Integrated Design and Process Technology, **IDPT-2006**
Printed in the United States of America, June, 2006

Import of WSDL Definitions in TTCN-3 Targeting Testing of Web Services *

Ina Schieferdecker[1,2], Diana-Elena Vega[2], Cosmin Rentea[1]
{schieferdecker, vega, rentea}@fokus.fraunhofer.de
[1] Fraunhofer FOKUS, Kaiserin-Augusta-Allee 31, 10589 Berlin, Germany
[2] ETS, TU Berlin, Franklinstr. 28-29, 10587 Berlin, Germany

ABSTRACT:
In the context of interoperability testing, TTCN-3 (Testing and Test Control Notation), already successfully used in testing middleware such as CORBA, CCM, EJB, matches perfectly in the field of Web services. This paper targets the development of TTCN-3 test suites for Web services directly derived from WSDL interfaces. A mapping between WSDL data descriptions to TTCN-3 data is presented to enable the automated derivation of language artefacts and of basic test scenarios. These mapping rules and the the implementation support are evidenced by means of a case study.

I. Introduction

Testing of Web services is useful to prevent late detection of errors, which typically requires complex and costly repairs. An automated test approach helps in particular to efficiently repeat tests whenever needed for new system releases in order to assure the fulfilment of established system features in the new release.

In the context of automated system testing it is necessary to consider as much as possible the available data types definitions exposed by the System under Test (SUT) during the test development phase in order to avoid superfluous redefinition of such data structures. The aim of easily addressing Web services descriptions expressed in standardized languages such as WSDL (Web Services Description Language) is of particular importance.

The Testing and Test Control Notation (TTCN-3, [1]) constitutes a good candidate for systematic, specification-based testing providing adequate language artifacts for mapping and handling WSDL definitions. This paper is meant to make an introduction into Web services testing and present how a mapping from WSDL to TTCN-3 could help automating the effort of testing when TTCN-3 is employed to attain this goal. We expose different approaches and techniques that have been already considered in related papers. Throughout a case study, we present our approach to generate TTCN-3 test scripts out of WSDL.

The paper is structured as follows: section two presents the Web services technologies insisting on the WSDL language used for exposing Web services. The next section introduces the TTCN-3 language and its test system architecture. After highlighting the general approaches in testing Web services, an insight of previous attempts to use TTCN-3 for Web services testing and importing languages into TTCN-3 is given in section four. Thereafter, our approach of importing WSDL definitions into TTCN-3 is detailed in the fifth section. Eventually the conclusions are presented and further possible extensions to this approach are outlined.

II. Presentation of Web Services

A Web service is an interface that describes a collection of operations that are network accessible through standardized XML messaging. It is described using a standard, formal XML notion, called its service description. Service description covers all the details necessary to interact with the service, including message formats (that detail the operations), transport protocols and location. The service description is published in a service registry commonly known as UDDI registry. The registry enables service consumers to find services that match their needs.

One of of the most advantages of Web services is that they can be developed using any programming language and can be deployed on any platform. Therefore, the platform and implementation language is not an obstacle for the client of the service.

Web services communicate over standard Web protocols using XML interfaces and XML messages, which any application can interpret. But XML by itself does not ensure effortless communication. The applications need standard formats and protocols that allow them to properly interpret the XML. Thus, three XML-based technologies could be considered as de-facto standards for Web services: Simple Object Access Protocol (SOAP, [11])- standard communication protocol for Web services, Web services Description Language (WSDL, [6]) - standard for formally describing Web services, Universal Description, Discovery and Integration (UDDI, [10]) - standard mechanism to register and discover Web services.

The Figure 1 illustrates the well-known architecture model for developing Web services that emphasizes also some ideas about various aspects of Web services testing: the discovery of Web services, the data format exchanged

* *This work has been supported by the "Krupp von Bohlen und Halbach - Stiftung"*

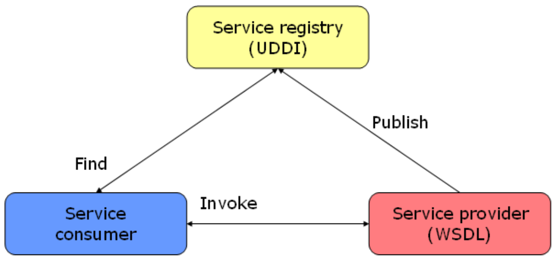

Fig. 1. The Web Services Architecture

(i.e. WSDL) and the request/response mechanisms (i.e. SOAP).

