Compliance Proofs for Collaborative Interactions using Aspect-Oriented Approach

Adomas Svirskas[1,2], Carine Courbis[1], Refik Molva[1], Justinas Bedžinskas[2]
[1]*Institut Eurécom, Sophia-Antipolis, France;* [2]*Vilnius University, Vilnius, Lithuania*
{adomas.svirskas, carine.courbis, refik.molva}@eurecom.fr; justinas.bedzinskas@gmail.com

Abstract

Service Oriented Architecture approach in general and the Web services technology in particular play significant role in modern collaborative environments. However, it is not enough to have the business functionality of the partners packaged as (Web) services; there is also a need for business-aligned order of interaction between these services (business protocols). Furthermore, it is necessary to guarantee that these protocols are enacted in compliance with the effective policies and regulations. This paper discusses business protocol compliance issues and suggests some techniques for enhancement of business protocols for better compliance. Such enhancements, for example, can support the proposed structured proof of compliance concept and its construction mechanism, which together address the issues of both correct course of collaboration at its critical steps and existence of a tangible proof of correctness of the whole collaborative interaction. Such proof, consisting of individually signed and time-stamped evidence statements, can serve for various audit purposes and, if necessary, in the court of law.

1. Introduction

The concepts of Service Oriented Architecture (SOA) and the recent developments in Web services field provide promising opportunities for integrating data, applications and business processes. The latter, however, is the most complex case of integration as it requires strong support for both business process semantics and technical infrastructure in order to tackle heterogeneity at all levels. Long-running interactions among the involved services can only be successful when the parties involved in the interactions follow commonly agreed protocols and provably comply with the external policies and regulations dictated by the legislative acts.

There are two types of protocols in coordination of service interactions – *peer-to-peer* and *centrally managed*. A process of specifying the business protocols in advance, distributing them to the parties and subsequent common run-time enactment of the protocols on peer-to-peer basis (without the central "hub") is known as *service choreography*. Potential example of such interaction can be a B2B purchasing transaction: request for quote, negotiation, placement of a purchase order, getting information about goods delivery process, payment, etc.

Choreography is well suited for multi-party interactions where parties belong to different domains of control and centralized collaboration management is not possible (or undesirable). Such interactions are referred to as *collaborative interactions* throughout this paper. Web Services Choreography Description Language [25] is the current standard of choreography in WS domain. As opposed to this interaction style, the *orchestration* specifies *individual behavior* of a participant in choreography by defining a description on a process to be executed. Orchestration assumes *existence of an entity, which is the central point of control* and governs overall workflow of activities, by composing a new service from existing services.

Having commonly agreed public collaboration protocols helps companies and government agencies to move closer to organizational interoperability. There are various issues such as correct specification and verification of the protocols, mapping between the public protocols and private implementation mechanisms, protocol lifecycle support by the collaboration partners (adaptivity issues) [1], etc. All these issues are, to a different degree, shared by all the parties and the solutions require common agreements, tools and resources to be provided by each party.

However there are a number of issues related to the involvement of third-parties: government regulatory agencies, law enforcement bodies, civil liberty and/or consumer protection organizations, etc. In other words,

2007 IEEE Congress on Services (SERVICES 2007)
0-7695-2926-7/07 $25.00 © 2007 IEEE

collaborative interactions of commercial and/or government organizations are increasingly regulated by growing number of national and international legal acts, bills and other external rules, in addition to the internal policies and procedures to comply with. There are different kinds of regulations, including corporate governance, security, privacy, financial reporting (e.g. Sarbanes Oxley, Basel II), trade & tariff, environmental, or combined, like the USA Patriot Act. These external policies affect all the participants of the interactions, as opposed to the internal policies of each participant and are, in most cases, legally binding.

Therefore, collaborative interactions in e-Business and e-Government areas need to be compliant with the applicable regulations and the participants must be able to prove their compliance to avoid sanctions and penalties. Compliance, in short, can be defined *as an act or process of complying with a demand or recommendation, observance of official requirements* [2]. Our work aims to complement existing business protocols and business rules solutions in the context of SOA/Web services in order to increase levels of provable compliance of multi-party collaborations. We introduce the concept of compliance structured proof - a verification process of mandatory collaborative interaction steps. The outcome of this process is a single document containing digital signatures, timestamps and other necessary artifacts to prove the validity of the collaboration.

