DISCOVERY AND SELECTION OF CERTIFIED WEB SERVICES THROUGH REGISTRY-BASED TESTING AND VERIFICATION

Dimitrios Kourtesis[1], Ervin Ramollari[1],
Dimitris Dranidis[2], Iraklis Paraskakis[1]

[1] *South East European Research Centre (SEERC),*
Research Centre of the University of Sheffield and CITY College
Mitropoleos 17, 54624, Thessaloniki, Greece
dkourtesis@seerc.org, erramollari@seerc.org, iparaskakis@seerc.org

[2] *Computer Science Department, CITY College,*
Affiliated Institution of the University of Sheffield,
Tsimiski 13, 54624 Thessaloniki, Greece
dranidis@city.academic.gr

Reliability and trust are fundamental prerequisites for the establishment of functional relationships among peers in a Collaborative Networked Organisation (CNO), especially in the context of Virtual Enterprises where economic benefits can be directly at stake. This paper presents a novel approach towards effective service discovery and selection that is no longer based on informal, ambiguous and potentially unreliable service descriptions, but on formal specifications that can be used to verify and certify the actual Web service implementations. We propose the use of Stream X-machines (SXMs) as a powerful modelling formalism for constructing the behavioural specification of a Web service, for performing verification through the generation of exhaustive test cases, and for performing validation through animation or model checking during service selection.

1. INTRODUCTION

Reliability is a fundamental prerequisite for establishing collaboration within a network of peers, be them human or machines, and issues such as trust management and evaluation of trustworthiness are significant challenges in Collaborative Networks research. Collaborative business processes built on top of service-oriented infrastructures in networked organisations are typically realised as compositions of autonomous but trustworthy Web services provided and consumed across intra- and inter-organisational boundaries. The individual Web services that a collaborative business process comprises are discovered and composed at design-time on the basis

of some specification that is meant to explicate the functional or non-functional properties of each service.

The *formality* of this specification may vary depending on the employed service description framework, and the degree of ambiguity or rigour in the description determines the extent to which a specification can be amenable to automated processing. The *correctness* of the implementation with respect to its corresponding specification may also vary, due to modelling inconsistencies that may be intentional or unintentional. In highly dynamic and loosely coordinated environments, service specifications are especially likely to become outdated and unreliable sources of information due to Web service implementations becoming modified or replaced, or due to other changes in infrastructure. To avoid making design-time decisions based of inaccurate information and incurring the associated cost of run-time errors, special care must be taken to ensure that service specifications are reliable, i.e. that they correspond to the actual service implementations they are meant to describe.

This paper introduces a novel approach towards extending the capabilities of a service registry with additional *functional testing* and behavioural verification functionality that can serve as a basis for overcoming the aforementioned challenge. We propose the use of Stream X-machines (SXMs) (Laycock, 1993) (Holcombe and Ipate, 1998) as a modelling formalism for constructing the behavioural specification of a service at the provider-side, and generating test cases at the registry-side to verify that the actual service implementation conforms to the specification. A significant advantage of Stream X-machines compared to other behavioural modelling and testing formalisms is in their associated testing method, which is guaranteed to reveal all inconsistencies among an implementation under test and an advertised specification (Dranidis, Kourtesis and Ramollari, 2007). The Web service test set that is generated by the registry is represented as a sequence of operation invocations with appropriate inputs and expected outputs. By applying the generated set of tests to a Web service implementation and evaluating its responses, the registry can conclude if it is behaviourally-equivalent to its associated specification. The registry acts as a *certification authority* for service specifications and as a trusted third party that service consumers can rely on for design-time *discovery*.

