We are IntechOpen, the world's leading publisher of Open Access books Built by scientists, for scientists

4,800
Open access books available

122,000
International authors and editors

135M
Downloads

Our authors are among the

154
Countries delivered to

TOP 1%
most cited scientists

12.2%
Contributors from top 500 universities

CLARIVATE ANALYTICS BOOK CITATION INDEX INDEXED

WEB OF SCIENCE™

Selection of our books indexed in the Book Citation Index in Web of Science™ Core Collection (BKCI)

Interested in publishing with us?
Contact book.department@intechopen.com

Numbers displayed above are based on latest data collected.
For more information visit www.intechopen.com

A testing process for Interoperability and Conformance of secure Web Services

Spyridon Papastergiou and Despina Polemi
University Department of Informatics, University of Piraeus
80 Karaoli & Dimitriou Str, 185 34 Piraeus, Hellas

1. Introduction

The design, development and implementation of electronic (e-) services relying on XML and Web Service (WS)-based technologies is the current trend in achieving interoperability. E-services can be offered either as autonomous Web Services or embedded in Service Oriented Architectures (SOAs) (High et al., 2005).
In this context, despite the fact that applications with similar business goals adopt the same technical standards, quite often their interactions capabilities are extremely limited. Thus, application developers show an increasing concern for evaluating interoperability between common services which are offered either autonomously or through a SOA. The creation of a proper framework (EIF) has a significant importance in the evaluation of interoperability of such services and is accomplished by the precise definition of the applied standards and guidelines which guarantee the interaction of the services. Existing testing methodologies developed by various organizations (e.g ISO/IEC 9646, ESTI) treat the interoperability of services as a generic problem. They merely provide guidelines and describe high level testing procedures that can be applied to test interoperability of various telecommunication as well as software and data communication systems. Most Web Service-oriented methodologies (i.e. WS-I, ebXML IIC framework) demonstrate weaknesses as they are not capable of testing all the required aspects that compose an interoperability framework and mostly the security aspects of the message content.
Additionally, in literature, specific testing types (Saglietti et al., 2008) have been presented defining diverse testing approaches that treat the applications under test either as white boxes having full knowledge of the software or as black boxes without any understanding of their internal behaviour or even as grey boxes with limited knowledge of their internal architecture. The nature the WSs (e.g. geographic distribution of the examined WSs and dependencies with external trusted third parties) plays an important role in the adoption of the most appropriate testing type as they raise specific challenges that should be underlined and taken into account.
Therefore, there is a specific need for targeted methodologies and frameworks that check and guarantee the end-to-end application interaction capabilities of common Web Services and follow and deploy the most appropriate testing strategies covering all WSs aspects. Identifying this need, this paper proposes a well-formed grey box testing methodology

entitled ICoM, able to test whether various services achieve communication effectively based on the adopted standards. It defines the precise structure of the involved parties, specifies distinct steps to follow, describes concrete tests that should be applied, and enables the execution of specific testing suites. ICoM has been applied in order to evaluate the interoperability of the existing autonomous SELIS e-invoicing service (Kaliontzoglou et al., 2006) and the SWEB e-invoicing service embedded in a SOA-based platform (SWEB), (Karantjias et al., 2008).

2. Prior Work

This section presents the existing testing methodologies and frameworks illustrating their weaknesses and indicating the need for a more holistic methodological framework. Widely used types of testing are also described identifying the most appropriate method that should be applied to WSs due to their inherent characteristics.

2.1 Existing Testing Methodologies and Frameworks

Traditionally, interoperability testing methodologies for the Internet Protocols have been used extensively in the telecommunication industry and in the Internet world. For example the Open Systems Interconnection - Conformance Testing Methodology and Framework (ISO/IEC 9646) (OSI, 1997) is a widely spread and successfully applied conformance testing methodology which has evolved over the years. Nevertheless, it is considered as overly generic framework that allows a high degree of freedom and gives little practical guidance (ETSI, 1998).

The European Telecommunications Standards Institute (ETSI), acknowledging the importance of the testing methodologies, has also contributed towards this direction. ETSI defined a more integrated framework that consists of two primitive types of tests, the conformance and the interoperability testing. The ETSI conformance testing (Moseley et al., 2003), (ETSI, 1995) is based on the ISO/IEC 9646 using its principles as a basis, but not as strict guidelines. It focuses mostly on making easier, more applicable and more readable the use of test suites in the proposed methodology. Therefore, ETSI defined a core language the Testing and Test Control Notation TTCN-3 (ETSI, 2002), (Dibuz & Kremer, 2003) that can be used for the specification of test suites which are independent of test methods, layers and protocols.

On the other hand, ETSI interoperability testing (ETSI, 2007) constitutes a generic approach merely providing guidance on the specification and execution of the interoperability tests. These guidelines are in the form of recommendations rather than strict rules. The TTCN-3 can also be used in this kind of testing offering a higher level of flexibility.

Despite the independent operation of both types of testing, they are closely connected satisfying different objectives (Kulvatunyou et al., 2003). Conformance Testing checks to what extent a solution conforms to the corresponding specification or standard. Interoperability Testing proves the end-to-end functionality between the solutions. It should be noted that the use of either type of test does not guarantee interoperability.

A significant limitation of both abovementioned methodologies is that they treat the interoperability of the WS-based services as a generic problem providing generic testing practices.

Currently, there are only two widely used WSs testing methodologies, WS-I and ebXML: The Web Service Interoperability Standardization Organization (WS-I) (Seely & Lauzon, 2005), (Ehnebuske, 2003) is an industry consortium chartered to promote WS interoperability across platforms, operating systems, and programming languages. It has released a number of profiles and testing tools that compose a scalable testing environment. The shortcomings of this methodology are the following:

1. WS-I does not support the definition of specific and discrete test cases that should be followed during the tests.
2. WS-I tools achieve to monitor only the message flow, without being able to control the testing execution.
3. WS-I tools do not achieve to support a wide range of evaluation criteria (interoperability areas or multiple types of testing). They achieve to test only the conformance of the exchanged messages and the defined services' descriptions against the appropriate standards. They fail to cover criteria regarding the interoperability and the conformance of the applied security features of the message content and the transformation of the exchanged documents between different formats.

OASIS ebXML (ebXML, 2001) is an end-to-end B2B XML framework that provides concrete specifications for dynamic B2B collaborations. It has specified the Implementation, Interoperability, and Conformance (IIC) test framework (OASIS, 2003), (Lee, 2005), (Kim & Yun, 2003) describing the required architecture and providing the necessary test material to be processed by the architecture, a mark-up language and format for representing test requirements, and test suites (set of Test Cases). This approach has the following shortcomings:

1. ebXML IIC is intended to support conformance and interoperability testing only for ebXML specifications and implementations.
2. ebXML IIC does not impute the responsibility of a communication failure to the corresponding system.
3. ebXML IIC does not cover criteria regarding the interoperability and the conformance of the applied security features of the message content and the transformation of the exchanged documents between different formats.

Despite their weaknesses, the above frameworks can be used as the basis for the development of enhanced and more targeted methodologies to test and guarantee the inter-working of common WS-based applications.

2.2 Testing Types Existing Testing

Software testing, including interoperability testing, may take place at different levels of depth, depending on the actual known technical details of the application under test. The bibliography acknowledges three main types of testing: black box, white box and grey box testing (Peyton et al., 2008).

- Black box testing treats the software as a black-box where only the inputs and outputs of the black box are tested without any understanding of the internal behaviour or the adopted specifications. It aims to test the functionality according to the requirements. Thus, the tester inputs data and only sees the output from the test object based on the objects published and known interfaces.

- In White box testing, the tester has access and knowledge of the internal data structures, code, and algorithms of the software. A White Box tester typically analyzes source code, derives the corresponding test cases and targets specific code paths to achieve a certain level of code coverage.
- In recent years the term Grey Box testing has come into common usage and it refers to a technique of testing the system with limited knowledge of its internal architecture. The tester usually has access to specification documents (beyond simple requirements) and generates tests based on information such as state-based models or architecture diagrams.

The selection of the most appropriate testing type relies on the nature of the application under test. Web Services (WSs) are composed by a set of related and integrated services (High et al., 2005). The main characteristics of these services are the following (Rizwan & Mamoon, 2007):

- They can be distributed in the sense that they are located in different geographic areas, having independent capabilities and are accessible only via specific interfaces that are described by WSDL documents.
- They can be implemented in diverse programming languages and support independent operating systems.
- They can be chained with dependencies on other trusted third parties such as PKI and timestamp authorities.