In order to produce such proof of compliance document, the interaction description and enactment need to be enhanced with additional features to validate the course of interaction and contribute to the proof construction at each critical step. We use the AOSD (Aspect-Oriented Software Development) approach [3] for enhancing both the orchestrated (the first prototype has already been implemented) and choreographed collaboration protocols, and, potentially, *interoperability gateways*. The latter is a uniform, policy-driven, software service, deployed on the boundary of each participant's domain to connect the internal systems using public choreographed protocols [1]. There is some related work [4, 5], which demonstrates usage of aspects to enhance orchestrated collaborations – BPEL [6] processes, in particular. Some more general AOP approach to enhance Web services interactions is described in [7].

The rest of this paper is structured as follows. Section 2 discusses compliance issues and potential solutions in the SOA context and explains why our solution can be useful. Section 3 describes the main features of our *compliance structured proof* concept; Section 4 explains the business protocol enhancement approach, followed by Section 5, which concludes the paper.

2. Compliance in SOA-based interactions

According to Tabet [8], the compliance problem is complex and impacts both the functional and technical sides of the business processes within an organization. Requirements to comply with regulations and internal policies often are not understood in a normalized way, creating a high degree of redundancy. From the organizational point of view, one of the initiatives aimed to provide help to the IT managers and architects is the OMG Regulatory Compliance Alliance (ORCA), established by the Object Management Group [9]. The Global Regulatory Information Database (Compliance GRID) [10] is an open database of rules, regulations, standards, and government guidance artifacts. The goal is to provide the de facto compliance reference guide for global (IT) compliance managers.

From these explanations we can see that in collaborative SOA-based peer-to-peer interactions we can have situations when legitimacy of some actions (and/or the whole result of the multi-step multi-party collaborative interaction) can be questioned. We have identified the following main questions:

- Have the necessary actions been performed or not at all? Is the whole result of collaboration valid or not?
- Have the actions been performed in the right order and accordingly to the regulations?
- Were these persons authorized to authorize the actions?
- Who certified the correctness and compliance of the actions?
- At what time exactly the actions have been performed?
- Has the evidence and/or timestamp itself not been tampered with?
- Is the evidence ready to be used in the court of law, according to the legislation in effect?

These and similar questions are notoriously difficult to answer even within the boundaries of one corporate/administrative domain where full access to all the ICT resources is possible (if compliance measures are put in place). Therefore, in multi-party collaborations with limited trust between partners, such questions often become practically unanswerable without a special mechanism agreed and put in place in advance. The proposed solution aims to help answering these questions.

As it was mentioned before, the task of applying compliance rules is complex and resource-consuming, yet unavoidable. A potential solution is to institute controls that enhance the transparency of communications, bringing to light any material deficiencies and highlighting key information that may be material to compliance. This allows controlling the way the key data is processed, distributed, retained, and accessed in day-to-day operations.

Speaking more technically, there is a need for IT tools, which would allow:

- Declarative description of interactions and processes (business protocols), as discussed before;
- Support of data lifecycle (creation, retention, access, flow);
- Verification that the controls meet the regulations (*and so can be shown to be compliant through computational means*) [8].

Ross-Talbot [2] discusses the notion of *Declarative Compliance Systems Architecture*, an approach based on declarative description of collaborative processes and business rules - logical statements about how a system operates, be expressed in the language of the business, referring to real-world business entities [11]. According to Yang et al. [11], business rules can represent, among other things, typical business situations such as escalation ("send this document to a supervisor for approval") and managing exceptions ("make sure that we deal with this within 30 min or as specified in the customer's service-level agreement").

Externalization (isolation, outboarding) of policies/rules (a policy can be defined as a declarative specification of guidelines, rules of conduct, organization, and behavior of entities in a given environment [12]) from business processes helps creating not only reusable, more generic processes, but also reusable and executable sets of rules that maximize business agility and improve visibility, helping to achieve better compliance. There are different rule languages, techniques, frameworks and tools assisting developers to achieve such separation.

It is important, however, to achieve a certain level of standardization when specifying rules and policies. The RuleML [13] is a rapidly evolving rule interchange platform for Distributed systems and Web services particular, including Semantic Web Rules and Semantic Web services. Ross-Talbot et al. in [12] introduce a policy specification language extending RuleML to handle various policy descriptions embedding rules and constraints marked-up in the RuleML language for Web services. The RuleML family of languages provides the expressive power of a declarative rules language with a markup approach enabling modular sublanguages integration for various policy representations [12]. This lays a good foundation for factoring the requirements to leverage commonalities by finding common rules and managing them together, thus eliminating redundancies in data, processes, and systems.