WSDL [6] is an XML grammar for describing Web services, an interaction interface between the service requestor and the service provider. The specification itself is divided into following major elements:

definitions - the root element of all WSDL documents where the name of the Web service, namespaces used are introduced

types - a container for data type definitions used between the client and the server. WSDL is not tied exclusively to a specific typing system, but it uses the W3C XML Schema specification as its default choice

message - an abstract, typed definition of the data being communicated; it defines the name of the message and contains zero or more message *part* elements

portType - an abstract set of *operations* supported by one or more endpoints; an operation is an abstract description of an action supported by the service

binding - a concrete protocol and data format specification for a particular port type (how the messages are transmitted on the wire, which are the SOAP-specific details)

service - a collection of related endpoints; most commonly it defines the address for invoking the specified service, including an URL for invoking the SOAP service; it contains multiple *ports* - single end points defined as a combination of a binding and a network address

In our implementation, the WSDL 1.1 specification was preferred over WSDL 2.0 because the latter is still under development at W3C (”Candidate Recommendation”) and the tools compliant to WSDL 2.0 (such as Apache Axis2) are also not mature enough for production use. Nonetheless, the importing mechanism can be easily adjusted to the forthcoming standard.

Nowadays, a debated concept in the context of Web services topic is Service Oriented Architecture (SOA): ”a set of components which can be invoked, and whose interface descriptions can be published and discovered”(W3C). Mainly it covers aspects as compositions of Web services (known as *orchestration* or *choreography*) which form the messaging backbone for enterprise communication. Tied to this architectural approach and to the need for creating complex business processes, software vendors have converged around the Business Process Execution Language(BPEL [12]) for high level description of services workflows.

III. Overview of TTCN-3

TTCN-3 is the standard testing language designed to address testing needs of modern telecom and datacom technologies and widen the applicability to many kinds of tests including interoperability, system, integration and performance testing. TTCN-3 enables to systematic, specification-based testing for various kinds of tests including functional, scalability, load, inter-operability, robustness, regression, system and integration testing. It allows an easy and efficient description of complex test behaviors in terms of sequences, alternatives, and loops of stimuli and responses [7].

TTCN-3 offers different constructs to describe the test data: types, templates, variables, procedure signatures etc. They can be used to express any SOAP messages and WSDL types and operations. The structured types like record, record of, set, set of, enumerated and union enable bindings from other languages. Templates are data structures used to define message patterns for the data sent or received over ports. They are used either to describe distinct values that are to be transmitted over ports or to evaluate if a received value matches a template specification. The communication between test components as well as the one between test components and test system interface are realized over ports. Relevant to WSDL are procedure-based ports which implement synchronous communication paradigm. Data transmission directions can be defined for each port: *in* (the received data), *out* (the sent data), *inout* (the data can be both sent or received).

TTCN-3 permits importing of data types from other languages than TTCN-3 like: ASN.1, IDL, XMLSchema [5] and now WSDL (as shown in Figure 2). The core language can be used independently of the presentation format; thus it is adequate that the mapping from WSDL to TTCN-3 focuses on the TTCN-3 core notation only.

A TTCN-3 test system defines in ATS (abstract test system) the SUT in terms of type definitions and interactions interfaces, which are usually named the internal representation of the SUT model in the ATS. With respect to Web services, the WSDL interfaces can be modeled in TTCN-3, thus the WSDL is seen as a model representation of the SUT. On top of this model, TTCN-3 is able to define data templates for interaction with SUT. The communication with SUT is continuously validated against this model, any non-conformance to it being reported as mismatch or invalid behavior.