The business design of WSs is essentially a composition of these repeatable services which represents and forms the desired business logic. The above features of the WSs introduce several challenges with respect to testing. White box testing especially in a SOA environment is quite impractical to perform due to the nature of the WSs that make access to source code or binaries very difficult. On the other hand, the WS interoperability testing methodologies (WS-I, ebXML IIC) adopted in the frameworks analyzed in the previous paragraph only enable black box testing of WS, leading to limited and inefficient test coverage due to the "blind" nature of that type of testing.
The distributed nature of Web Services makes Grey Box testing ideal for detecting interoperability flaws on the communication channels between WSs, mostly by leveraging the rich information contained in the descriptions of the services interfaces (WSDL documents). A Grey Box Tester is able to identify at a high level the internal structure of the tested WSs, defining the composed services and accessing the provided interfaces having limited or even no access to the actual code. Additionally, several test cases regarding the deployment of the security features and the communication protocols can be performed covering all the aspects of the WSs. Therefore grey box testing has been adopted as the most appropriate type to use in the methodology proposed in this paper.

3. The ICoM methodology

In this Section, we will describe the structure of the proposed methodology. Before presenting its main features and implementation steps, it is imperative to present the requirements it satisfies which overpass the weaknesses of the existing frameworks. The requirements that ICoM satisfies are:

- **Clarity**: the methodology specifies concisely the evaluated entities and the required information items for the testing process (e.g. test cases and test data).
- **Adaptability** & Extensibility: the methodology is extensible in the sense it may easily evaluate new aspects of the WSs and adopt and integrate new testing tools and libraries.
- **Flexibility**: the methodology is parameterizable in the sense that different parts of the methodology can be adopted for the realization of specific sets of test suites.
- **Structural**: the methodology is structured and comprises a definite and precise set of implementation steps.
- **Independency & scalability**: the methodology offers a high level of independency from testing technologies, the number of entities involved and the platforms hosting systems under test. This fact will offer the possibility of testing interoperability on several different WSs.

In order for the methodology to accomplish its objective goals comprises of four district phases:

- *Phase 1* "Entity identification and setting": the entities involved are identified and their specific setting and order for the tests is defined. The involved entities are the Systems under Test (SuTs) being evaluated for interoperability and conformance and the Test Coordination Infrastructure (TCI) which monitors the testing suites applied by the SuTs in order to identify any erroneous behaviour.
- *Phase 2* "Entity structure definition": the structure of the involved entities is identified. This includes the SuTs, which consist of the actual WS under evaluation and the specific structural additions of the testing infrastructure that are required in the testing procedure. The internal structure of the WS is immutable and an analysis of the available services is carried out. This phase also includes the specification of the precise structure of the TCI.
- *Phase 3* "Conformance testing": the definition and generation of the executed conformance test cases for each SuT, based on which the parameterization of the SuTs structural additions and the TCI is completed. Each SuT is evaluated against the adopted standards.
- *Phase 4* "Interoperability testing": the formulation and the derivation of the interoperability test cases, performing the necessary parameterization of the the SuTs structural additions and the TCI. The actual interoperability testing between all SuTs' communications takes place during this phase.

The following section describes these four phases in detail.

3.1 Phase 1: Entity Identification and Setting

As shown in the Figure 1, the initial step is the identification of the exact number, order and setting of all entities participating in the testing suite. The full methodology deployment demands the execution of the following process:

- *Entities (SuTs) definition*: defining the entities (SuT 1… SuT N (Figure 1), TCI) that participate in the test.
- *Web Service (WS) declaration*: declaring the Web Service that will be tested.

- *Testing Scenario*: specifying a testing scenario. A testing scenario consists of a sequence of actions described at a high level which must be performed by the SuTs in order to execute a specific electronic transaction.
- *Role determination*: defining the entities' role based on the specified testing scenario. Specifically, the entity that initializes the execution of the testing suite and the sequence of the other entities, as this is described in the testing scenario.

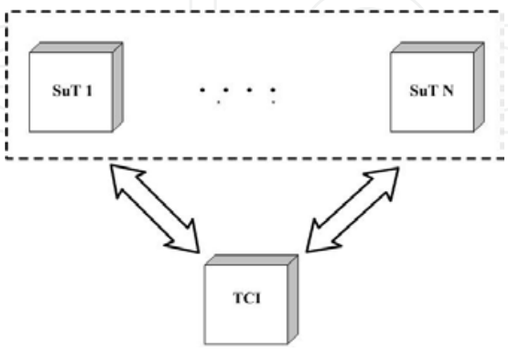


Fig. 1. General Entity Setting

The TCI, as presented in Phases 3 and 4 of the proposed methodology, acting as the initiator and an intermediate entity of the testing suite analyzes the testing results and indicates the conformance and the interoperability of the entities under test.

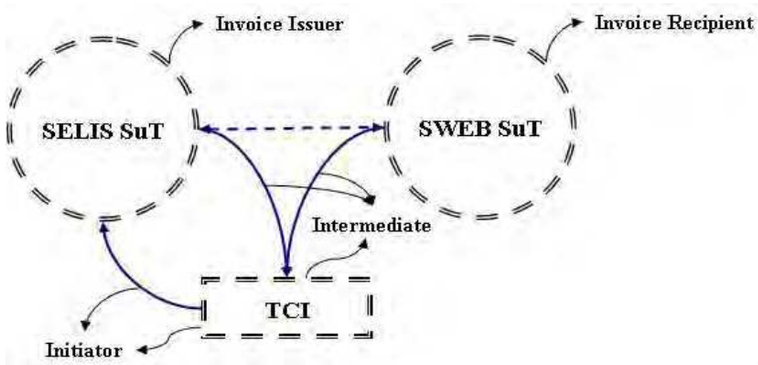


Fig. 2. Example Entities Identification

In this paper, we will use ICoM to test the interaction capabilities of two existing and fully operational WS-based solutions for e-invoicing, the autonomous SELIS service (Papastergiou et al., 2007b) and the SOA-based SWEB service (Papastergiou et al., 2007a). These services shall be used as demonstration material (case study) throughout the methodology presentation to better show its capabilities with specific examples. A case study example that imprints the main steps of the current phase is the following:

Example 1

The involved entities that participate in the test, Figure 2, are the SELIS SuT and the SWEB SuT while the provided invoicing service constitutes the Web Service that will be tested. The testing scenario we shall be following involves a communication initiated by SELIS SuT,

acting as the Invoice Issuer (*role: Issuer*) that invokes SWEB SuT, which acts as the Invoice Recipient (*role: Recipient*) handling the dispatched invoice and responding with an acknowledgment. This scenario covers specific aspects of an invoicing transaction such as the issuance, dispatch and receipt of an invoice document. The TCI triggers, monitors the execution of the applied testing suites.

3.2 Phase 2: Entity Structure Definition

During this phase, the actual structure of the involved entities, TCI and SuTs, should be defined and shaped precisely by the responsible operators, indicating their constituent parts. The final form of each entity depends upon the tests that will be performed and the suites that will be followed.

3.2.1 Definition System under Test (SuT)

A SuT primarily contains the WS to be tested (WSuT). A common WSuT, as depicted in Figure 3, includes a set of primary services $\{S_1,\ldots,S_n\}$ that are combined to form and execute the WS's business logic. Each service encapsulates an explicit function having a number of Inputs and producing a specific Output. It may also interact with other services in order to complete its objective goal.

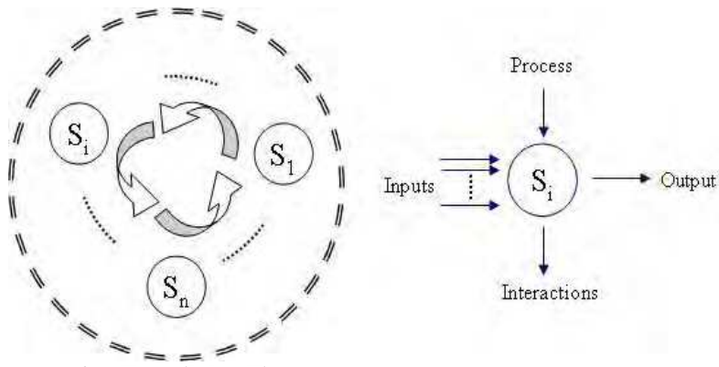


Fig. 3. Web Service under Test (WSuT)

A representative example of an advanced WSuT (Papastergiou et al., 2009), (Karantjias et al., 2008) may include a set of primary services such as the following:

- communication with legacy systems and proper data transformation,
- document management and application of digital signatures and / or encryption as a secure mechanism on the business level, and
- messaging formulation, processing and application of security mechanisms on the messaging level.

All these services, following concrete procedures, manage and derive data that are based on specific standards. The factual conformance of these data to the individual requirements specified by the corresponding specifications and the ability of other systems to handle these data correctly constitute factors that are able to leverage and infer the interoperability and the conformance capabilities of a WSuT. Take for example the case where the security service of a "WSuT A" signs a XML document according to the W3C XML Digital Signature

standard. Initially, the produced signed document should be tested for compliance with the principles of this standard and then there should be test that another "WSuT B" is able to successfully verify that signature. The conclusion that comes from this example is that the "WSuT A" produces signed documents that conform to the W3C XML Digital Signature standard and is able to interoperate successfully with the "WSuT B" on the application of signature at the business level.

Thus, in order for a WSuT to derive the required evaluation data, it should perform a certain number of activities as represented and indicated by specific test cases. A test case comprises a set of conditions and variable and depicts the test control logic which is sufficient for the execution of a testing suite.

