time dependent during the mission.

3.0 Description of Proposed Technique

We propose a service-oriented reliability model that dynamically calculates the reliability of composite web services with rollback recovery based on the real-time reliabilities of the atomic web services of the composition. Proposed methods are workflow message consumed and produced by an operation is defined in the WSDL document of the BPEL process. We also propose a method for composing and validating policies in composite services according to rules that depend on the variable assignments and their properties. We also propose introducing fault tolerance for a fault or error at the web service level by using an adaption of the recovery block technique. Proposed method introduces two languages for defining monitoring and recovery and explains how to use them to enrich BPEL processes with self-supervision capabilities. We evaluate and classify of different SOA-platforms and security frameworks regarding secure cross-organizational service invocation. We also evaluate the performance implications of our security infrastructure to the overall service response time.

3.1 Service-oriented reliability model

We develop the new model for the reliability of a web service in order to include the influence of different parameters of the environment. We focus on the modeling and analysis of the reliability attribute in Service-Oriented Architectures, with particular emphasis on two aspects of this problem: (i) the mathematical foundations of reliability modelling of a Service-Oriented Architecture as a function of the reliability characteristics of its basic elements and (ii) the automatization of service composition driven by reliability criteria. The variance concerning the transitions among environment states depends on the load of the server where the web service resides and the type of processing being conducted.

In an SOA environment, services are expected to publish information needed to correctly invoke them over the network. This information, expressed by a suitable language like WSDL, includes the name of the provided operations, and the name and type of their input and output parameters. To support predictive analysis of some QoS attribute like the service reliability, each service must also publish QoS-related information.

3.2 Hybrid reliability model

The hybrid reliability of a service shall be associated to the service description at the time the provider publishes the service on a registry. We focus variable assignments in a composite process, not a process flow itself. Therefore, a number of variable assignments would have an impact on performance of policy composition inference, and complexity or length of composite process might have a big impact on performance in our approach. Here we had an experiment to evaluate how number of assignments affects inference performance.

On the other hand, the hybrid reliability can be estimated as the ratio between the number of service invocations and the number of failures that occur. We point out that, in the case of a composite service, the failures that should be recorded at a composite service site are those generated by the internal segment of the service. Collecting failure statistics about the used external services could not be significant, as at different time instants we could bind to different implementations of the same abstract service and, given the autonomy principle of the SOA environment, we are not generally aware of these changes.

3.3 BPEL processes with self-supervision

BPEL is a workflow-based web service composition language. It specifies the composition as a process, which declares the web services participating in the composition (partners), data containers (variables), and a set of activities with specific patterns of control and data flow. In fig 1.0 shown the building blocks of BPEL processes are activities. There are primitive activities such as <invoke> and <assign> and structured activities such as <sequence> and <flow>. Structured activities manage the order of execution of their enclosed activities. BPEL processes can run on any BPEL-compliant orchestration engine. The engine orchestrates the invocations of the partner web services according to the process specification.

Our approach separates service selection from the process business logic, providing runtime adaptability for the process through dynamic binding, by automatically transforming the process at deployment time. In this way, we achieve transparency for the process developer, who does not have to provide any special constructs, and full compatibility with standard BPEL engines. Our monitoring mechanism supports both complete monitoring and sampling, allowing balancing between monitoring overhead and the amount of gathered service execution statistics that enable self-supervision. The service selection mechanism maintains pools of functionally equivalent services and periodically (re-)assigns service selection probabilities according to the monitored service performance.

We introduce a new approach to transparent BPEL process self-supervision. We provide a flexible process monitoring mechanism that allows balancing the amount of data collected and the monitoring overhead. Furthermore, we present a probabilistic service selection algorithm that takes the monitored service performance into account.

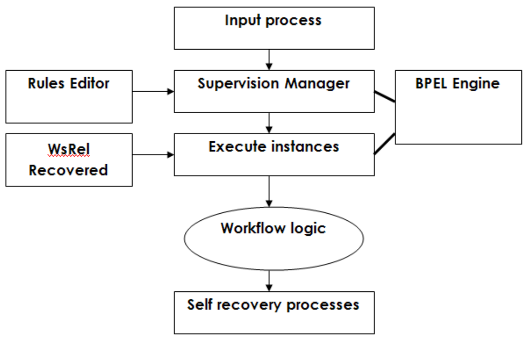


Fig 1.0 BPEL workflow-based Web Service

3.4 The following methods will be used in this paper:

Service type – Unique identifier of a group of functionally equivalent services that can substitute for each other