IV. Testing of Web Services

Basically, Web services testing levels cover *unit testing* during their development, *functional testing* for ensuring that the functionality is as expected, *security/authorization testing* which refers to the protection of messages according to partners’ agreements and policies. Other test types are

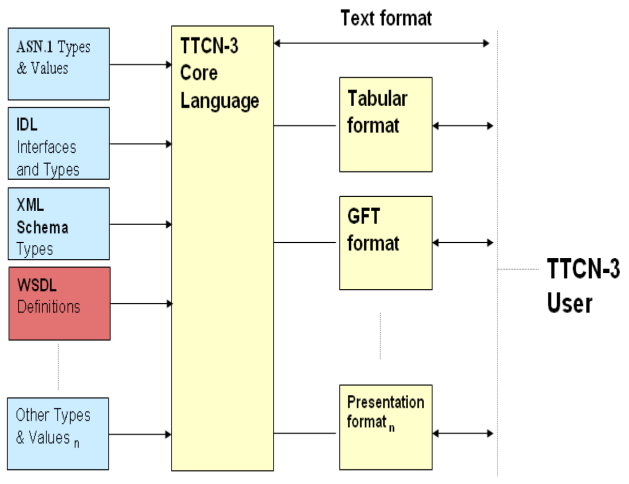

Fig. 2. TTCN-3 Language Extensibility

performance, stress, load testing and verification for *concurrency and synchronization*.

The aspects to be verified in order to assure the correctness of a Web service have been included in standardization committees activities. Web services interoperability profiles WS-I (Basic Profile, WS-I Basic Security Profile, [9]) have been published along with the Monitor and Analyzer tools.

In [14] it is considered that WSDL does not offer enough information with respect to the testing purpose; four language extensions are proposed so that test scripts could be automatically generated. In [4] an approach of automatic conformance testing for validating Web Services as a condition before allowing their registration has been proposed. By introducing a Meta language in XML for describing test cases, in [17], an approach for automatic testing for SOAs has been proposed. It has been demonstrated by implementing a prototype whose design comprises concepts as test agents, online testing, test daemon, batch testing; test strategies, performance and reliability aspects were discussed.

Specification-based automated testing, where abstract test specifications independent of the concrete system to be tested and independent of the test platform are used, are superior to previous approaches: they improve the transparency of the test process, increase the objectiveness of the tests, and make test results comparable. This is mainly due to the fact that abstract test specifications are defined in an unambiguous, standardized notation, which is easier to understand, document, communicate and to discuss. In this context, TTCN-3 – the only standardized testing language – is nominated as the natural choice.

In [18] a TTCN-3 abstract testing approach is proposed for Web services, which distributes the test activities to both server and client sides, thus facilitating testing while the traditional approaches fail due to the difficulties brought by the language and platform-independence of Web services. Nevertheless, this approach lacks generality as it is tailored toward specific Web services even in the abstract test suite, and the adaptation layer can hardly be adapted to new real-world services. The binding between Web service definition and the TTCN-3 system is not automated after the test case design phase.

A test framework for Web services with XML/SOAP interfaces using TTCN-3 is presented in [16]; this framework aims to provide functional, service interaction, and load tests. This is the first proposal of mapping DTD or XML Schema types to TTCN-3 definitions by type extraction and then type translation. DOM enables to construct a new tree of XML definitions which can then then be parsed for the translation step. Yet, the generation of TTCN-3 definitions other than types and of test data are not properly considered.

Further work on importing XML Schema types together with a complex case study from the automotive industry were presented in [15]; realistic tool implementation ideas were also included which were beneficial to the current endeavor. The mapping described above became the foundation of the *”Part 9: The XML to TTCN-3 Mapping”* section from the latest TTCN-3 specification.

With a different perspective, [8] describes an automated TTCN-3 test framework whose purpose is easing the creation and deployment of distributed functional and load tests. The test framework is exemplified for Web service tests and presents results obtained from testing a specific Web service.

However, we go beyond classical approaches towards specification-based automated testing, which until now mainly concentrated on the automated test implementation and execution: we consider test generation aspects as well as the efficient reuse of test procedures in a hierarchy of tests. Testing of Web services has to target three aspects: the discovery of Web services (i.e. UDDI being considered here), the data format exchanged (i.e. WSDL), and request/response mechanisms (i.e. SOAP). The data format and request/response mechanisms can be tested within one test approach: by invoking services using SOAP requests and observing responses with test data representing valid and invalid data formats or values.