An additional feature of the proposed approach that sets it apart from other solutions in the literature is that the SXM specification and the generated test sets can be used not only for registry-side *verification*, but also for consumer-side *validation* after discovery and during service *selection*. Service consumers can validate the behaviour of every certified candidate service that the registry returns as a match to their needs, in order to select the most appropriate one. Validation can take place either by executing the SXM specification with an X-machine animator and visually inspecting its behaviour, or by performing model-checking to assert desirable or undesirable properties described by temporal logic formulae. This allows service consumers to essentially simulate the service behaviour and evaluate it without having to perform real testing with the associated overhead for both service consumer and service provider.

The rest of this paper is organised as follows. Section 2 presents a summary of related work in the domain of model-based Web service testing and verification. Section 3 provides an overview of the Stream X-machine modelling formalism that this paper suggests as a suitable means to model the behaviour of stateful Web services. Section 4 provides an overview of the proposed holistic approach for discovery and selection of certified services through registry-based testing and

verification, presenting the approach from the perspectives of the provider, the registry and the consumer, and emphasising on their associated activities. Section 5 concludes the paper by summarising the main points of the presented work and suggesting directions for future research.

2. RELATED WORK

A number of approaches have been proposed for the verification of Web services by employing model-based testing. In (Sinha and Paradkar, 2006) a method is proposed for annotating a WSDL document with concepts from an OWL ontology representing inputs, outputs, preconditions and effects, and automatically translating the resulting WSDL-S specification into a semantically-equivalent extended Finite State Machine (EFSM) model. A set of manual or automated techniques for generating test cases based on the EFSM model is also provided. The techniques vary in terms of adequacy criteria, coverage and completeness.

The use of an EFSM modelling formalism for describing the dynamic behaviour of a Web service is also proposed in (Keum, Kang and Ko, 2006), where a manual procedure is suggested for deriving the EFSM model from a WSDL description. The proposed EFSM model is an FSM extended with memory, predicate conditions and computing blocks for state transitions. With proper tool support the EFSM model can be used for automatically generating Web service test cases with increased test coverage that includes both control flow and data flow. The authors provide experimental results showing that their method has the potential to find more faults compared to other methods, but notably without completeness guarantees.

The number of research works proposing the incorporation of Web service model-based testing and verification functionality in service registries is rather limited. The addition of a lightweight verification mechanism to UDDI service registries was first proposed in (Tsai et al, 2003). The key idea was to attach so-called “test scripts” to Web service specifications for both service registry and service consumers to use. Before publishing a service advertisement at the service registry or before consuming a service the associated test scripts could be used to test the actual service and verify its behaviour. The proposed approach is very abstract and does not prescribe the use of a specific formal or informal method of representing service behaviour, nor one for generating the test scripts.

In (Bertolino et al., 2005) the authors propose a framework with an enhanced UDDI registry that generates test cases for Web services, executes them, and monitors the interactions between the service under test and other services already registered with the framework in order to verify conformance to the published specification. Emphasis is placed on verifying that a Web service is interoperable with other registered services, and the framework is called an “audition framework” in the sense that a Web service undergoes a monitored trial before being admitted. The authors suggest that the behavioural service specification should be expressed as a UML 2.0 Protocol State Machine (PSM) diagram that can be semi-automatically transformed into a Symbolic Transition System (STS) on which existing automated test generation methods can be readily applied. The utilisation of the proposed behavioural specification formalism for matchmaking among service advertisements and requests is left undefined. Discovery is assumed to be supported

by the typical means available in UDDI, i.e. keyword-based search and categorisation.

In (Heckel and Mariani, 2005) the authors propose a "high-quality service discovery" approach that incorporates automatic testing and verification of Web Services before allowing their registration to the service registry. The authors propose Graph Transformation (GT) rules as the modelling formalism to be used for constructing behavioural service specifications. Conformance test cases are to be automatically generated from the provided specification and executed against the target Web Service. If the test is successfully passed, the service can be registered. Apart from testing and verification the GT-based service specifications can be also used for matchmaking among services and service requests that have been also expressed via GT rules. The proposed approach does not prescribe the use of UDDI or any other specific service registry specification as the technical infrastructure to support the approach.