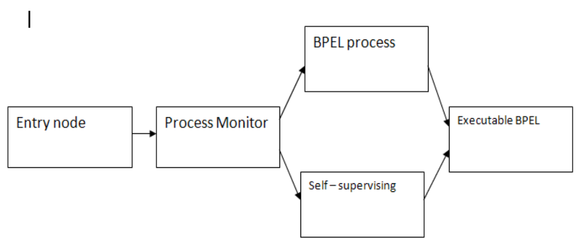
Selection probabilities – For each service type, a tuple containing the probabilities of the corresponding services to be selected. The selection probabilities are periodically recomputed according to the monitored service performance.

Monitoring probability – Probability of a service to be monitored

Observing interval – Period of time in which service performance is monitored, while the selection probabilities tuples remain unchanged.

The adaptation engine is able to interpret and execute instances of the presented model for adaptations and to apply the adaptations to deployed processes and its instances to any process execution engine. The definition of a process contains a set of global variables and the workflow logic expressed as a composition of activities; where implicit or explicit scopes help define variables and activities at different visibility levels. BPEL does not come with a standard graphical representation.

The new BPEL process, which is obtained based on the refactoring approach, has more (strictly, no fewer) partners than the original one. Let us take a look at the structure of a BPDG first because this is helpful to understand our refactoring approach. An important characteristic of a BPDG is that its control dependence sub-graph is a tree with entry node as the root. In the tree, the internal nodes (except the entry) are the control nodes that represent the predicate expressions of the BPEL process, while the leaf nodes represent basic communicating and non-communicating activities.



4.0 Implementation Process

Create an Empty BPEL process project (Using Empty BPEL Process template). Add a new schema MathOp.xsd to the BPEL process project. In this project I used simple types for input and output. You may have complex types depending on your business use case. Also as all operations are binary and all of them share the same input and output signature, Request and Response elements are reused in this project.

During this step also create two global variables 'inputVariable' and 'outputVariable' for the Request and Response data using the Project WSDL.

This activity waits for the occurrence of one event in a set of events and performs the activity associated with that event.

In our case an event translates to a web service operation. If more than one of the events occurs, then the selection of the activity to perform depends on which event occurred first.

If the events occur nearly simultaneously, there is a race and the choice of activity to be performed is dependent on both timing and implementation. As our implementation is going to be state-less atomic services, we may not have a racing condition.

Although WSReL is built for maximum flexibility, there are some constraints we must keep in mind when building a recovery strategy. We must consider: Whether the recovery is associated with a pre or a post condition. Whether the recovery is deal with stateful or conversational services.

In fact, recalling a service might not even be an option, while actions rebind and change Partner link could cause problems if used when the process is in the middle of a conversation. Which actions require that monitoring be re-enacted to discover if they were successful in fixing the anomaly. Note that some actions, ignore, notify, halt, and call, can always be considered successful.

BPEL as a specification does not provide any security concepts that we could leverage. All security aspects are left to the BPEL engine or, in other words, to the BPEL engine wrapper. BPEL processes are implemented as SCA components. So for BPEL processes, we can leverage all security constructs that SCA architecture offers.

A security policy specifies integrity and confidentiality requirements by the combination of the multiple security policy assertions. The signature and encryption predicates are not corresponding to an element of the security policy one to- one. BPEL processes can run on any BPEL-compliant orchestration engine. The engine orchestrates the invocations of the partner web services according to the process specification. The deployment descriptor for the BPEL process below

```
<bpel-dd>

<selectors>

<selector type="activity" id="1">

//invoke[portType="payWSPT" and

@operation ="pay"]

</selector>

</selectors>
<services>

<service name="security">

<propertyclass="authentication"

type="usernametoken"

selectorId="1" >

<propertydata>

<username>deptstore</username>

<password type="digest">edreoptts</password>

</propertydata>

</property>

</service>
…
</services>

</bpel-dd>
```

In the deployment descriptor, we specify the <invoke> activities that require authentication and the username and password to be used for that purpose. The <selector> element is an XPath expression that selects a set of activities with a shared security requirement. The <service> element contains the configuration of a specific middleware service, here the security service; it can contain one or more <property> elements which express security requirements of some process activities. For security, there are three possible values of the attribute class of a property element: authentication, integrity, and confidentiality. The <propertydata> element contains the necessary.