As we can see, many business rules are about *obligations*, they specify action that must be performed. Sometimes, however, systems and people do not act according to the business rules, intentionally or not. This is where compliance becomes important. Rules can ensure compliance within IT Systems, however IT systems cannot always carry out real business actions, and sometimes they can only inform/direct people in the business to act [8]. This is one of the main drivers for some complementary techniques, which would help to build evidence of compliance, to be introduced. The proposed compliance structured proof concept, described in the next section, aims to serve such purpose by providing a mechanism to specify which interaction steps are critical, enforce their ordered enactment and collect testimonies signed by authorized principals that the steps were *performed in compliance with the policies*.

3. Proposed compliance proof approach

As discussed above, declarative descriptions of the collaborative processes and externalized business rules (controls) can help to achieve regulatory compliance of the interactions, however additional tools are needed to verify that the controls meet the regulations. More precisely, automatic verification of processes and rules is highly desirable, so that the execution can be shown to conform to the description.

For this purpose, we introduce the notion of *structured proof* - a verification process of mandatory (and critical) collaborative interaction steps, resulting in a single document containing digital signatures, timestamps and other necessary artifacts to prove the validity of the collaboration. In the context of collaborative multi-party interactions among commercial bodies, public administrations, judicial institutions from, quite often, different countries, it can be useful to have a mechanism that would be twofold:

- Enforce, at runtime, that the critical interactions are performed by the right (duly authorized) person, group, or service and these interactions follow the right order of execution and, maybe, at the right time, if time constraints are in effect.

- Produce a tangible proof of interaction validity that can be stored and used, in front of a court, in case of a dispute and/or investigation.

The proof – as a set of evidences – is incrementally constructed by adding evidence after the execution of each critical collaboration step:

- Sufficient (in terms of amount and accuracy) data must be collected to produce an evidence of the correctness of the collaborative interaction. Evidence can include either a timestamp or a digital signature of one or more documents, certain data exchanged between parties, and/or previous evidences.
- It should not be possible to tamper with the individual evidences, i.e. no alterations are allowed after the evidences have been produced, as described above.
- No collaboration party should be able to deny any evidence they have produced during the interaction.

3.1. Phased structured proof mechanism

The whole process of structured proof can be decomposed into three distinct phases, as follows:

The *design phase*: during this phase, the critical (compliance) steps of a given inter-domain collaborative interaction (workflow) description are specified (marked). In addition, the types of necessary evidence(s) are attached to the critical steps, denoting the information to be produced as evidence and the principals responsible for that. This information, which effectively describes how to produce a complete structured proof document, is specified by those responsible for interaction compliance and is subsequently used by the services implementing the concept. All the information is contained in a document, referred to as *compliance path* (CP) document. The concept of *compliance path* is also used in other areas, for example construction industry for checking of buildings design compliance to the energy codes etc. The purpose is the same as we are discussing – to show, that the procedures are compliant with the regulations.

The *injection/enhancement phase*: this phase is needed to "bracket" the critical collaboration steps by "injecting" the invocations of the services responsible for verification of previous critical steps and the evidence production at the end of each critical step. In other words, the description of the original interactions is enhanced by inserting additional code (advice in an aspect) according to the information contained in the *compliance path* document. The latter, as mentioned before, indicates which steps are considered critical and also contains "evidence templates" for these steps. Section 4 describes the proposed solution of automated enhancement of collaborative interaction descriptions (and supporting enactment mechanisms, e.g. interoperability gateways [1]) with non-functional features.

The *run time phase*: during the enactment of the inter-domain interactions, two actions related to the structured proof mechanism must take place at each critical step – *verification* and *evidence production*.

Firstly, the *verification procedure* (Figure 1) takes as its input a) the structured proof document being constructed, b) the compliance path document, and c) a pointer to the current "location" of the critical step in the overall interaction.