A significant drawback in the above model-based verification approaches is that the test case derivation methods they employ cannot guarantee completeness in testing of the service implementations. In contrast, the Stream X-machine testing method on which our approach relies is proven to generate a complete set of test cases that can reveal all inconsistencies among an implementation under test and an SXM specification (Ipate and Holcombe, 1997). Moreover, a novel proposition in our approach is the use of the behavioural service specification by service consumers to perform validation after discovery, during the phase of service selection, through model animation or model checking. Validation is an important utility for service consumers, since it can assist them in selecting the most appropriate services from a list of candidates, regardless of the matchmaking and discovery method that was used to deliver this list.

3. MODELLING SERVICES AS STREAM X-MACHINES

Stream X-machines (SXMs) are a computational model capable of representing both the data and the control of a system. SXMs are special instances of the X-machines introduced in 1974 by Samuel Eilenberg (Eilenberg, 1974). They employ a diagrammatic approach of modelling control flow by extending the expressive power of finite state machines. In contrast to finite state machines, SXMs are capable of modelling non-trivial data structures by employing a memory attached to the state machine. Moreover, transitions between states are not labelled with simple input symbols but with processing functions. Processing functions receive input symbols and read memory values, and produce output symbols while modifying memory values. The benefit of adding the memory construct is that the state explosion is avoided and the number of states is reduced to those states which are considered critical for the correct modelling of the system's abstract control structure. A divide-and-conquer approach to design allows the model to hide some of the complexity in the transition functions, which are later exposed as simpler SXMs at the next level.

A Stream X-machine is defined as an 8-tuple, $(\Sigma, \Gamma, Q, M, \Phi, F, q_0, m_0)$ where:

- Σ and Γ is the input and output finite alphabet respectively;
- Q is the finite set of states;

- M is the (possibly) infinite set called memory;
- Φ, which is called the type of the machine SXM, is a finite set of partial functions (processing functions) φ that map an input and a memory state to an output and a new memory state, $\varphi : \Sigma \times M \rightarrow \Gamma \times M$;
- F is the next state partial function that given a state and a function from the type Φ, provides the next state, $F : Q \times \Phi \rightarrow Q$ (F is often described as a state transition diagram);
- q_0 and m_0 are the initial state and memory respectively.

Apart from being formal as well as proven to possess the computational power of Turing machines (Holcombe and Ipate, 1998), SXMs offer a highly effective testing method for verifying the conformance of a system's implementation against a specification. Stream X-machine models can be represented in XMDL (X-Machine Definition Language), a special-purpose markup language introduced in (Kapeti and Kefalas, 2000). XMDL has served as a common language for the development of numerous tools supporting Stream X-machines (Kefalas, Eleftherakis and Sotiriadou, 2003). An extension of XMDL to support an object-based notation was suggested in (Dranidis, Eleftherakis and Kefalas, 2005). The object-based extension, called XMDL-O, enables an easier and more readable specification of Stream X-machines.

In order to model the behaviour of a Web service using a Stream X-machine, the modeller must perform data-level and behaviour-level analysis to derive the appropriate SXM modelling constructs. Parallels can be easily drawn among a stateful Web service and a Stream X-machine, since they both accept inputs and produce outputs, while performing specific actions and moving from one internal state to another. SXM inputs correspond to SOAP request messages, outputs correspond to SOAP response messages, and processing functions correspond to Web service operation invocations in specific contexts (an operation invocation may map to more than one processing functions because of the potentially different input data parameters values provided). In addition, the modeller has to define the memory structure, not only as a substitute for internal state, but also to supply sample test data that can become part of the generated test sequences. SXM-based modelling is applicable in the context of complex conversational Web services where the result obtained from invoking a Web service operation depends not only on the consumer's input, but also on the internal state of the service.