5.0 Conclusion and Future Work

Service compositions provide unprecedented levels of dynamism and flexibility. Using services exposed by third parties, we construct systems whose ownerships are intrinsically distributed, making it hard to reason about the actual functionality and quality of service we can ensure at runtime. The challenge lies in providing composite systems that are robust and dependable. To this end, we blur the lines between design time and runtime validation and provide self-supervision to identify and autonomously react to anomalous situations that may occur during execution. The project has presented one of the few integrated frameworks for both the monitoring and recovery of BPEL processes. We also propose introducing fault tolerance for a fault or error at the web service level by using an adaption of the recovery block technique.

Our proposed solution to the dynamic WS selection problem consists of AR, a novel metric to measure the reliability of each configuration in a WS composition, and strategies to use the computed aggregated reliabilities for dynamically selecting atomic WSs for the operations to be performed on the composite WS.

Aggregated reliability -based Selection Strategy. The rationale behind the AR-based selection strategy is to select an atomic WS for each incoming operation of the composite WS so as to achieve maximum reliability. At runtime, when an incoming operation arrives at a configuration, we first sort the candidate WS operations in non-increasing order of the products of their reliabilities and aggregated reliabilities of the destination configurations

Finally our proposed method introduces two languages for defining monitoring and recovery and explains how to use them to enrich BPEL processes with self-supervision capabilities. We evaluate and classify of different SOA-platforms and security frameworks regarding secure cross-organizational service invocation. Our model is a hybrid reliability model based on both path-based and state-based models. By using the web services estimations of the failure rates based on the doubly stochastic and renewal processes, we determine the reliability of composite web services using the bounded set approach to manage the states explosion in the case of complex systems. We further plan to evaluate the performance implications of our security infrastructure to the overall supporting a specific access control representation for generating the valid composite policies dynamic web service activity response time.

6. REFERENCES

[1] T.S. Dillon, W. Chen, and E. Chang, "Reference Architectural Styles for Service-Oriented Computing," Proc. IFIP Network and Parallel Conf., 2008

[2] Measuring Reliability of Applications Composed of Web Services Hangjung Zo, Derek L. Nazareth, Hemant K. Jain, Proceedings of the 40th Annual Hawaii International Conference on System Sciences (HICSS'07).

[3] Dynamic Web Service Selection for Reliable Web Service Composition IEEE TRANSACTIONS ON SERVICES COMPUTING, VOL. 1, NO. 2, APRIL-JUNE 2008

[4] A.J. Lee, J.P. Boyer, L.E. Olson, and C.A. Gunter, "Defeasible Security Policy Composition for Web Services," Proc. Fourth ACM Workshop Formal Methods in Security (FMSE '06), pp. 45-54, 2006.

[5] Orchestration under Security Constraints Turning Intruders into Mediators Yannick Chevalier, Mohamed Anis Mekki, Micha¨el Rusinowitch LORIA & INRIA Nancy Grand Est, France 2007

[6] J. Cao and T. Dillon, "Checkpointing and Rollback of Wide Area Distributed Applications Using Mobile Agents," Proc. 15th Int'l Parallel and Distributed Processing Symp. (IPDPS '01), 2001.

[7] R. Iyer and P. Velardi, "A Statistical Study of Hardware Related Software Errors in MVS," Proc. IEEE Int'l Symp. Fault-Tolerant Computing (FTCS '84), pp. 192-197, 1984.

[8]D. Gaver, "Random Hazards in Reliability Problems," Technometrics, vol. 5, pp. 211-226, 1963.

[9] Reliability of Series-Parallel Systems, http://www.mathpages. com/home/kmath560/kmath560.htm, 2011.

[10] W. Abramowicz, M. Kaczmarek, and D. Zyskowski, "Duality in Web Services Reliability," Proc. Advanced Int'l Conf. Telecomm. And Int'l Conf. Internet and Web Applications and Services (AICT/ICIW), 2006.

[11] Y. Pan, "Will Reliability Kill the Web Service Composition?" technical report, Dept. of Computer Science, Rutgers Univ., 2009.

[12] A.Charfi and M. Mezini, "Using Aspects for Security Engineering of Web Service Compositions," Proc. IEEE Int'l Conf. Web Services (ICWS '05), pp. 59-66, 2005