ICoM deals with the execution of test cases and the collection of the respective data defining two structural components the *Service Orchestration Engine* and the *Report Engine* as seen on the following figure. These components are part of the testing infrastructure and along with the WSuT compose an integrated SuT.

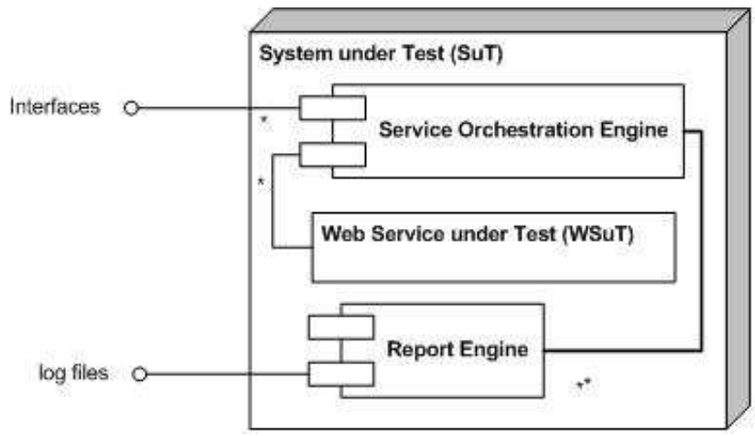


Fig. 4. System under Test (SuT)

The Service Orchestration Engine is responsible to coordinate the available WSuT's services to perform specific workflows. Each workflow is represented by a predefined BPEL process which imprints the logic of an executed test case. The BPEL test cases that will be adopted and executed are designed during phases 3 and 4 where the actual conformance and interoperability testing is performed. In these phases, the Engine undergoes the appropriate parameterization in order to run the defined BPEL processes. The parameterization includes also the provision of specific interfaces that can be invoked by the TCI in order to initiate the execution of a BPEL test case promoting the concept of automation of the testing procedure.

Thus, the Service Orchestration Engine provides a thorough control of the testing suite and enables the effective accumulation of the evaluated data by adopting BPEL processes as forms to represent and execute test cases. The nature of the accumulated data depends on the type and the purpose of the executed test case and varies from simple XML documents and documents with security features to secure or not SOAP messages and supported XSLT files used for document transformation.

The second structural component of the SuT, the Report Engine acts as the collector of the evaluated data. Its main responsible focuses on gathering and consolidating them in log files.

Concluding, ICoM defines four major steps that lead to the definition of a SuT.

1. *service identification*: Identification of the primary services of the WSuT.
2. *interface analysis*: Analysis of the services' interfaces and data types used per service, extracting the messages types of the inputs and outputs as described in the WSDL documents. This may include an extensive analysis of several WSDL documents.
3. *standards definition*: definition of the standards that each WSuT adopts and implements.
4. *testing components adaptation*: adaptation of the two structural testing components, the Service Orchestration Engine and the Report Engine to the WSuT.

It should be noted that the above steps are completely independent of the underlying infrastructure and the WSuT implementation language and do not bind ICoM to a specific technological solution. The application of these steps in our demonstrated case study is depicted in the following example:

Example 2

The SELIS and SWEB SuTs are setup following the aforementioned four basic steps. In SELIS, we have identified three specific areas of services with diverse functions. These services are divided into:

- *Basic services*, which provide the basic functions used to perform primitive tasks. SELIS includes document management services, message and document transformation services, message forwarding services, publication and query services and notification services.
- *Security mechanisms and services* that address the security requirements of SELIS-invoicing. It supports the following security mechanisms: digital signatures, advanced electronic signatures, encryption, timestamping and credential management.
- *Infrastructure support services*, which manage the connection with the back-office systems such as databases, wrapper software on top of legacy systems, existing ERPs etc.

Each of these services has specific known interfaces as described by the corresponding WSDL documents having concise inputs and giving a concrete output. The standards that adopted in the services implementations are the following:

- the XML common business library version 4.0 (xCBL 4.0) and the Exact ERP XML schema, for the representation of the invoice information,
- the Extensible Stylesheet Language (XSL) Transformations, for the transformation of the invoice documents,
- several security standards such as W3C XML Encryption, W3C XML Digital Signature and XML Advanced Electronic Signature (XAdES) for the application of the appropriate security features on the business level and
- SOAP and the WS-Security standard for Web Services invocation and
- the Web Services Description Language (WSDL) for the description of invoked services.

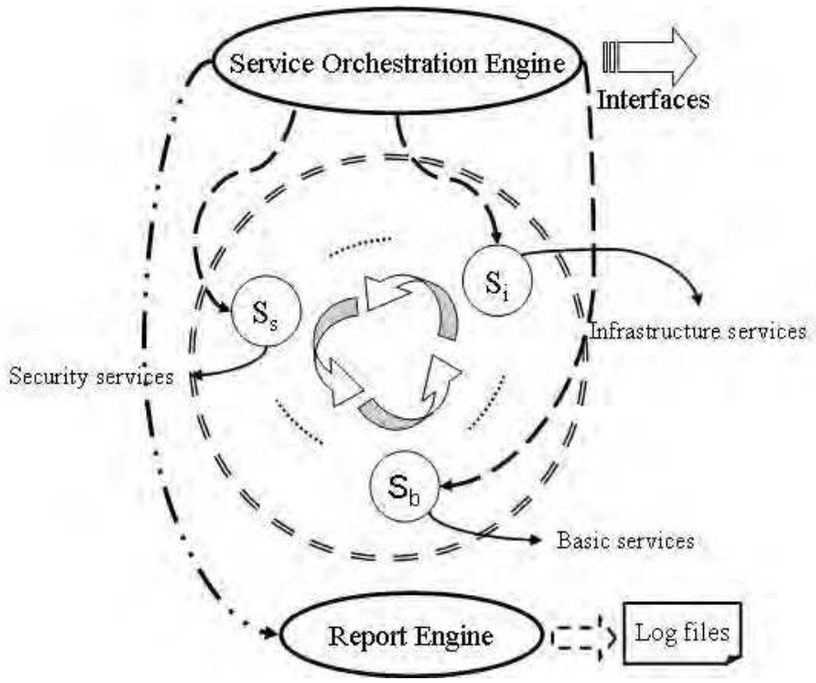


Fig. 5. SELIS SuT

Figure 5 depicts the overall form of the defined SELIS SuT which is composed by the two proposed structural testing components, the Service Orchestration Engine and the Report Engine and the SELIS WSuT. The process is similar in the case of SWEB.

3.2.2 Test Coordination Infrastructure (TCI)

The logical and organizational structure of the TCI is independent of the nature of the SuTs. On the contrary, technologically, it offers significant flexibility and scalability allowing the upgrade of already adopted testing tools and libraries and the integration of new more advanced ones. This upgrade enables the testing of different systems in various aspects.

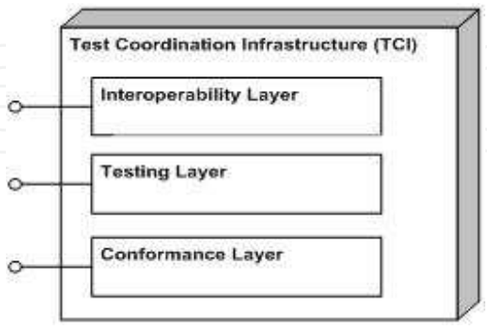


Fig. 6. Test Coordination Infrastructure

The general structure and functionality of the TCI is depicted in the Figure 6. Three fundamental layers, the *Testing Layer*, the *Interoperability Layer* and the Conformance Layer compose the TCI at a high level.

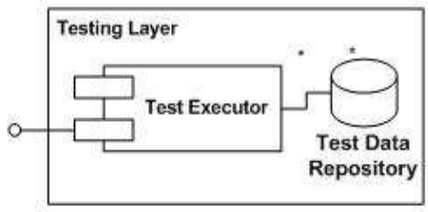


Fig. 7. Testing Layer

The Testing Layer (Figure 7) consists of the *Test Data Repository* and the *Test Executor*. The Test Executor orchestrates the execution of the deployed BPEL test cases in ICoM phases 3 and 4, based on a predefined schedule formed during these phases. Following this schedule, the Executor invokes sequentially the interfaces provided by the SuT Service Orchestration Engine in order to initiate the execution of the corresponding BPEL process. The invocation of the interfaces may require as input specific parameters (test data) which are retrieved from the Test Data Repository. As we will see in the following Section, the required test data rely on the nature of the deployed BPEL processes and are created along with the definition of the test cases.

