LTSA-WS: A Tool for Model-Based Verification of Web Service Compositions and Choreography

Howard Foster, Sebastian Uchitel, Jeff Magee, Jeff Kramer
Department of Computing, Imperial College London
180 Queen's Gate, SW7 2BZ
United Kingdom
{hf1,su2,jnm,jk}@doc.ic.ac.uk

ABSTRACT
In this paper we describe a tool for a model-based approach to verifying compositions of web service implementations. The tool supports verification of properties created from design specifications and implementation models to confirm expected results from the viewpoints of both the designer and implementer. Scenarios are modeled in UML, in the form of Message Sequence Charts (MSCs), and then compiled into the Finite State Process (FSP) process algebra to concisely model the required behavior. BPEL4WS implementations are mechanically translated to FSP to allow an equivalence trace verification process to be performed. By providing early design verification and validation, the implementation, testing and deployment of web service compositions can be eased through the understanding of the behavior exhibited by the composition. The approach is implemented as a plug-in for the Eclipse development environment providing cooperating tools for specification, formal modeling, verification and validation of the composition process.

Categories and Subject Descriptors
D.2.4 [**Software/Program Verification**]: Model Checking;

General Terms
Design, Languages, Verification.

Keywords
Web Service Compositions, Choreography, Model-Checking, BPEL4WS, WS-CDL.

1. INTRODUCTION
A Web Services Architecture (WS-A)[1] is an emerging distributed software architecture that harnesses the flexibility and reach of the internet with that of extended distributed systems engineering practices. Web Service composition languages, such as the Business Process Execution Language (BPEL4WS)[2], aim to fulfill the requirement of a coordinated and collaborative service invocation specification to support long running transactions and multi-service scenarios. However, a composition alone does not fulfill the requirement of an assured collaboration in cross-enterprise service domains. Participating services must adhere to policies set out to support these collaborative roles in a WS-A with permissions and obligations constraining the interactions between services. Whilst policies are generally considered to be resource access based (e.g. security and access control permissions), obligations are equally important in ensuring collaboration is conducted in an appropriate manner and that the behavior exhibited by participating clients is suitable in a given scenario. This issue is collectively wrapped up in the term Web Service Choreography [3]. In addition the design and implementation of service components in this architecture style must support the original policies as defined by the service owners. These interacting services can be constructed using various emerging standards and managed by multiple parties in their domain of interest and as such the task of linking these activities across workflows within this domain is crucial. Therefore, of clear interest is the need to support such engineering tasks as process verification, partner service usability, and other property checking to verify the roles of web service users and their actions [4]. There is also high value in providing a simulated workflow mechanism to visually compare expected with simulated results of a workflow invocation which can increase expectations of a successful outcome prior to deployment [5].

In this paper, we elaborate on the approach discussed in our earlier work on web service composition verification [6, 7] and illustrate an implementation of the approach in Eclipse to support mechanical engineering tasks that aid designers and implementers of web service compositions.

ICSE'06, May 20-28, 2006, Shanghai, China.
ACM 1-59593-085-X/06/0005.

2. BACKGROUND
Web Service composition languages aim to fulfill the requirement of a coordinated and collaborative service invocation specification to support long running and multi-service transactions. This is seen as an important element of making web services viable for wide spread use, and to provide a closer representation of business transactions in cross-domain enterprises. The effect of using earlier architecture styles has been prone to issues of semantic failure and difficulties in providing the necessary compensation handling sequences [8]. This has been attributed to the strict binding of services with specific technologies. Where previously designers of the workflow had to work very closely with the developers of the technical solution, we now have a mechanism to support technology independent workflow service invocation. This provides an opportunity for the designers to concentrate on exactly what is required from the workflow without hindrance from technical limitations implementation effort. Web Service compositions can also be seen as the implementation layer of a Multi-Stakeholder Distributed System (MSDS) [9]. An MSDS is defined as; "a distributed system in which subsets of the nodes are designed, owned, or operated by distinct stakeholders. The nodes of

the system may, therefore, be designed or operated in ignorance of one another, or with different, possibly conflicting goals". With a service-oriented architecture the focus is on interaction with multiple parties and the behavior could be somewhat ad-hoc depending on the requirements of the partner services. The desirable element here is that to concisely reason about the applicability of a solution we must be able to determine if a distributed solution is correctly orchestrated. Properties to satisfy this verification may consist of a series of questions about the composition; for example; if a request to purchase a product is sent to a partner process, will the process eventually confirm the purchase?