Since a Web service is a remote application, which will be accessed by multiple users, not only functionality in terms of sequences of request/response and performance in terms of response time, but also scalability in terms of functionality and performance under load conditions matters. Therefore we have developed a hierarchy of test settings starting with separate functional tests for the individual services of a Web service, to a service interaction test checking the simultaneous request of different services, to a separate load tests for the individual services up to a combined load test for a mixture of requests for different services.

In the next section, we will discuss our idea of importing WSDL information into TTCN-3, which is deemed to cope with some of the above issues.

V. Generating TTCN-3 definitions from WSDL

Language bindings to XML (software mechanisms that transform XML data into values that programmers can access and manipulate from their language of choice) can be built upon the following standard methods: SAX (Simple API for XML) or DOM (Document Object Model). Other examples of language-specific XML bindings are: JAXB, WSDL2C, WSDL2Java. Every such code generation process from a language to another is built on a set of mapping rules.

The import rules for WSDL

Considering the WSDL structure and the TTCN-3 test system requirements for the interaction with SUT, a set of WSDL elements was selected and mapped into TTCN-3. The Table I illustrates the associations between WSDL and TTCN-3 that were envisioned.

TABLE I
The Mapping Rules between WSDL and TTCN-3 Definitions

WSDL Definition	WSDL Tag	TTCN-3 Definition
XML Schema Type	<types>	TTCN-3 Type declaration, simple or complex (e.g. set, record, union)
Message part	<part>	Signature parameter or return type
Message	<message>	Set of *in* or set of *out* parameters of a signature
Fault	<fault>	Exception
Operation	<operation>	Signature
Port Type	<portType>	Group of one TTCN-3 port type and signatures for the contained operations
Binding	<binding>	Record of constants passed to the Adapter (e.g. protocol, transport, style, encoding)
Service	<service>	Component Type with TTCN-3 ports correlated to each enclosed <*port*> instance; for each such port, a group of basic testcases corresponding to its operations
Definitions	<definitions>	Module

The selected elements to be mapped are split in two categories: abstract service descriptions and invocation details.

Every built-in XML Schema type is mapped with its built-in TTCN-3 type, in a separate importable module. The identifiers of types, signatures, parameters, ports, exceptions, groups, constants are obtained from their WSDL definition counterparts. Besides parallel test components (derived from each <*service*>), an Abstract Test System Interface component [1] is fabricated. Although the test logic cannot be entirely defined, we are able to generate also basic testcases for each <*operation*> of the service ports. This is done using templates derived from generated types instead of actual values both at *calls* and *verdicts* establishment. The specific test data must still be manually entered in these templates.

The parameters for the connection with each SUT are passed to the underlying adapter by using an external function. They derive from the <*binding*> and <*service*> WSDL sections and specify the protocol (i.e. SOAP), the binding style (e.g. document or RPC), transport (HTTP, SMTP, etc.) and endpoint identifiers (URI, URL).

Implementation

Aiming to obtain concrete TTCN-3 definitions, an existing TTCN-3 compiler was extended (TTthree) for importing and compiling and later TTworkbench was used for running the resulted test suites. Both tools were provided by Testing Tech GmbH.

TTthree is a TTCN-3 TDOM-based compiler, that translates TTCN-3 sources first into Java files which are then normally compiled. The compiler has a client-server architecture. Every compilation starts a new *Client* process, and every class loading and verification of the TDOM-model is done once, by the *Server* process. *TDOM* is a Typed Document Object Model (described in [19]), that can be seen as a low-level XML representation of a TTCN-3 module.

When an import statement that relates to a WSDL file is encountered in a TTCN-3 module, the compiler behaves as depicted in Figure 3. At the logical level, the generation is done in two steps, both of them being tightly integrated in the TTthree compiler.