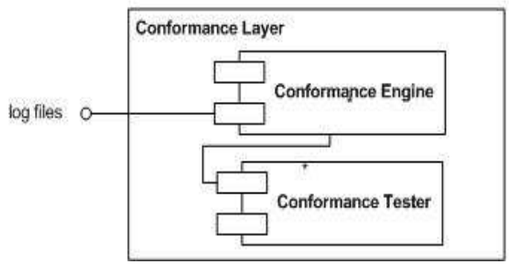


Fig. 8. Conformance Layer

The next layer of the TCI is the Conformance Layer (Figure 8) its main responsibility is to evaluate the conformance of each SuT against the adopted standards in phase 3. The evaluation process is performed based on the data produced by the SuT during the execution of the BPEL test cases. This layer contains two sub-components, the *Conformance Engine* and the *Conformance Tester*.

The Conformance Tester consists of a set of testing tools and libraries that actually assess the conformance of these data to the corresponding standards. The exact tools and libraries that should be adopted depend on the SuT's standards that have been defined during the formulation of a SuT (step 3). These range from tools that verify the conformance of a XML document to the corresponding XML schema and libraries that validate the security features of the XML documents to tools that check the conformance of the exchanged SOAP messages and message descriptions against the respective standards. Therefore, ICoM as methodology is quite extensible allowing the integration of new more updated tools/libraries in the Conformance Tester.

In the current phase, the steps that should be performed are the following:

- ✓ the identification and the integration of the required tools in the Conformance Tester,
- ✓ the implementation of the appropriate tools' interfaces that enable Conformance Engine to invoke them. In case that the required interfaces can not be implemented due to the

nature of the testing tools, a manual process is performed for the communication of these components.

In this layer, the Conformance Engine acts as the recipient and the analyzer of the log files that are produced by the SuTs in order to specify the tools that should be used for the evaluated data respectively.

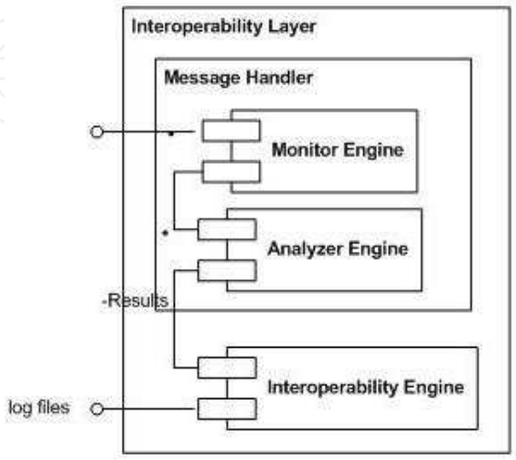


Fig. 9. Interoperability Layer

The Interoperability Layer (Figure 9) is the last but equally important layer comprising the *Message Handler* and *Interoperability Engine*. The Message Handler operates as the intermediate node during Interoperability testing (phase 4) that intercepts and delegates the exchanged messages to the SuTs. It comprises the Monitor Engine which intercepts the SOAP messages exchanged among the SuTs and the Analyzer Engine that determines whether these messages conform to their corresponding message descriptions.
The second subcomponent of the layer, the Interoperability Engine, is the actual interoperability consolidation point. It obtains as input the log files that are produced by the SuTs and the Analyzer Engine's conformance results providing them to the test operator. Based on these results, the test operator is able to do the following:

- ✓ notify the SuTs' interaction possibilities,
- ✓ detect the inaccurate points and
- ✓ impute the responsibility of the communication failure to the corresponding SuT.

Deductions are made by carrying out a process that consists of three main steps. The first one is to verify that the SuTs possess and handle the evaluated data without the detection of any erroneous behaviour. This means that the evaluated data are valid for all the SuTs. The second step includes a comparison process. During this process the exchanged data (e.g. exchanged SOAP messages and documents) are compared in order to acknowledge that the SuTs possess the same data. In the last step, the test operator confirms that the exchanged messages are part of the business processes that the SuTs have defined in their service descriptions. These steps enable the operator to identify whether the SuTs are interoperable and to what extent during the last phase of the proposed methodology.
In the following example we present the form of the TCI in our working example.

Example 3

The TCI has been shaped as presented in Figure 10. All the layer's components, Conformance Engine, Test Executor and Interoperability Engine have been implemented in Java technology. In the Testing Layer, a native XML Database, eXist has been adopted as the Test Data Repository enabling the storage of XML-based test data. The Conformance Tester of the Conformance Layer is shaped based on the SELIS and SWEB standards adopting testing tools that perform the conformance evaluation. Representative tools include testing environment and libaries for XML signature validation (IBM XML security suite (IBM XML), IAIK XML signature library (IAIK XML), IAIK XAdES toolkit (IAIK XAdES)) and tools for XML Schema document conformance (Altova XMLSpy).

The WS-I Tools (Brittenham, 2003), monitor (Brittenham et al., 2005) and analyzer (Brittenham, 2005), have been used in order to operate as the two subcomponents of the Interoperability Layer's Message Handler, Monitor and Analyzer Engine correspondingly. These tools provide an unobtrusive and automated way to log and analyze Web Service messages producing the respective conformance results that will be used for the interoperability evaluation.

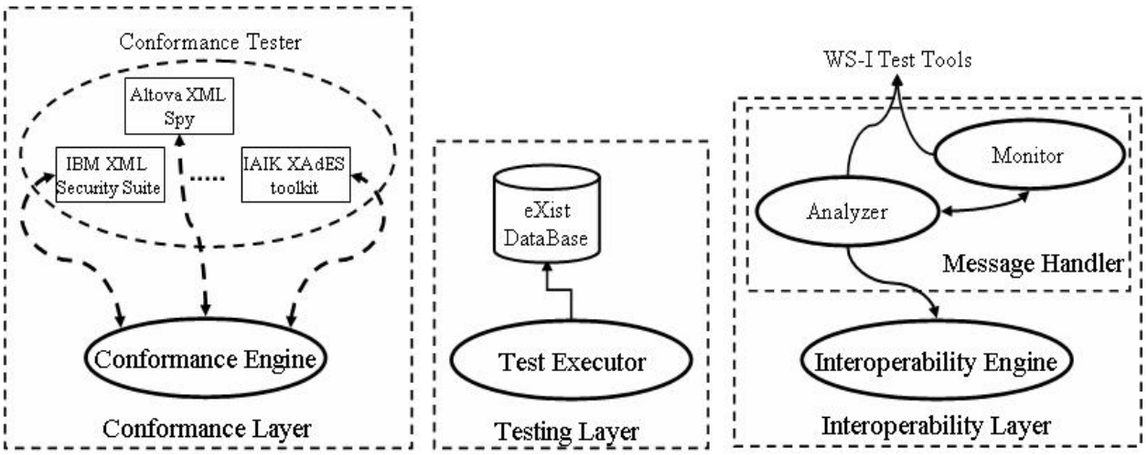


Fig. 10. TCI Working Example

Based on the above mentioned form of the TCI, the SELIS and SWEB SuT are able to be assessed taking into account the corresponding evaluation results that are derived from the execution of the defined test case in phase 3 and 4 of the proposed methodology.

3.3 Phase 3: Conformance Testing

Conformance testing has the primary goal of testing a WS against the standard it implements. This testing type involves a single SuT plus the TCI. The main steps that ICoM proposes are:

- *Definition of Conformance Test Cases*: In Section 0, we specified that ICoM adopts a BPEL representation of the deployed test cases. Our decision was based on the identification of a number of limitations (Pentafronimos et al., 2008) that the existing test case languages present and the nature and features of the BPEL language.
  Generally, BPEL allows the definition and the representation of specific business flows in a XML format. It has unique features in both syntax (e.g. flow with activity

synchronization, join condition) and semantics (e.g. dead-path-elimination) that make BPEL a highly compact and expressive language. In literature, there is intensive research on providing precise semantics for BPEL and verification of BPEL models (Fostre et al., 2006), (Xu et al., 2006). Additionally, there exist frameworks and models (Zheng et al., 2007), (Sinha & Paradkar, 2006), (Yuan et al., 2006), (Yan et al., 2006) that enable the automatic generation and definition of BPEL-based test cases.

Currently, most of the derived BPEL test cases are generated to test whether the implementation of a WS conforms to the BPEL behaviour and WSDL interface models. In ICoM, the BPEL processes specified and adopted depict activity flows that produce the appropriate evaluated data based on which the conformance and the interoperability capabilities of a WSuT are assessed.

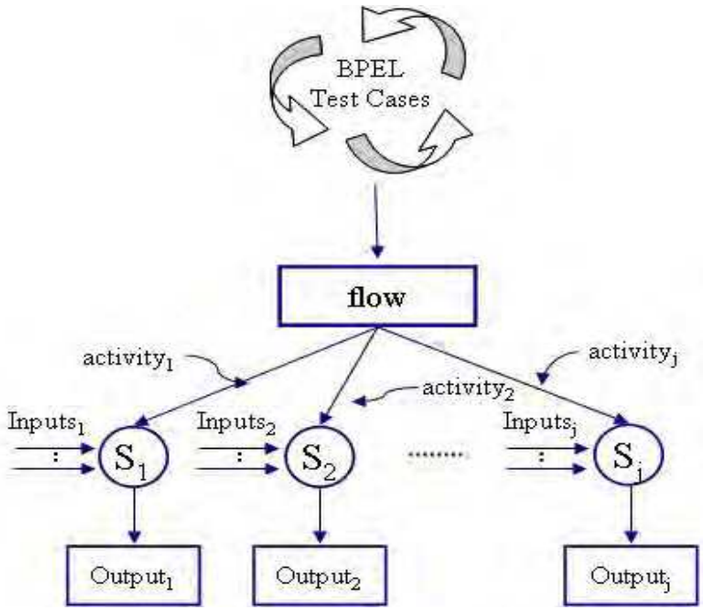


Fig. 11. BPEL Test Case

As depicted in Figure 11, a BPEL test case is a partially-ordered list of basic activities that that should be executed during a specific test run. The structural definition of a BPEL test case is as follows:

BPEL test case = {$Activity_j$, S_j, $Input_j$, $Output_j$}.