One key part of the verification in this context is to check the trace equivalence with reference to the actions of the design specification, and specifically which order the requests are made and replies sent to the services of the workflow. Whilst there have been other attempts to use model-checking techniques for reliable web service verification, such as in [10-11], there has been little published on the process of using message sequence charts and combining these with translated web service workflow language specifications to verify and validate possible service interactions against those specified in the requirements.

3. THE APPROACH

The approach, illustrated in Figure 1 and fully described in [6], is undertaken as follows. A designer, given a set of web service interaction requirements, specifies a series of MSCs to describe how the services will be used and to model how each service interacts (i.e. invokes, receives or replies) in a given service scenario. The resulting set of scenarios is composed and synthesized to generate a behavioral model, in the form of the Finite State Process (FSP) calculus and then compiled in to a Labelled Transition System (LTS).

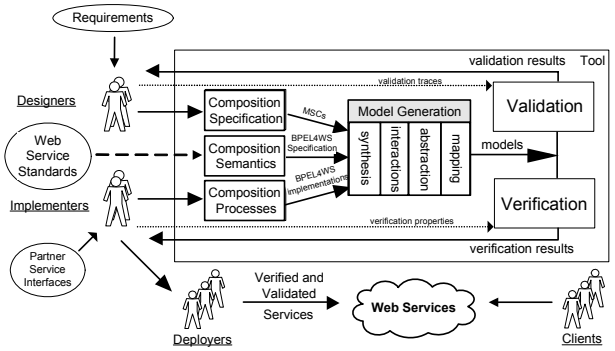

Figure 1 An Approach to Rigorous Web Service Composition Development

The service implementation is undertaken by a BPEL4WS engineer, who builds the BPEL4WS process directly from either specification or requirements. The BPEL4WS implementation and its semantics are used to generate a second behavioral model by a process of abstracting the BPEL4WS with respect to data, and to yield a model of interaction based upon specified semantics applied to BPEL4WS through the FSP algebra. Verification and validation consists of comparing and observing traces of these two transition systems. The approach can assist in determining whether the implementation contains all the specified scenarios of the design and whether any additional scenarios exhibited by the implementation are acceptable to the end-user. In addition, checks can be made on the models with respect to desirable general global properties such as absence of deadlock and liveness (using model-checking). Feedback to the users is in the form of MSCs. The approach is to hide the underlying LTS representations and let the user view only the BPLE4WS specifications or the MSCs as a simple intuitive and visual formalism accessible to most engineers [12].

4. TOOL IMPLEMENTATION

The tool was originally written as a prototype plug-in to the existing LTSA tool suite [13]. This provided the groundwork for a Java implementation that collaborated in other extensions to the suite, such as the Message Sequence Chart editor and graphical LTS Draw functions, and which could contribute to future extensions. LTSA uses the FSP to specify behaviour models. From the FSP description, the tool generates a LTS model. The user can animate the LTS by stepping through the sequences of actions it models, and model-check the LTS for various properties, including deadlock freedom, safety and progress properties. The MSC extension builds on this introducing a graphical editor for MSCs and by generating an FSP specification from a scenario description [12]. FSP code is generated for the architecture, trace and constraint models described previously. LTSA model checking capabilities (for safety and liveness checks) are then used to detect implied scenarios. The LTSA-WS Eclipse plug-in architecture (Figure 2) leverages the previous work and utilises the model-view-controller pattern.