Definitions:

KSA1 - private key of actor A1
KPA1 - public key of actor A1
h - a hash function
KSTTP - private key of the Trusted Third Party
KPTTP - public key of the Trusted Third Party

Validation of a signature:

Signature = $[h(value)]_{KSA1}$
h(value) =? Decryption(signature) = $[Signature]_{KPA1}$ = $[[h(value)]_{KSA1}]_{KPA1}$

Validation of a timestamp:

Timestamp = $[time \mid h(value)]_{KSTTP}$
Decryption(timestamp) = $[timestamp]_{KPTTP}$ = $[[time \mid h(value)]_{KSTTP}]_{KPTTP}$ = time | h(value)

Figure 1. Run-time verification of the steps

Using these input parameters, the procedure can check whether all the expected evidences have been produced by the right principals and the content of evidences is valid. If the conditions are met, the pending critical step (a business service) can be invoked. Otherwise, a breach of foreseen compliance path is detected and a human interaction is required or the collaboration needs to be suspended.

At each critical step, the verification (enforcement of the collaboration course) of previous critical steps is performed before the step can be invoked. When the business actions of the step are performed, production of evidences can take place, provided that the action did not end with an exception, i.e. the step "succeeded".

In this case, the *evidence production* procedure commences, taking as its input the information about evidence(s) to be produced and, as input/output, the structured proof document. As mentioned before, there are two types of evidences: *signature* and *timestamp*. *Timestamp evidence* is produced by a trusted third party (the timestamp authority) that takes as input the hash code of the value to sign with the current time using its private key. The value to timestamp can be a previous evidence to indicate when the action has been done, the whole structured proof document, or an arbitrary value from the message related to the step. In a complex multi-domain/multi-country collaboration case, synchronization between the time stamping authorities is an issue itself. This issue, of course, is out of the scope of our work. There are research projects (such as BALTICTIME [14]) dedicated to developing the interface between National Time Standard Authorities (NTSA) and Time Stamping Authorities (TSA) ensuring coherent time scale synchronization and time scale transfer between atomic clocks of NTSA and time stamp server of TSA. Our solution would rely on presence of such TSA, as long as it delivers on its promise:

e-Document+eSignature+eTime Stamp = Legal Power

Signature evidence is produced by an actor of the collaboration, which takes value to sign as input along with the previously produced evidence. Then the respective hash codes are calculated and signed (one signature) with the actor's private key. Objects to be signed can be external documents (accessible via an URI, for example), inline data fragments exchanged between partners, and previous signatures or timestamps contained in the structured proof document. As this digital signature is generated by encrypting data with the private key of an actor, it serves the purposes of proving data origin and integrity, as well non-repudiation.

It is necessary to point out, that in many cases such evidence is roughly equivalent to a paper-based signature, i.e. it can attest something which has or has not been performed. Compliance structured proof, in other words, is a set of affidavits - formal statements of facts, signed by the *declarants* (or the *affiants*, more precisely) and "witnessed" electronically, in our case.

3.2. Conceptual solution architecture

A number of different artifacts are needed to support the whole structured proof lifecycle. Firstly, we have created two XML schemas to model the data part of the structured proof mechanism: one for the structured proof and one for the compliance path. At the moment we using any GUI tool to facilitate the design phase of the structured proof; this is subject of the future work, we are looking into possibilities to use Domain Specific Languages (and tools) for this.

The injection/enhancement phase is explained in Section 4. At the run-time, for each compliance-critical step of the collaborative interaction, three additional services need to be used, in addition to the business service itself in order to support the run-time phase of the structured proof construction. These services can be described as follows:

The *verification service* helps to enforce the right execution order of the compliance-critical steps. According to the given compliance path specification, the structured proof document being circulated, and the current critical step, this service verifies that all the expected previous evidences are in the right order and correct. The principal, who has signed the evidence, must be the one indicated in the compliance path document as responsible for signing or needs to delegate the right accordingly. In case of any mismatch during verification, this service raises an exception.

The *signing service* is responsible for signing one or more objects and for inserting the signature (evidence), into the right place in the structured proof document. Security of these mechanisms is based on the asymmetric cryptographic system where the private key of an actor is used to sign whereas his public one to decrypt it. As the private keys are very sensitive items, and anyone possessing the private key can sign documents claiming to be the person to whom the private key belongs, each collaborating party must have a signing service inside their organization to diminish the risks associated with distribution of the private keys.

We are using the standardized XML signature format [15] for digital signatures. As our prototype is based on the Java Platform, we will be able to use the newest security features of Java Platform Standard Edition 6 (Java SE 6), which contains built-in functionality dedicated to the creation and manipulation of XML signatures [16]. The new signature-related features of Java SE 6 API are also very helpful for integration with the JAXB (Java Architecture for XML Binding) library [17] to handle the mapping between XML and Java.