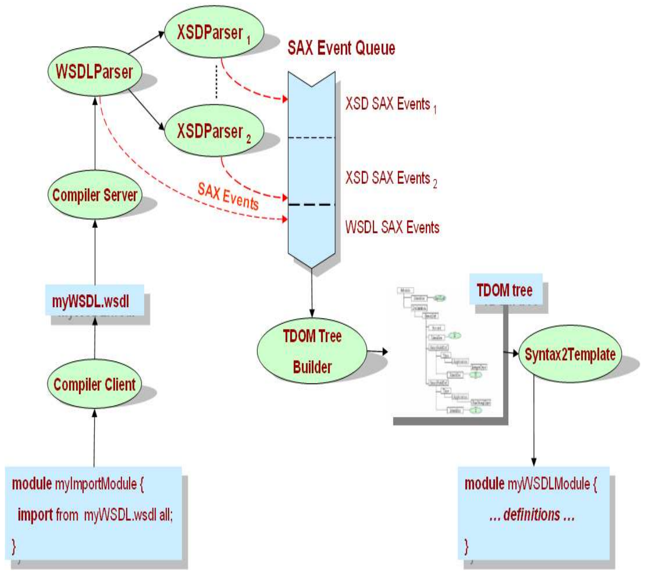

Fig. 3. The Workflow of the WSDL Import Implementation

First, in the **explicit** mapping, the WSDL document is translated into a TDOM. When the TTthree Server component receives the WSDL file from the Client (whose creation was triggered by the import statement), a WSDL-Parser thread starts to parse and validate the WSDL definitions using JDOM. The aim of this parsing thread is to fill in an event queue (SAXEventQueue) whose occurrences are related to the WSDL-specific elements encountered during parsing. Concerning implementation details, important is the multithreading mechanism occurring at parsing. WSDL types are usually defined in XML Schema; this leads to the need of importing also XML Schema types. First *wsdl4j* was employed to extract these embedded types, but it proved to be cumbersome. Instead it was decided to remodel and use the XMLSchema2TTCN3 importer [15] by creating a new XSDParser thread whenever a type definition is found in the document. Both types of threads fire events using a SAXEventGenerator in the shared SAXEventQueue, for all encountered WSDL or XML Schema elements. Depending on the type of the new definition instance, it is known what events are fired, and in which order. All these events contained in the queue are then dispatched to another component, a TDOM tree builder, that converts them to TDOM nodes.

In the second step – the **implicit** mapping – the obtained TDOM is used for TTCN-3 plain code generation. The component responsible for TDOM traversal is a TTthree built-in tool called Syntax2Template.

For the TTCN-3 adaptation layer, a new and generic Test Adapter (based on W3C and ETSI standards) was implemented. This Adapter is configurable by using generic parameters transmitted from TTCN-3 (e.g. WS endpoints, connection parameters) and translates the TTCN-3 messages back and forth to SOAP envelopes that are sent to / received from the SUT.

A Basic Example

To prove the concept, chosen a simple case study was chosen. The tested Web services provide interfaces for converting between various temperature scales (i.e. Celsius and Fahrenheit).

Two sample SUTs were developed in order to have control over the implementation of the services; they are providing two operations and four compound data-types. The first one was implemented using the Web Tools Project of Eclipse (the underlying technologies were Apache Tomcat and Axis [13]). The second one was written in C# using the Microsoft .NET framework and publishes the same WSDL. The reason of having two SUTs implementing the same interface was to attest that our technique enables testing of many different Web services development platforms using the same system model.

```
<definitions targetNamespace="http://tempuri.org/"
 xmlns:soap="http://schemas.xmlsoap.org/wsdl/soap/">
<types>
 <s:schema ... >
  <s:element name="CelsiusToFahrenheit">
   <s:complexType><s:sequence>
    <s:element minOccurs="1" maxOccurs="1"
      name="Fahrenheit" type="s:float"/>
   </s:sequence> </s:complexType>
  </s:element>
  <s:element name="CelsiusToFahrenheitResponse">
   <s:complexType> <s:sequence>
    <s:element name="CelsiusToFahrenheitResult"
      minOccurs="1" maxOccurs="1" type="s:float"/>
   </s:sequence> </s:complexType>
  </s:element>
  ...
 </s:schema>
</types>
<message name="CelsiusToFahrenheitSoapIn">
 <part name="parameters"
    element="s:CelsiusToFahrenheit"/>
</message>
...
<portType name="TempConvertSoap">
 <operation name="FahrenheitToCelsius">
  <input message="s:FahrenheitToCelsiusSoapIn"/>
  <output message="s:FahrenheitToCelsiusSoapOut"/>
 </operation>
 <operation name="CelsiusToFahrenheit">
  <input message="s:CelsiusToFahrenheitSoapIn"/>
  <output message="s:CelsiusToFahrenheitSoapOut"/>
 </operation>
</portType>
<binding name="TempConvertSoap"
    type="s:TempConvertSoap" >
 <soap:binding style="document" transport=
    "http://schemas.xmlsoap.org/soap/http"/>
 <operation name="FahrenheitToCelsius">
  <soap:operation  style="document" soapAction=
    "http://tempuri.org/FahrenheitToCelsius"/>
  <input><soap:body use="literal"/></input>
  <output><soap:body use="literal"/></output>
 </operation>
 <operation name="CelsiusToFahrenheit">
  <soap:operation style="document" soapAction=
    "http://tempuri.org/CelsiusToFahrenheit"/>
  <input><soap:body use="literal"/></input>
  <output><soap:body use="literal"/></output>
 </operation>
</binding>
<service name="TempConvert">
 <port name="TempConvertSoap"
    binding="s:TempConvertSoap">
  <soap:address location=
"http://localhost:8080/Convertor/services/Converter"/>
 </port>
</service>
</definitions>
```