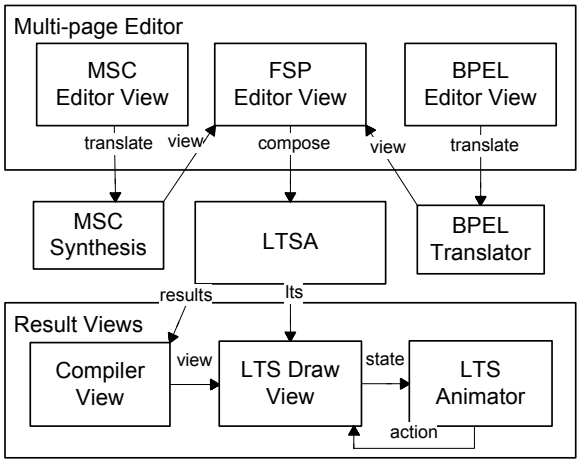

Figure 2 Tool Component Architecture

The service implementation model is the BPEL4WS XML source code, and used managed by editing in the form of a standard XML editor. The model is also parsed to provide useful editor functions, such as content outline and syntax highlighting. Parsing is also performed upon restore or save actions, whereby the translation function is called to view activities specified in the composition. The BPEL4WS engineer is able to build one or many web service compositions which aids in enterprise service decomposition. For each composition selected, the engineer can either translate a single composition or compose multiple compositions for choreography and translate them in to FSP by way of changing editor views. The translation module is written as an independent module (itself potentially a web service), which takes as input a BPEL4WS

implementation and in turn, traverses the source building a representation model in FSP. The mapping semantics from BPEL4WS to FSP has been reported in [6], although a full guide is given as a technical report in [14]. A partial view of a BPEL4WS process for a *marketplace* service is given in Figure 3 and associated translation view in Figure 4.

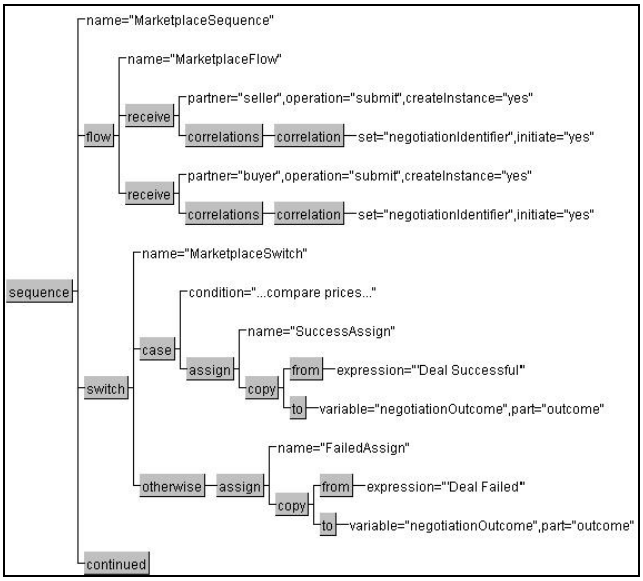

Figure 3 Partial BPEL4WS Process structure for Marketplace Composition

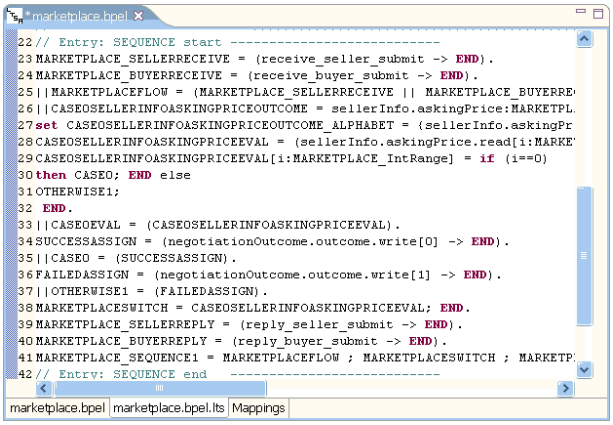

Figure 4 BPEL4WS Translation to FSP View

Multiple composition translation includes interaction mapping by employing a *choreography linking algorithm[14]* to check partner links between services invoke, receive and reply actions. In addition, the MSC specification synthesized to FSP using the LTSA-MSC plug-in can be included in the composed model. To enable this, a visual mapping table is available to the implementer to link activities in design and implementation. Results of checks provide implementers and designers with useful details such as missing interaction cycles (e.g. a missing receive or reply action). Checks are undertaken by the main LTSA function module. An output view summaries actions undertaken by the LTS compiler, and reports on property violations, such as deadlock, liveness or other safety properties. For example, a check on the marketplace process given previously provided the following results on a safety property check:

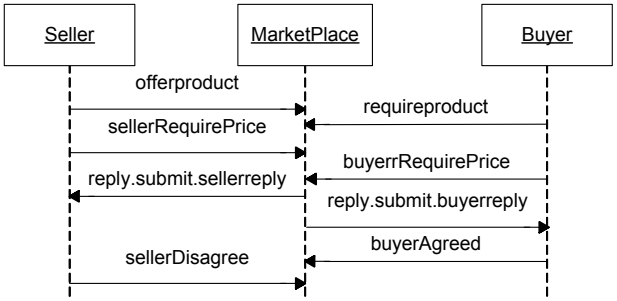

Figure 5 BPEL4WS Process Trace to Violation

From the trace, we can observe that through the sequence of requests, the buyer is able to agree, yet the seller is able to submit a disagreement. This is in breach of a design property which if a buyer agrees to a price then the seller cannot subsequently disagree. The BPEL4WS engineer can make a change in the BPEL to reflect this, or the designer can introduce a requirement back into the MSC design. Iteration of the MSC synthesis, BPEL4WS translation and property checking provides the main essence of the approach. Another use of the approach is to translate and verify service compositions against WS-CDL documents which is an emerging standard to define permitted service interaction policies between partners. The extensibility of our tool provides for future specification translations to be added and used as additional models for verification of process sequences. Although we have suggested that dead-lock and trace equivalence are the only permitted properties to verify, the user can also specify further properties (such as progress of specific actions in a composition) by manually adding FSP progress definitions before verification. We would like to provide a graphical aid for this as part of our future work desires. Figure 6 illustrates example LTS views of the process models that can be analysed in our tool for both BPEL4WS and WS-CDL.

5. CONCLUSIONS AND FUTURE WORK

We have presented a tool to integrate specifications and implementations for rigorous engineering of web service compositions and their choreography. Using the Eclipse framework opens the potential to link the tool with a network of other Eclipse plug-in contributions and aims to simplify the number of different, bespoke tools used in software engineering as a whole. Indeed, amongst these contributions are commercial BPEL4WS graphical editors, although the reader is invited to browse plug-in web sites as the list of contributors is continuously expanding. The LTSA-WS plug-in and specifically the version supporting BPEL4WS, is being considered for use on a number of medium sized case studies, including within projects undertaken by University College London and the UK Police IT Organization (PITO), and we are keen to evolve the tool to support a growing number of web service standard notations.. BPEL4WS provides a work for forming an XML specification language for defining and implementing business process workflows for web services.