- $Activity_j$ is a set of activities that should be performed.
- S_j is a set of primary WSuT services that are invoked and orchestrated for the execution of a test case. Each service is associated with the realization of a specific activity.
- $Output_j$ is the result that each S_j derives. These results are gathered as the evaluated data of the SuTs.
- $Input_j$ is set of parameters that each service S_j requires according to the provided functionality in order to complete a defined process. The inputs can be:

- test data that are fed by the TCI during the initiation of a BPEL process,
- outputs of other primary services,
- parameters that have been defined and are included in the BPEL process.

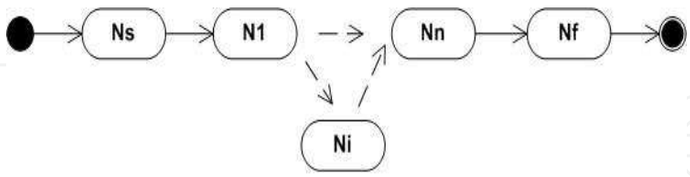


Fig. 12. Test Case Activity Graph Diagram

An activity graph diagram is also able to provide a visualization of the BPEL test cases (Figure 12). It illustrates and reflects the activity flow of the BPEL process in an effective manner. A test case activity diagram is composed by {N, E, N_s, N_f} where
N is a set of nodes {N_1, …, N_n} that correspond to the applied activities,
E is a set of edges {E_1, …, E_n} that are the implicit sequence concatenations as defined by the performed workflow,
N_s is the Start Node and N_f is the Final Node of the workflow.

In conformance testing, the designed (conformance) test cases include only internal activities. This means that merely the services of the examined SuT interact with each other in order to derive the evaluated data. Usually, in this type of test these data come from the final node of the test case.
The main objective of the current step focuses on the definition of the deployed BPEL processes based on the standards evaluated against. In this sense, the ICoM is quite adaptable since it is able to evaluate different aspects of the WSuT.
The test operator is able to create these processes utilizing existing BPEL test case generation frameworks or any other BPEL creation engine (ActiveBPEL, 2006). The processes are embedded in the SuT Service Orchestration Engine and the appropriate interfaces are implemented. During the BPEL generation phase, the required test data for each test case are specified and produced by the conformance evaluated tools of the TCI or by any other process.
Additionally, the test operator constructs a test case execution schedule taking into account the complexity of the applied processes. The schedule is a XML document that consists of the URLs of the BPEL processes interfaces and the corresponding required test data. This schedule along with the test data is stored in the Test Data Repository of the TCI. The steps that follow present and include the actual conformance testing sequence of the proposed methodology.

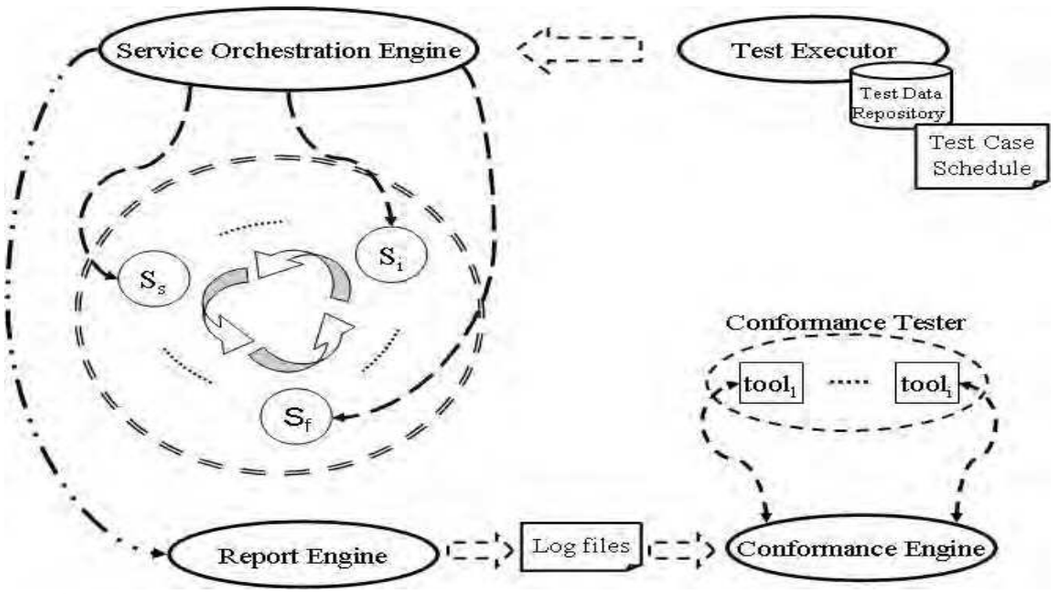


Fig. 13. Conformance Testing suite

- *Execution of test cases*: In this step, the TCI Test Executor automatically initiates the testing procedure by invoking the interface of the Service Orchestration Engine following the execution schedule. This action triggers the execution of the corresponding BPEL process running the defined test case. In Figure 13, the S_s is the initial service that is invoked and S_i is a set of services that contribute to the derivation of the evaluated data by the S_f.
- *Collection of results*: At the end of the test case, the Service Orchestration Engine has accumulated the evaluated data that are produced as outputs by the involved service of the WSuT. The Report Engine collects and consolidates these data in log files which are prepared for evaluation.
- *Results analysis*. Initially, the Conformance Engine analyzes the log files and extracts the produced data. Then, the Engine specifies the tools (tool1, … ,tooli) of the Conformance Tester that should be used based on the nature of these data. The Engine feeds the data via the implemented interfaces to the corresponding tools, which in turn infer the conformance of the implementation to the respective standard. Suggestions concerning any corrections to the implementation are also provided to the SuT, enabling the deployment of a fine-tuning process that will correct discrepancies.
- *Corrective actions*: Corrective actions by the WSuTs' developers may include both re-design and re-implementation or only updates of specific services within the WSuT.
- *Re-execution of failed tests*: When the corrective actions are complete, the SuT undergoes a new test round to check if the previously failed tests are now successfully passed.

At the end of the last step, the process begins again from "Execution of test cases" step in subsequent iterations until all tests are successful.
In the example that follows, it is illustrated an instance of the execution of the current phase's steps in our demonstrated case study.

Example 4

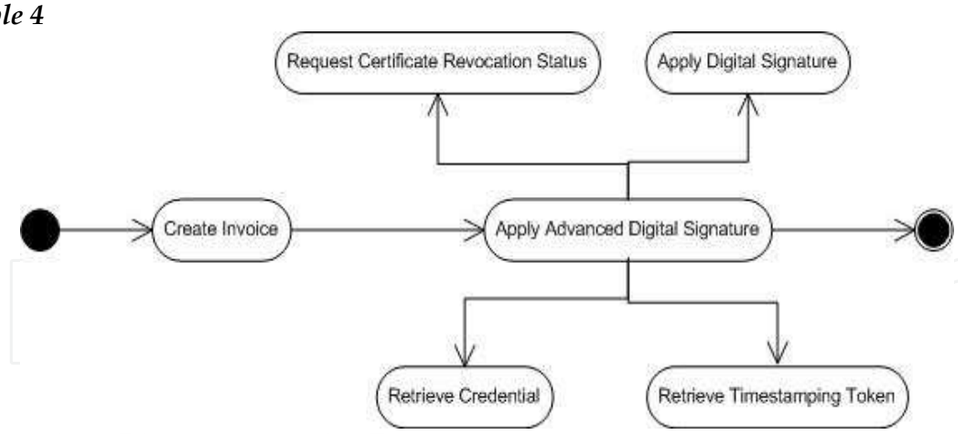


Fig. 14. Activity Diagram of the XAdES conformance test case

Figure 14 depicts an activity diagram representing an example test case for the conformance testing of the SELIS XAdES implementation against the XAdES standard. The actions that compose the test case are the following:

- ✓ creation of a XML Invoice Document based on the xCBL standard,
- ✓ application of Advanced Digital Signature on the produced document. This process includes four separate sub-processes which occur transparently:
  - o certificate retrieval of the certificate that will be used for the signature process,
  - o digital signing of the invoice and certain other properties according to the W3C XML Digital Signature standard,
  - o time stamping by requesting, and embedding a time stamp token on the generated signature according to the IETF 3161 standard, and finally
  - o revocation information inclusion by embedding certificate revocation status data after requesting them on-line from the server of the Certification Authority that has issued the signer's certificate.