One should bear in mind, however, that Java XML Digital Signature API's `sign` and `validate` operations are computationally expensive: they can take up more than 30 percent of CPU time. The PKCS#11 Cryptographic Token Interface Standard [18] defines the native programming interfaces to the cryptographic tokens such as hardware cryptographic accelerators and smart cards. PKCS#11 provides

increased performance and scaling through transparent access to hardware cryptographic acceleration without requiring modification of existing application code if Java Cryptography Extension (JCE) [19] has been configured to use the Sun PKCS#11 provider, which in turn has been configured to use the underlying accelerator hardware [20].

Finally, the *time stamping service*: a trusted third party (e.g. Time Stamping Authority, as proposed in [14]), which creates and signs (with its private key) the timestamps for evidence. Obtained timestamp then needs to be inserted into the right place of the structured proof document. A trusted third party that acts as a notary might also be used to write the evidences in the structured proof document only when it has received all of them from the different partners involved in the interactions who do not trust each other.

4. Enhancement of collaboration protocols

This section explains the *injection/enhancement phase of the structured proof mechanism* in greater level of detail. In general, the chosen approach can also be used to add other non-functional features to Web services interactions. Our goal is to control the correct sequence of critical interactions steps and collect the evidences (testimonies) of compliance with the foreseen policies, expressed as compliance path, in this case.

If we look into the real collaboration protocol descriptions (e.g. BPEL), we can notice that the implementation details of a feature such as structured proof mechanism can be scattered across different artifacts of collaboration protocol description. Enhancing a workflow definition with such a feature is complex and error-prone as the enhancement cross-cuts across different artifacts of different nature. For example, in the case of a BPEL process, the artifacts that need to be enhanced are the following: the BPEL process description itself; the WSDL (Web Service Description Language) of the BPEL Web service; the Process Deployment Descriptor (PDD) document; the Web Service Deployment Descriptor (WSDD) document, and maybe the Java stub of the BPEL Web service. Therefore, when a new feature needs to be introduced, an automated method to modify these artifacts altogether consistently would be highly desirable. We believe that an aspect-oriented approach is appropriate for this.

In this case, a module (*an aspect*), containing all the information related to the enhancements of different artifacts, is necessary. Such aspect has to be designed and created when a new feature needs to be plugged into the collaboration. As the entire enhancement information is kept now in one place, creating and maintaining such module is easier than maintaining several artifacts. An aspect is composed of three parts, like AspectJ [21], the most mature AOP language to specify aspects on Java classes:

- The *pointcuts* part, which specifies the locations (such as the invocation of some service in a BPEL process) where the enhancements should be injected
- The *advice* part, which indicates what kind of enhancement must be performed (such as add a new `invoke` activity in a BPEL document) before the invocation of functional service;
- The inter-type declarations, which indicate structural information such as new instance variables in Java classes. In our case, we will call them *inter-protocol declarations*. Examples of such declarations are namespaces, service names, port numbers, and WSDL of the new Web services to be invoked etc.

A service – *aspect-weaver* – is doing the actual injection of the pieces of advice at the locations selected by the pointcuts in the different artifacts. The weaver service "interprets" an aspect and automatically injects the enhancements into the different documents. In comparison, the AspectJ weaver only deals with one data format – Java classes, while our solution can process many types of artifacts.

The mechanism used to denote the aspects needs to be generic enough to meet the enhancement needs of both orchestrated and choreographed workflows i.e. only the aspect weavers are workflow language (such as BPEL) and, if necessary, deployment platform dependent, whereas the aspect language itself is platform-independent. The latter also needs to be easy to understand and use by people who are neither workflow nor aspect-oriented technology experts. For example, when designing the pointcut mechanism to select nodes in the XML documents, we are considering these alternatives: XPath [22], a powerful but rather complex language, or some potentially easier mechanism to select one or more WS service invocations.

As it was mentioned earlier, there are a number of technical artifacts to be modified when injecting the new features. Some of these artifacts are used to describe the process itself while the other ones are used to deploy the process. The weaver is organized as a set of transformers, which can be configured into different combinations in particular environments. The need for such flexibility is dictated by the fact that

some of the artifacts depend on particular BPEL engine and the application servers used to deploy individual services. Figure 2 shows an overview of the proposed architecture, which allows injecting new security features into a BPEL process description.