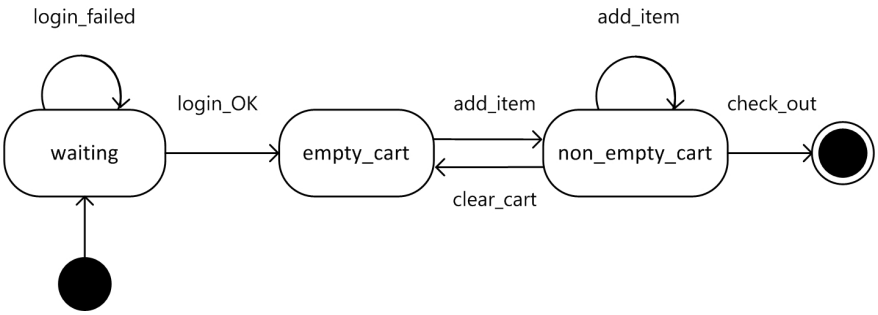

Figure 1 – Example of Stream X-machine model for a shopping cart Web service

Figure 1 illustrates an example SXM model of a simple Web service that provides the backend functionality of a shopping cart to Web-based client applications. The service comprises four operations (login, addToCart, clearCart, and checkout) allowing customers to perform authentication, add items to the shopping cart, clear the cart, and proceed to checkout. The SXM modelling constructs depicted in Figure 1 are the states belonging to set *Q*, the names of processing functions belonging to set *Φ*, and the state transition diagram corresponding to *F*. A fully-detailed description of this modelling example, including a complete definition of all processing functions, inputs, outputs and memory is provided in (Dranidis, Kourtesis and Ramollari, 2007).

4. ROLES AND ACTIVITIES IN THE PROPOSED APPROACH

The approach that we put forward in this paper involves all three types of stakeholders in a SOA environment, i.e. service providers, service registries, and service requestors (consumers). As depicted in Figure 2, the role of each stakeholder is associated with a number of activities. In brief, we propose that the behaviour of a Web service should be formally modelled at the provider-side, in order to facilitate registry-side verification at the time of service publication and consumer-side validation at the time of service selection. In the following three sections we present an overview of the activities performed by each stakeholder in the scheme.

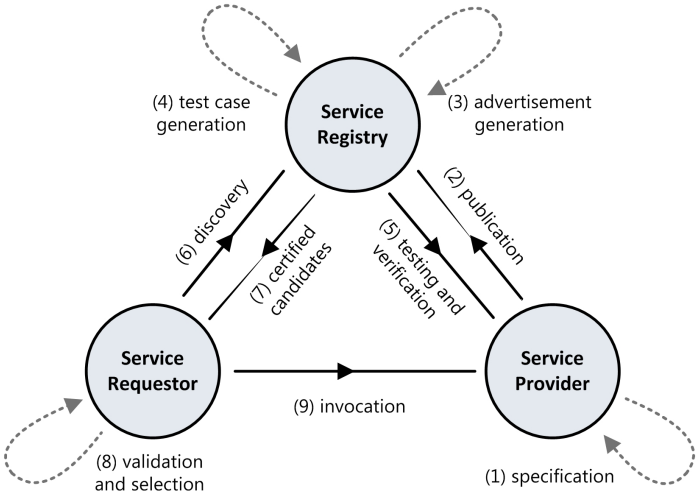

Figure 2 – Stakeholder roles and ordering of activities in the proposed approach

4.1 Provider-side construction of a behavioural specification

The objective of the service provider is to construct a formal model reflecting the behaviour of the service to be published (activity 1 in Figure 2) using the Stream X-machine (SXM) formalism as described in section 3. The SXM model must be

encoded in XMDL-O and stored in an external document that must be subsequently "linked" with the service's WSDL document. The association among the two document artefacts can be established by employing the SAWSDL (Semantic Annotations for WSDL) (Farrell and Lausen, 2007) specification and its mechanism for annotating Web service descriptions with pointers to externally maintained semantically-rich specifications. In order to indicate the association between the two documents an SAWSDL *modelReference* annotation pointing to the URL of the SXM specification document must be placed within the *wsdl:portType* definition of the service's WSDL document.