It comprises of two parts, one providing the abstract test suite (ATS) and the other providing the generic adaptation layer. The following test definitions have been automatically generated using our tool:

```
type record CelsiusToFahrenheit {
  floatXSD Celsius }
type record CelsiusToFahrenheitResponse {
  floatXSD CelsiusToFahrenheitResult }
group Interface{
  signature FahrenheitToCelsius {
    in FahrenheitToCelsius parameters)
    return FahrenheitToCelsiusResponse;
  signature CelsiusToFahrenheit {
    in CelsiusToFahrenheit parameters)
    return CelsiusToFahrenheitResponse;
  type port WSport procedure {
    inout FahrenheitToCelsius;
    inout CelsiusToFahrenheit;
    in FahrenheitToCelsiusResponse,
       CelsiusToFahrenheitResponse; }
}
type component ClientComp {
  port WSport opPortClient; }
type component OperationSys {
  port WSport opPortSystem; }
```

A Web service operation invocation may generate also *faults*. TTCN-3 procedure-based communication has the possibility of defining exceptions (from their *fault* counterparts). This is the reason for which it was chosen instead of the message-based interaction with the SUT. This fact explains the generated signatures that maps to each operation exposed by the WS. The two WSDL files published by the SUTs slightly differ in terms of invocation details (e.g. endpoint location). Two simple testcases are created, each testing a different operation. The specific SUT invocation details are reflected by generating a constant (*wsConnectorDetails*) used in each test case as a parameter and passed to the Test Adapter using an external function(*createConnector*). For each test case, template data for the TTCN-3 generated types were hand written and used for stimulation and verification of the response of the SUT. The following code snippet is an example of a basic request-response interaction scenario corresponding to an <*operation*> defined in the imported WSDL:

```
const float waitingTime := 10.0;
template CelsiusToFahrenheit
  sent_CelsiusToFahrenheit := { Celsius := 20.0 }
template CelsiusToFahrenheitResponse
  expected_CelsiusToFahrenheit :=
  { CelsiusToFahrenheitResult := 68.0 }
testcase Test_CelsiusToFahrenheit
  (InvocationDetailsType wsConnectorDetails)
  runs on ClientComp system OperationSys {
var CelsiusToFahrenheitResponse
  v_CelsiusToFahrenheitResponse;
createConnector(wsServerConnector,
  wsConnectorDetails);
map(self: opPortClient, system: opPortSystem);
opPortClient.call(CelsiusToFahrenheit:
    {sent_CelsiusToFahrenheit},waitingTime) {
 [] opPortClient.getreply (CelsiusToFahrenheit:
        {-} value ?) ->
        value v_CelsiusToFahrenheitResponse {
    if (v_CelsiusToFahrenheitResponse.
        CelsiusToFahrenheitResult ==
        expected_CelsiusToFahrenheit) {
        setverdict (pass);
    } else {
        setverdict (fail);
    }
 }
 [] opPortClient.catch (timeout) {
        setverdict(fail); }
 }
}
```

The generated test suite was used to test the two implementations under test (IUT). Though the goal was not to extensively test the IUTs, the test was repeated several times with diverse combinations of stimuli and oracle data. However, the outcome of the experiment reveals that the test suite generated from a system model suffices to easily validate multiple IUTs exposing from the same system model.