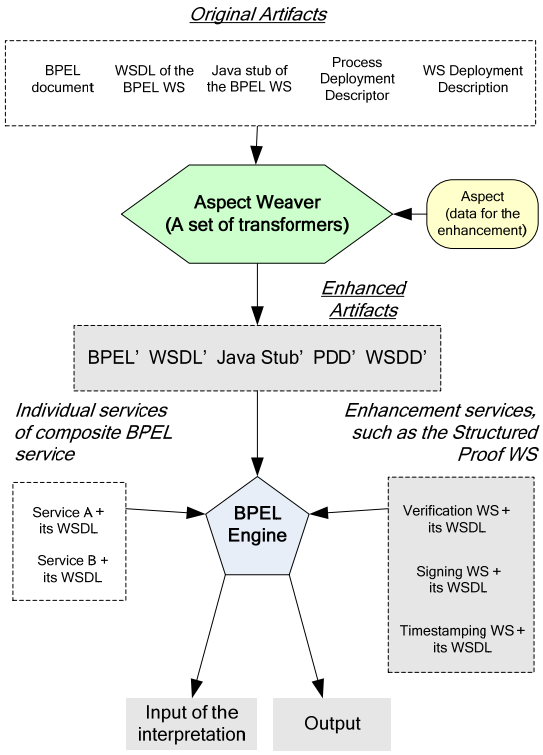

Figure 2. Overall picture of the interaction enhancement solution architecture

Figure 3 illustrates functionality of the BPEL process transformer, which has the following main (and other, as depicted) features:

- Namespaces of the new verifier and signer services (WS) are added in the process node
- Partner link for each WS are added into the partnerLinks node
- New fault & compensate handlers may be added if needed by the new WS
- New invoke activities to new WS are added to "bracket" the critical step
- New link elements need to be declared if the workflow uses links to explicitly connect a source activity to a target activity instead of following the activities nested directly

Looking from the deployment point of view, the Process Deployment Descriptor (PDD, .pdd) file describes the relationship between the partner links defined in the BPEL (the process description itself) file and the implementation required to interact with actual partner endpoints. The .pdd file indicates where the actual *endpoint references* are. An endpoint reference conveys the information needed to access a Web service endpoint, which indicates a specific location for accessing a Web service using a specific protocol and data format.

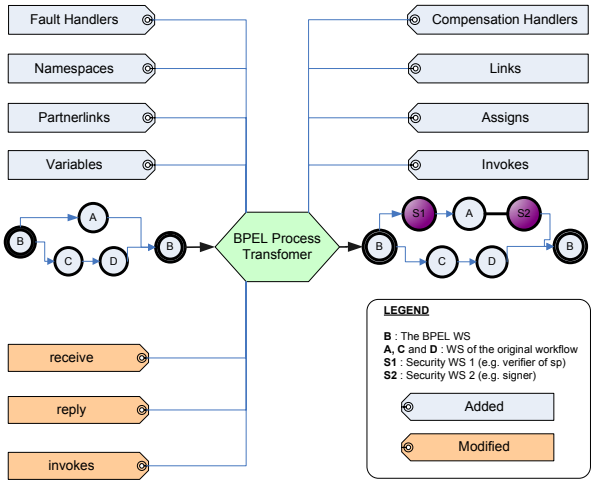

Figure 3. BPEL process transformer architecture

During deployment of a process, each role defined in a partner link is assigned an endpoint reference. Using endpoint references in deployment makes possible the dynamic selection of a service partner. The .pdd file is an integral part of the deployment package for the process; therefore it needs to be modified accordingly, along with the BPEL file itself, during the enhancement step:

- Service binding information (partnerLink element) of the new services needs to be added into the partnerLinks element of PDD
- The namespace and WSDL location of the new WS need to be added into the wsdlReferences element of PDD

In our case the PDD file format [23] is specific to the ActiveBPEL product available from Active Endpoints, Inc., which we are using for our experiments.

Such modular organization of the weaver increases its applicability and reusability – different "pipelines" of transformers can meet different needs. A particular instance of the weaver can be assembled dynamically based of the aspect description and the configuration, which matches particular deployment needs. For the sake of brevity we discussed here only the BPEL and PDD transformers.

5. Conclusions

The number of complex multi-domain/multi-country collaborations is constantly increasing, as the SOA concepts and supporting tools are maturing. To avoid repudiations of execution and ensure compliance with the business rules as well as the external policies and regulations, new mechanisms need to be proposed.