The process of constructing an SXM model from a WSDL description can be automated to a great extent by modelling inputs, outputs, preconditions and effects (IOPE) as concepts in an OWL ontology and then pointing to them from within a WSDL document through SAWSDL annotations. The description of an approach for modelling Web service inputs and outputs in an OWL-DL ontology and then creating semantically annotated service descriptions using SAWSDL is provided in (Kourtesis and Paraskakis, 2008). The method has been extended for modelling preconditions and effects in an OWL-DL ontology and for capturing the IOPE semantics of Web service interfaces through SAWSDL annotations. An algorithm has been also defined for the semi-automated transformation of the resulting SAWSDL specification into an SXM specification, but the presentation of these extensions is beyond the scope of this paper. Modelling of IOPE semantics in the aforementioned manner would not only assist in increasing the automation of the SXM model construction process, but would also serve as a basis for performing semantically-enriched matchmaking and discovery for high-precision retrieval of services, as discussed in (Kourtesis and Paraskakis, 2008b).

Regardless of the method used to construct the SXM specification, manual or semi-automated, as soon as the semantically annotated WSDL document is complete, the provider must submit it to the service registry for processing and publication (activity 2).

4.2 Registry-side generation of test cases and verification

The objective of the service registry is to verify that the service implementation is functionally conformant to its advertised specification, and if this holds, provide a certification for the service advertisement. All activities within the service registry are automated, and their ordering is as follows. Firstly, the registry processes the incoming SAWSDL description and creates a service advertisement with a status of pending certification (activity 3). Secondly, the attached SXM specification is used for deriving a complete set of test cases that can reveal all inconsistencies in the service implementation to be verified (activity 4). Lastly, the executable tests are run by the registry's SOAP testing engine and if the results are successful (i.e. if the produced outputs match the expected ones) the service advertisement obtains certification status (activity 5).

The benefit of performing this procedure at the registry-side and at the time of publication, as opposed to performing it on the consumer-side at the time of service selection, is that it needs to be performed only once, by a trusted third party that assumes the responsibility of certification and can be held liable for its decisions. Since only successfully tested services receive certification status by the registry,

consumers can be sure that the specifications of the services they discover are reliable sources of information.

As already mentioned, the SXM testing method that serves as the foundation of our approach is guaranteed to generate a complete, finite set of test cases that can reveal all inconsistencies among an SXM specification and an implementation under test. This is an important criterion for entrusting the process of verification and certification to the registry. The SXM testing method is a generalization of the W-method (Chow, 1978) and works on the basis that both specification and implementation could be represented as Stream X-machines with the same type Φ (i.e. both specification and implementation have the same processing functions), where Φ satisfies two fundamental design for test conditions: (i) completeness with respect to memory – all processing functions can be exercised from any memory value using appropriate inputs, and (ii) output distinguishability – any two different processing functions will produce different outputs if applied on the same memory/input pair. More details about the derivation of the test sequences are provided in (Dranidis, Kourtesis and Ramollari, 2007).

For our testing approach to be applicable, we assume that the operations of the Web service under test follow the request-response message exchange pattern, i.e. they accept a request message from the consumer and return a response message. This makes it possible to fulfil the condition for output distinguishability, and also enables the testing engine to understand which processing functions have been activated during an execution path based on responses of the service. We do not consider this restriction important or unrealistic, since the request-response message exchange pattern is currently the typical way of engineering Web services.