The above test case is to be used as follows: initially the TCI Test Executor initiates the execution of the corresponding BPEL process of the SELIS Service Orchestration Engine invoking the appropriate interface. The choreography of the required WSuT services derives an xCBL invoice document signed according to the XAdES standard as illustrated in Figure 15. The Report Engine embeds the signed document to a log file which is retrieved by the TCI's Conformance Engine. The Engine delegates the document to the integrated IAIK toolkit to validate the applied signature.

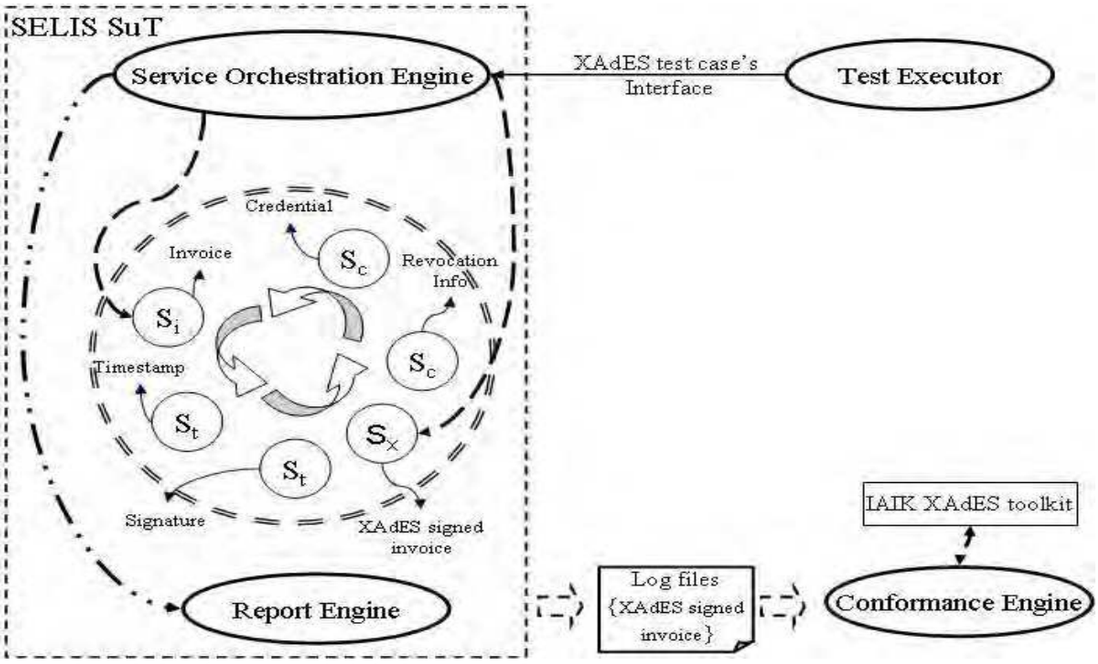


Fig. 15. SELIS Conformance XAdES Testing Suite

During our testing effort, in the actual first execution of this test case the toolkit indicated an incompatibility of the produced signature to the corresponding XML schema. This discrepancy was pointed out to the developers of the SELIS-invoicing which performed the necessary corrections. The re-execution of the test case denoted the conformance of SELIS with the XAdES standard.

3.4 Phase 4: Interoperability Testing

Interoperability testing is a more complex process than conformance testing because it usually involves at least two SuTs wishing to intercommunicate (Figure 16). The interoperability steps proposed by ICoM are not different from the corresponding conformance steps described in the previous section with regards to the logic and the sequence of the performed steps. On the contrary, the objective goals and the operation of these steps present significant differentiations. Thus, interoperability testing includes:

- *Definition of Conformance Test Cases*: The nature and the structure of the (interoperability) test cases that are designed and used in this phase are almost similar with the conformance ones, presented in Section 0. These test cases apart from internal activities include also and external activities. This means that the primary services of a SuT do not interact only with each other, but also with the services of other SuTs (Figure 16). Thus, the complexity of the BPEL processes that should be adopted to represent the logic of a test case is increased at a significant level.
  The definition of the interoperability test cases takes into consideration two parameters. The first one is the testing scenario and the role assignment to the SuTs which was defined in the phase I of ICoM. The second parameter is the standards that the SuTs have adopted and are evaluated against. The specified interoperability test cases should

constitute instance of this scenario taking into account the variants of this scenario based on the adopted standards.

BPEL processes should be formulated merely for the SuTs that initiate the execution of a testing suite. For example, the specified scenario, as depicted in Figure 16, is that the "SuT A" interacts with the "SuT B" retrieving a response. Therefore, based on a defined test case, the "WSuT A" orchestrates its corresponding services e.g. As and Ak to execute a specific process that enables service Ai to communicate with the service Bj of the "WSuT B" expecting a reply which is handled it appropriately. The BPEL process that corresponds to the above test case includes merely the actions which are executed by the "WSuT A". "WSuT B" reacts to this interaction creating the reply according to the logic that is included in the implementation.

Identically to the conformance testing, the BPEL processes are created via BPEL generation framework or any other available BPEL engine and are embedded in the Service Orchestration Engine of the respective SuT. The execution of the test cases is defined by a testing schedule that is prepared.

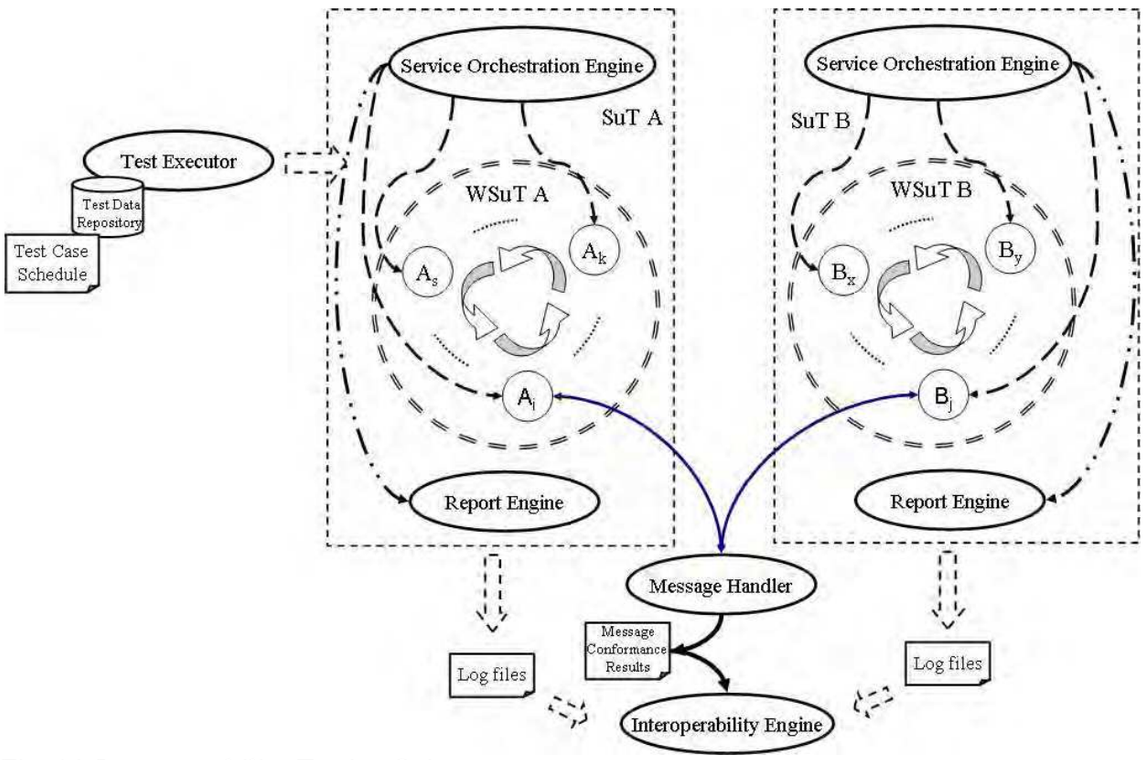


Fig. 16. Interoperability Testing Suite

- *Execution of test cases*: As presented in Figure 16, this step is completed following a similar process with the corresponding of the conformance testing. The execution of the external interactions among the involved WSuTs introduces an additional factor that should be taken into account and concerns the conformance of the exchanged messages to the defined business process. Thus, the TCI's Message Handler participates in the external interactions intercepting and analyzing the exchanged messages, and checking that they belong to the proper message domain.

- *Collection of results*: The Service Orchestration Engines of the SuTs taken part in the test case accumulate the whole of the evaluated data that are produced by all the involved WSuTs' services. Then the corresponding Report Engines collect and consolidate them in log files.
- *Results analysis*. The test results are analyzed and conclusions are made on the interoperability capabilities of the SuTs, as presented in Section 0.
- *Corrective actions*: Corrective actions may include both re-design and re-implementation or only updates of specific areas within the SuT.
- *Re-execution of failed tests*: As soon as the corrective actions have finished, the SuTs undergo a new test round to check if the previously failed tests are now successfully passed.