This paper proposes a mechanism of compliance structured proof that can enforce, at runtime, the critical interactions and produce a tangible proof of interaction validity that can be used in a court of law if applicable. The outcome of this process is a single document containing digital signatures, timestamps and other necessary artifacts to prove the validity of the collaboration. Such compliance proof is incrementally constructed by adding signatures and timestamps after the execution of each critical collaboration step. In addition to that, the mechanism verifies that each critical step along the compliance path is performed in correct order, i.e. having the previous critical steps properly completed. If a breach of the foreseen compliance path is detected, an exception is raised and the collaboration is suspended.

In order to inject, in a systematic way, this mechanism into any existing collaboration, this paper proposes to use an AOSD-based approach. The aspects contain the necessary pieces of information to enhance the orchestrated and choreographed collaborations, as well as interoperability gateways with non-functional features such as the explained compliance structured proof mechanism.

A first prototype of the solution has been implemented using BPEL orchestration platform. This prototype will be used to assess solution performance and suitability before moving towards enhancing choreographed interactions. A good case study for application of such compliance proof mechanism can be collaboration between public administrations of different EU Member States in legal/law enforcement domain where validity of interaction steps is highly important.

6. Acknowledgements

The work presented here is partially funded by the European Commission under contract IST-2004-026650 through the project R4eGov [24]. The authors would like to thank members of the organizations involved in R4eGov for their contribution: SAP Research Labs, Thales Security, University of Hamburg, Unisys Belgium, Europol, Eurojust, Austrian Federal Government and others.

We also thank our master-level students at Institut Eurécom – M. Serpantie, C. Lanza, H. Kadri and R. Cosson – for their contribution to this work.

10. References

[1] Svirskas A. et al., Adaptive Support of Inter-Domain Collaborative Protocols using Web Services and Software Agents, Frontiers in Artificial Intelligence and Applications, volume 155, IOS Press, 2007

[2] Ross-Talbot, S., A Declarative Compliance Systems Architecture, European Business Rules Conference, 2005

[3] Kiczales, G. et al. Aspect Oriented Programming, ECOOP, 1997

[4] Courbis, C. & Finkelstein, A., Weaving Aspects into Web Service Orchestrations, Proceedings of the 3rd International Conference on Web Services (ICWS), 2005

[5] Charfi A. & Mezini M., Aspect-Oriented Workflow Languages Proceedings of the 14th International Conference on Cooperative Information Systems (CoopIS), 2006

[6] BPEL, Web Services Business Process Execution Language, OASIS, version 2, 2007

[7] Ortiz G., Hernández, J., Sánchez, F., Model Driven Extra-Functional Properties for Web Services, Proceedings of the IEEE Services Computing Workshops (SCW'06), 2006

[8] Tabet, S., SOA and Regulatory Compliance, 2004

[9] OMG Regulatory Compliance Alliance (ORCA)

[10] ORCA's Global Regulatory Information Database

[11] Yang, J. et al., "A Rule Based Approach to the Service Composition Life-Cycle", Proceedings of the Fourth International Conference on Web Information Systems Engineering (WISE'03), 2003

[12] Ross-Talbot, S. et al. "A generalized RuleML-based Declarative Policy specification language for Web Services", 2004

[13] Schema Specification of RuleML 0.91, the Rule Markup Initiative, 2006

[14] Reinforcing eGovernment services in Baltic States through legal and accountable digital Time stamp (BALTICTIME), an EU IST FP6 STREP research project

[15] W3C, XML_Signature Syntax and Processing, W3C recommendation, Feb 2002

[16] Sun Microsystems, The Java Tutorial, Trail: Security, Lesson: Generating and verifying Signatures

[17] Sun Microsystems, Java Architecture for XML Binding

[18] PKCS#11 – Cryptographic Token Interface Standard

[19] Sun Microsystems, Java Cryptography Extension, 2006

[20] Sun Microsystems, JCE and Hardware Acceleration/ Smartcard Support for Java Platform SE 6, 2006

[21] AspectJ, The AspectJ Programming Guide, AspectJ v5

[22] XML Path Language (XPath), Version 1.0, W3C Recommendation, 1999

[23] ActiveBPEL, PDD Format for ActiveBPEL 2.0

[24] Towards e-Administration in the large (R4eGov), an EU IST FP6 Integrated research project

[25] Web Services Choreography Description Language, WS-CDL, W3C Recommendation, 2005