4.3 Consumer-side validation and service selection

The next activity in the process is for the service consumer to formulate a discovery query and submit it to the service registry (activity 6). The registry will perform some form of matchmaking based on the available advertisements and the specified request, and return the results (activity 7). The discovery and matchmaking method by which the candidate services will be derived is independent from the rest of the approach, and can be based on any existing method. A semantically-enhanced service matchmaking method such as the one described in (Kourtesis and Paraskakis, 2008) would however by strongly encouraged, since it is free of ambiguity, takes more information into consideration, and has the potential of resulting in more accurate matches. In any case, if the registry returns more than one certified services as matching candidates, the consumer must go through a service selection process (activity 8).

As already discussed, the SXM specification that is associated with each of the certified candidate services can be used not only for registry-side verification, but also for consumer-side validation during service selection. A method that enables behavioural validation is model animation through appropriate tools. During animation the consumer feeds the SXM model with sample inputs while observing the current state, transitions, processing functions, memory values, as well as outputs. The sample inputs to be provided for driving the animator can be the actual test data that were generated and used by the service registry at the phase of verification. This would relieve the service consumer from the burden of re-generating the data from the SXM specification.

The animation process can be readily supported by existing tools. X-System (Kapeti and Kefalas, 2000) is a Prolog-based tool supporting the animation of Stream X-machine models, while a Java-based graphical user interface on top of X-System is also available. In addition to animation, model checking techniques can be employed on the SXM model to check for desirable or undesirable properties specified with temporal logic formulae. Research on X-machines already offers a model-checking logic, called XmCTL, which extends Computation Tree Logic (CTL) with memory quantifiers in order to facilitate model-checking of temporal properties in X-machine models (Eleftherakis, Kefalas and Sotiriadou, 2001).

5. CONCLUSIONS

The contemporary IT infrastructure landscape is changing rapidly, and SOA-based solutions are becoming dominant. Collaborative business processes layered on top of SOA infrastructures are typically realised as compositions of autonomous Web services that are discovered and composed on the basis of some specification explicating their functional or non-functional properties. In highly dynamic and loosely coordinated environments service specifications can easily become outdated and dissolve into unreliable sources of information. To avoid making design-time decisions based of inaccurate information and incurring the associated cost of run-time errors we propose a registry-based testing and verification approach that can be used for ensuring that service specifications are reliable, i.e. that they correspond to the actual service implementations they are meant to model.

The approach that we put forward in this paper involves all three types of stakeholders in a SOA environment, i.e. service providers, service registries, and service consumers. The service registry becomes a trusted third party and certification authority that undertakes the responsibility of testing a service's implementation to verify that it conforms to its advertised formal specification. We propose the use of Stream X-machines (SXMs) as a powerful modelling formalism for constructing the behavioural specification of a Web service at the provider-side, in order to facilitate registry-side verification at the time of service publication and consumer-side validation at the time of service selection.

The particular strengths of the presented approach compared to other works in the literature can be summarised in three points. Firstly, a significant advantage of Stream X-machines compared to other behavioural modelling and testing formalisms is in their associated complete testing method, which is guaranteed to reveal all inconsistencies among a specification and the implementation under test. Secondly, the SXM specification and the generated test sets can be used not only for registry-side verification, but also for consumer-side validation after discovery and during service selection. Thirdly, the proposed approach can be readily supported by a number of existing tools for SXM modelling, test case generation, verification, and validation, as well as an existing open source service registry implementation for performing semantically-enhanced publication and discovery of services. The main objective for future research is the consolidation of existing techniques, methods and tools into a comprehensive application framework and the development of the connecting components and user-friendly interfaces that would be required in order to yield an all-inclusive solution with industrial applicability.

6. ACKNOWLEDGMENTS

This research work was partially supported by FUSION (Business process fusion based on semantically-enabled service-oriented business applications), a research project funded by the European Commission's 6th Framework Programme for RTD under contract number FP6-IST-2004-170835 (http://www.fusion-strep.eu/).