At the end of last step, the process begins again from second one in subsequent iterations until all tests are successful. In the Example 5 that follows the operation of the interoperability testing is described in detail.

Example 5

Figure 17 represents the activity diagram of an interoperability test case that is an instance of our working demonstration scenario, as implemented by the SELIS and SWEB SuTs. This test case includes only the actions performed by SELIS which is the actual initiator of the testing procedure.

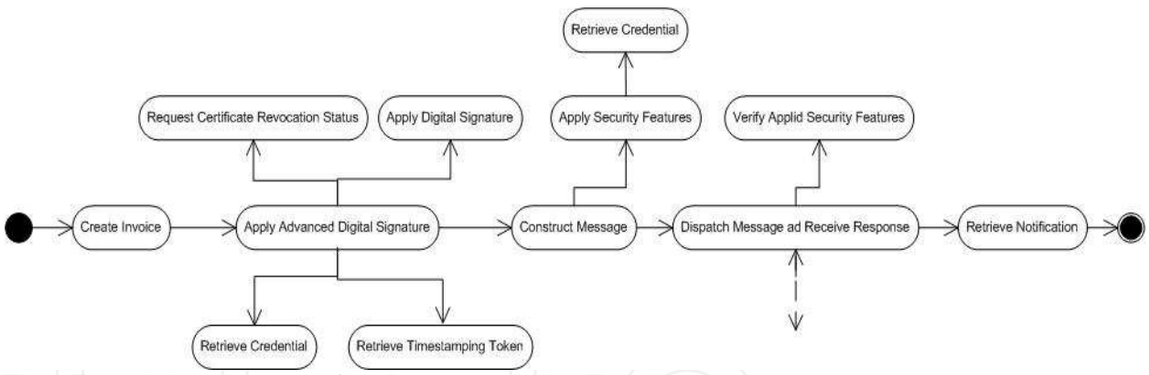


Fig. 17. Activity diagram of an Interoperability Test Case

As depicted in Figure 18 the process begins from the service A_1 of SELIS WSuT that creates a xCBL invoicing document (I_1). The document is signed by service A_2 according to the XAdES standard (SI_1) gathering the required time stamps and related certificate revocation status information data from their respective sources. Service A_4 packages the signed invoice in a SOAP message (M_1) where the WS Security features are applied (SM_1) using the service A_3. Finally, the service A_5 dispatches the message to the to the SWEB.

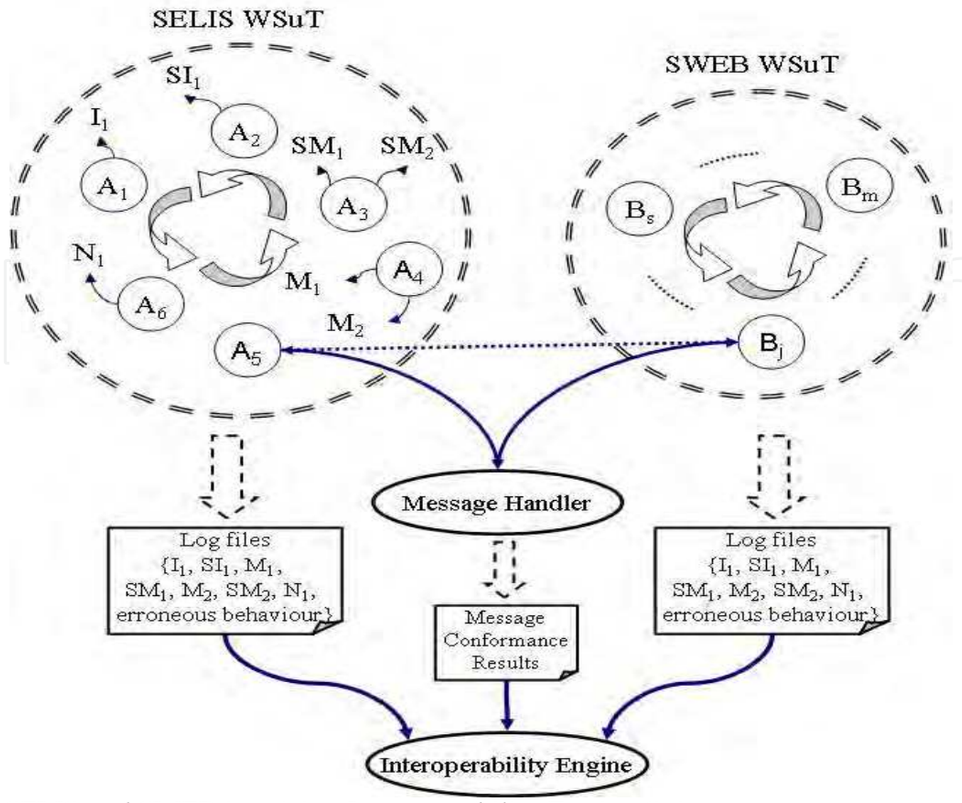


Fig. 18. SELIS and SWEB-invoicing Interoperability Testing Suite

SWEB handles the received SOAP message according to its predefined internal processes without following any steps specified by the test operator. Thus, SWEB receives the SOAP message, decrypts it and verifies the WS Security features. Then, the invoice document is extracted and the XAdES signature is verified (along with the rest of the cryptographic information it encompasses like any time stamps). A notification (N_1) is created that is packaged in a new SOAP message (M_2) where the WS-Security extensions are applied (M_2) and is sent to SELIS.

The latter as soon as receives the response, handles it according to the test case actions. Service A_5 retrieves the SOAP message and using the service A_3 validates the applied security features. Finally, the service A_6 manages the notification.

It should be noted that, the exchanged SOAP messages (SM_1 and SM_2) are intercepted by the TCI Message Handler. The Handler analyzes them checking that these messages conform to the defined SuTs' WSDL documents and produces the appropriate conformance results. The SuT Report Engines of both SuTs collect all the evaluation results and import them in log files. These consist of the exchanged SOAP messages (M_1, SM_1, M_2, and SM_2), the exchanged documents (I_1, SI_1, N1) and any erroneous behaviour that was reported during the whole process e.g. the application and validation of the applied security features of the invoice document.

The TCI Interoperability Engine receives the SuTs' log files and the Message Handler conformance results that are provided to the test operator which is responsible to analyze

them and indicate the SuTs communication capability. The analysis we have performed in our demonstration systems verified that:

- ✓ the SuTs possess the same documents (I_1, SI_1, N_1) and messages (M_1, SM_1, M_2, SM_2) that were created either form the one or form the other system,
- ✓ no erroneous behavour has been detected during the interaction e.g. the validation of all message and document security features was successful,
- ✓ the exchanged messages (SM_1, SM_2) are part of the SuTs' business processes.

Therefore, according to the abovementioned analysis the two systems are interoperable.

4. Conclusions and Future Work

The creation of an interoperability framework is based on two factors, the precise definition of the used standards and the specification of a strict set of guidelines that should be followed. The testing methodologies are able to guarantee the development of this framework. Nevertheless, the existing testing methodologies and frameworks are not able to cover all interoperability aspects of a Web Services-based environment.

This paper presents an enhanced well-formed testing methodology named ICoM, which is able to test and guarantee the end-to-end application interaction capabilities of WS-based services offered autonomously or via a SOA. ICoM achieves to overcome the weaknesses of the existing methodologies being able to satisfy various requirements. ICoM is based on the principles of existing interoperability methodologies and frameworks adopting widely used testing types, covering a set of interoperability and conformance aspects of the WSs and finally formulating an adaptable, extensible and flexible framework. ICoM was demonstrated using an e-invoicing service which is offered as an autonomous WS (SELIS) and via the SOA-based platform SWEB.

Future work in this area includes the integration of new types of testing such as integration, performance and stress testing in the proposed methodology. These new testing types will enrich ICoM allowing to test various dimensions of the evaluated services and enabling the creation of a more integrated and enhanced testing framework.

5. Acknowledgment

This work has been supported by the GSRT (PENED) programme and the IST project SWEB (IST-2006-2.6.5). The authors would like to thank all the participants for valuable discussions and the European Union for funding the SWEB project.

6. References

High, R.; Kinder, S. & Graham, S. (2005). *IBM's SOA Foundation – An architectural Introduction and Overview*.

European Interoperability Framework for Pan-European eGovernment Services (EIF), available at http://europa.eu.int/idabc/

Kaliontzoglou, A.; Boutsi, P. & Polemi, D. (2006). eInvoke: Secure e-Invoicing based on Web Services. *Electronic Commerce Research*, vol 6, Numbers 3-4, pp. 337-353, Springer.

Papastergiou, S.; Karantjias, A. & Polemi, D. (2007a). Innovative, Secure and Interoperable E/M-Governmental Invoicing. 1*8th Annual IEEE International Symposium on Personal, Indoor and Mobile Radio Communications (PIMRC)*, Athens.