VI. Conclusions and Further Work

This paper presents a method of generating TTCN-3 test suites out of WSDL documents. The method allows mapping of virtually all WSDL definitions into TTCN-3 constructs.

Furthermore, there are some additional ideas for future work. A cornerstone issue addresses the generation of concrete test data starting from XML Schema type definitions. This data should be used directly in the automated testing process, without requiring manual intervention. The next working issue is to extend the method for supporting UDDI and the whole Web Services architecture. In the context of composite Web services (SOA), generation of full-fledged TTCN-3 directly from BPEL processes becomes imperative.

In particular, test automation will be essential to a sound and efficient Web services development process, for the assessment of the functionality, scalability and performance as well as for the acceptance of Web services built by application providers.

References

[1] ETSI: "*TTCN-3 Standard Part 1: ES 201 873-1 V3.1.1 - TTCN-3 Core Language*", 2005-06

[2] TestingTech: "*Using TTthree - Users Manual and Programming Guide*", `http://www.testingtech.de/`

[3] W. T. Tsai, R. Paul, Z. Cao, L. Yu, A. Saimi, B. Xiao: "*Verification of Web Services Using an Enhanced UDDI Server*", Object-Oriented Real-Time Dependable Systems - WORDS 2003

[4] Reiko Heckel ,Leonardo Mariani: "*Automatic Conformance Testing of Web Services*", FASE 2005, pp 34-48

[5] Dafina-Maria Jeaca: "*XML Schema to TTCN-3 Mapping - Importing XML Schema Datatypes into TTCN-3*", UPB Bucharest, Diploma thesis, 2004

[6] W3C: "*Web Services Description Language (WSDL) Version 2.0 Part 1: Core Language*", `http://www.w3.org/TR/wsdl20/`

[7] J. Grabowski, D. Hogrefe, G. Rethy, I. Schieferdecker, A. Wiles, C. Willcock: "*An Introduction into the Testing and Test Control Notation (TTCN-3)*", Computer Networks, 2003, (Vol. 42, Issue 3), pp.375-403

[8] I. Schieferdecker, G. Din, D. Apostolidis: "*Distributed functional and load tests for Web services*", International Journal on Software Tools for Technology Transfer (STTT), 2005, (Vol. 7, Issue 4), pp. 351-360

[9] WS-I: "*WS-I Profiles*", `http://www.ws-i.org/deliverables/Default.aspx`

[10] OASIS, UDDI Spec TC: "*UDDI Version 3.0.2*", `http://www.uddi.org/`

[11] W3C: "*SOAP Version 1.2*", `http://www.w3.org/TR/soap/`

[12] OASIS: "*BPEL*", `http://www.oasis-open.org/committees/documents.php?wg_abbrev=wsbpel`

[13] Apache: "*Apache Axis*", `http://ws.apache.org/axis/`

[14] W. T. Tsai, Ray Paul, Yamin Wang, Chun Fan, Dong Wang: "*Extending WSDL to Facilitate Web Services Testing*", IEEE Computer Society, 2002, Proceedings of the 7th IEEE International Symposium on High Assurance Systems Engineering (HASE'02), pp. 171

[15] D.-M. Jeaca, G. Din, A. Rennoch: "*Importing XML Schema datatypes into TTCN-3*", Proceedings of 3rd Workshop on Systems Testing and Validation (STV04), 2004

[16] I. Schieferdecker, B. Stepien: "*Automated Testing of XML/SOAP Based Web Services*", 13th Fachkonferenz "Kommunikation in Verteilten Systemen", 2003

[17] Schahram Dustdar, Stephan Haslinger: "*Testing of Service Oriented Architectures - A practical approach*", 2004

[18] P. Xiong, R. L. Probert, B. Stepien: "*An Efficient Formal Testing Approach for Web Service with TTCN-3*", International Conference on Software, Telecommunications and Computer Networks, 2005

[19] B. Trancón y Widemann, M. Lepper, J. Wieland, P. Pepper: "*An Efficient Formal Testing Approach for Web Service with TTCN-3*", Proc. of the XSE Workshop, 2001