7. REFERENCES

1. Bertolino A., Frantzen I., Polini A. and Tretmans J. Audition of Web Services for Testing Conformance to Open Specified Protocols. *Architecting Systems with Trustworthy Components*, Springer LNCS 3938, 2006, pp. 1-25.
2. Chow T.S. Testing Software Design Modelled by Finite State Machines. *IEEE Transactions on Software Engineering, Vol. 4*, 1978, pp. 178-187.
3. Dranidis D., Eleftherakis G., and Kefalas P. Object-based Language for Generalized State Machines. *Annals of Mathematics, Computing and Teleinformatics (AMCT), Vol. 1, No. 3*, 2005, pp. 8-17.
4. Dranidis D., Kourtesis D. and Ramollari E. Formal Verification of Web Service Behavioural Conformance through Testing. *Annals of Mathematics, Computing & Teleinformatics (AMCT), Vol. 1, No. 5*, 2007, pp. 36-43.
5. Eilenberg S. *Automata, Languages and Machines, Volume A.* Academic Press, New York, 1974.
6. Eleftherakis G., Kefalas P., and Sotiriadou A. XmCTL: Extending Temporal Logic to Facilitate Formal Verification of X-machines. *Analele Universitatii Bucuresti, Matematica-Informatica*, 50:79-95, 2001.
7. Farrell J. and Lausen H. (Eds). *Semantic Annotations for WSDL and XML Schema (SAWSDL)*. W3C Recommendation, August 2007.
8. Heckel R. and Mariani L. Automatic Conformance Testing of Web Services. In Cerioli M. (Ed.): *FASE 2005*, LNCS 3442, Springer-Verlag Berlin Heidelberg 2005, pp. 34-48.
9. Holcombe M. and Ipate F. *Correct Systems: Building Business Process Solutions*. Springer Verlag, Berlin, 1998.
10. Ipate F. and Holcombe M. An integration testing method that is proved to find all faults. *International Journal of Computer Mathematics, Vol. 63*, 1997, pp. 159-178.
11. Kapeti E. and Kefalas P. A Design Language and Tool for X-Machine Specification. *Advances in Informatics*, Fotiadis D. and Nikolopoulos S. (Eds), World Scientific, 2000, pp. 134-145.
12. Kefalas P., Eleftherakis G. and Sotiriadou A. *Developing Tools for Formal Methods*. In Proceedings of the 9th Panhellenic Conference in Informatics (PCI 2003), November 2003, pp. 625-639.
13. Keum C., Kang S. and Ko I.Y. *Generating Test Cases for Web Services using Extended Finite State Machine*. In Proceedings of the 18th IFIP International Conference on Testing Communicating Systems (TestCom 2006), Springer, 2006, pp. 103-117.
14. Kourtesis D. and Paraskakis I. Web Service Discovery in the FUSION Semantic Registry. In Abramowicz W. and Fensel D. (Eds.): *BIS 2008*, LNBIP 7, Springer-Verlag Berlin Heidelberg 2008, pp. 285–296.
15. Kourtesis D. and Paraskakis I. Combining SAWSDL, OWL-DL and UDDI for Semantically Enhanced Web Service Discovery. In Bechhofer S. et al.(Eds.): *ESWC 2008*, LNCS 5021, Springer-Verlag Berlin Heidelberg 2008, pp. 614–628.
16. Laycock G. *The Theory and Practice of Specification-Based Software Testing*. PhD thesis, Department of Computer Science, University of Sheffield, UK, 1993.
17. Sinha A. and Paradkar A. *Model-based Functional Conformance Testing of Web Services Operating on Persistent Data*. In Proceedings of Workshop on Testing, Analysis and Verification of Web Services and Applications (TAV-WEB'06), July 2006, pp. 17-22.
18. Tsai W.T., Paul R., Cao Z., Yu L., Saimi A. and Xiao B. *Verification of Web Services using an Enhanced UDDI Server*. In Proceedings of 8th IEEE International Workshop on Object-oriented Real-time Dependable Systems (WORDS 2003), January 2003, pp. 131-138.