Secure, interoperable, cross border m-services contributing towards a trustful European cooperation with the non-EU member Western Balkan countries (SWEB), IST-2006-2.6.5, 6th Framework Programme, Priority 2, Information Society Technologies (2007), www.sweb-project.org/.

OSI - Open System Interconnection, Conformance testing methodology and framework (1997), ISO/IEC 9646.

European Telecommunications Standards Institute (ETSI), available at http://portal.etsi.org/mbs/Testing/testing.htm

Moseley, S.; Randall, S. & Wiles, A. (2003). Experience within ETSI of the combined roles of conformance testing and interoperability testing. *Standardization and Innovation in Information Technology*, pp 177- 189, IEEE Press.

ETSI 300 406 (1995). Methods for testing and Specification (MTS); Protocol and profile conformance testing specifications; Standardization methodology.

ETSI ES 201 873 (2002): The Testing and Test Control Notation version 3, v2.2.0.

Dibuz, S. & Kremer, P. (2003). Interoperability Testing - Framework and Model for Automated Interoperability Test and Its Application to ROHC. *TestCom 2003*, LNCS vol 2644, pp. 243–257, Computer Science.

ETSI EG 202 237 V1.1.2 (2007-04): Methods for Testing and Specification (MTS); Internet Protocol Testing (IPT); Generic approach to interoperability testing.

Kulvatunyou, B.; Ivezic, N.; Martin, M. & Jones, A.T. (2003). A Business-to-Business Interoperability Testbed: An Overview. *The Fifth International Conference on Electronic Commerence*, Pittsburg, PA October.

Seely, S. & Lauzon, D. (2005). *WS-I monitor tool functional specification*, ver. 1.1. Technical report, WS-I.

Ehnebuske, D. et al. (2003). *WS-I Overview*. Available at www.ws-i.org/docs/20021003.wsi.introduction.pdf

ebXML Technical Architecture Project Team, (2001). ebXML Technical Architecture Specification v1.0.4, February 16.

OASIS ebXML Implementation, Interoperability and Conformance Technical Committee:, (2003). ebXML Test Framework v0.3, 07 March.

Lee, Y. (2005). ebXML Test Framework on Dually Coupled Asynchronous MSH. *Fourth Annual ACIS International Conference on Computer and Information Science (ICIS'05)*, pp. 198-203, IEEE Press.

Kim, D. & Yun, J.H. (2003): Development -of an ebXML Conformance Test System for e-Business Solutions. *EC-Web 2003*, LNCS, vol 2738, pp. 145-154, Springer-Verlag Berlin Heidelberg

IBM XML security suite, available at http://www.alphaworks.ibm.com/tech/xmlsecuritysuite.

IAIK XML signature library, available at http://jce.iaik.tugraz.at/sic/products/xml_security.

IAIK XAdES library, http://jce.iaik.tugraz.at/sic/products/xml_security/xades

Papastergiou, S.; Kaliontzoglou, A. & Polemi, D. (2007b) Interoperability issues of a Secure ELectronic Invoicing Service (SELIS). *18th Annual IEEE International Symposium on Personal, Indoor and Mobile Radio Communications (PIMRC 2007)*, 3-7 September, Athens.

ETSI Technical Report 266 (1996): Methods for testing and Specification (MTS); Test Purpose style guide.

Saglietti, F.; Oster, N. & Pinte, F. (2008).White and grey-box verification and validation approaches for safety- and security-critical software systems. Information Security Technical Report, Vol. 13, *Elsevier Advanced Technology*, ISSN 1363-4127, p. 10 – 16.

Peyton L.; Stepien, B. & Seguin, P. (2008). Integration Testing of Composite Applications, *Proceedings of the Proceedings of the 41st Annual Hawaii International Conference on System Sciences*, IEEE Computer Society.

Rizwan M.; Mamoon Y., (2007). SOA Testing using Black, White and Gray Box Techniques, White paper Crosscheck Networks, available at http://www.crosschecknet.com/resources/articles/soatesting-v2.htm.

Brittenham, P. (2003). Understanding the WS-I Test Tool, available at http://www-128.ibm.com/developerworks/webservices/library/ws-wsitest/.

Brittenham, P.; Durand, J.; Kleijkers, L. & Stobie, K. (2005a). WS-I Monitor Tool Functional Specification, WS-I Testing Work Group.

Brittenham, P., et al. (2005b). WS-I Analyzer Tool Functional Specification WS-I Testing Work Group.

Pentafronimos G.; Papastergiou S. & Polemi D. (2008). Definition and representation of test cases for e-government Web Services, *2nd International Conference on Theory and Practice of Electronic Governance (ICEGOV2008)*, 1 - 4 December, 2008, Cairo, Egypt.

Fostre, H.; Uchitel, S.; Magee, J. & Kramer, J. (2006). Model based analysis of obligations in web service choreography, *Proc. of AICT-ICIW*. IEEE Computer Society.

Xu, K.; Liu, Y. & Pu, G., (2006). Formalization, verification and restructuring of bpel models with pi calculus and model checking, IBM, Tech. Rep.

Zheng, Y.; Zhou, J. & Krause, P., (2007). An Automatic Test Case Generation Framework for Web Services. *Journal of Software*, vol. 2(3), pp. 64-77.

Sinha, A. & Paradkar, A. (2006). "Model based functional conformance testing of web services operating on persistent data," in Proc. of TAV-WEB. ACM Press, pp. 17–22.

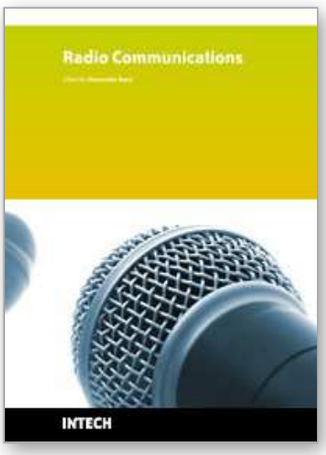
Yuan, Y.; Li, Z. & Sun, W., (2006). A graph-search based approach to bpel4ws test generation, *Proc. of ICSEA*. IEEE Computer Society, p. 14.

Yan, J.; Li, Z.; Yuan, Y.; Sun, W & Zhang, J. (2006). Bpel4ws unit testing: Test case generation using a concurrent path analysis approach, *Proc. of ISSRE. IEEE Computer Society*, 2006, pp. 75–84.

ActiveBPEL Engine Architecture (2006), available at http://www.activebpel.org/docs/architecture.html.

Karantjias A.; Papastergiou S. & Polemi D. (2008). Holistic Electronic & Mobile Government Platform Architecture. *8th Joint Conference on Knowledge - Based Software Engineering 2008 (JCKBSE 08)*, IOS Press, August 25-28, Athens, Greece.

Papastergiou, S.; Polemi, D. & Douligeris, C., (2009). SWEB: An advanced mobile Residence Certificate Service, *3rd International Conference on e-Democracy "Next Generation Society: Technological and Legal Issues"*, Lecture Notes of ICST (LNICST), Springer, ISBN 978-963-9799-54-7, Athens, Greece.



Radio Communications

Edited by Alessandro Bazzi

ISBN 978-953-307-091-9
Hard cover, 712 pages
Publisher InTech
Published online 01, April, 2010
Published in print edition April, 2010

In the last decades the restless evolution of information and communication technologies (ICT) brought to a deep transformation of our habits. The growth of the Internet and the advances in hardware and software implementations modified our way to communicate and to share information. In this book, an overview of the major issues faced today by researchers in the field of radio communications is given through 35 high quality chapters written by specialists working in universities and research centers all over the world. Various aspects will be deeply discussed: channel modeling, beamforming, multiple antennas, cooperative networks, opportunistic scheduling, advanced admission control, handover management, systems performance assessment, routing issues in mobility conditions, localization, web security. Advanced techniques for the radio resource management will be discussed both in single and multiple radio technologies; either in infrastructure, mesh or ad hoc networks.

How to reference

In order to correctly reference this scholarly work, feel free to copy and paste the following:

Spyridon Papastergiou and Despina Polemi (2010). A Testing Process for Interoperability and Conformance of Secure Web Services, Radio Communications, Alessandro Bazzi (Ed.), ISBN: 978-953-307-091-9, InTech, Available from: http://www.intechopen.com/books/radio-communications/a-testing-process-for-interoperability-and-conformance-of-secure-web-services

INTECH
open science | open minds

InTech Europe
University Campus STeP Ri
Slavka Krautzeka 83/A
51000 Rijeka, Croatia
Phone: +385 (51) 770 447
Fax: +385 (51) 686 166
www.intechopen.com

InTech China
Unit 405, Office Block, Hotel Equatorial Shanghai
No.65, Yan An Road (West), Shanghai, 200040, China
中国上海市延安西路65号上海国际贵都大饭店办公楼405单元
Phone: +86-21-62489820
Fax: +86-21-62489821

IntechOpen