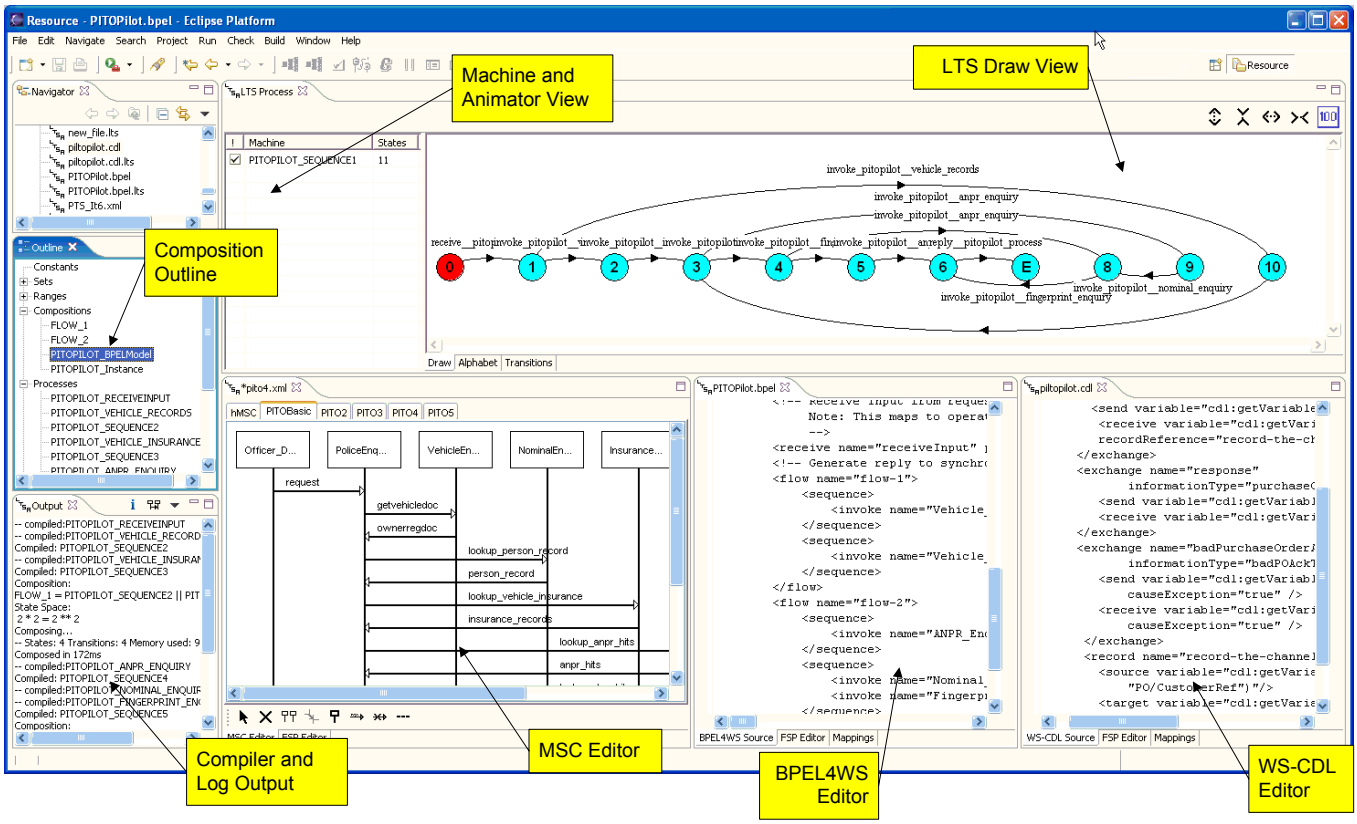

Figure 6 LTSA-WS in Eclipse: View of compositions as BPEL4WS, MSC and LTS Process

We plan to expand the approach to consider dynamic analysis of policies for service interactions in service choreography and also the analysis of service composition deployments on distributed architectures. In this paper we presented an approach towards our goals in the form of a static analysis tool for MSCs, and an implementation tool for equal requirements in BPEL. This work has been funded partly by the STATUS ESPIRIT project (IST-2001-32298), the EPSRC READS project (GR/S03270/01) and by an IBM Innovation Award (2005). The plug-in is available for download from the following web page: **http://www.doc.ic.ac.uk/ltsa**.

6. REFERENCES

[1] W3C-WS-A, "Web Services Architecture (WS-A)," vol. 2004: W3C Working Group Note 11 February 2004, 2002.

[2] F. Curbera, Y. Goland, J. Klein, F. Leymann, D. Roller, S. Thatte, and S. Weerawarana, "Business Process Execution Language For Web Services, Version 1.1," 2003.

[3] W3C-WSCI, "Web Service Choreography Interface (WSCI) 1.0," W3C - Web Services Choreography Working Group 2002.

[4] R. Akkiraju, D. Flaxer, H. Chang, T. Chao, L.-J. Zhang, F. Wu, and J.-J. Jeng, "A Framework for Facilitating Dynamic e-Business via Web Services," presented at OOPSLA 2001 - Workshop on Object-Oriented Web Services, Tampa, FL, 2001.

[5] C. Karamanolis, D.Giannakopoulou, J.Magee, and S.Wheater, "Modelling and Analysis of Workflow Processes," Imperial College of Science, Technology and Medicine, London 1999.

[6] H. Foster, S. Uchitel, J. Magee, and J. Kramer, "Model-based Verification of Web Service Compositions," presented at Eighteenth IEEE International Conference on Automated Software Engineering (ASE), Montreal, Canada, 2003a.

[7] H. Foster, S. Uchitel, J. Magee, and J. Kramer, "Compatibility for Web Service Choreography," presented at 3rd IEEE International Conference on Web Services (ICWS), San Diego, CA, 2004a.

[8] O. Bukhres and C.J.Crawley, "Failure Handling in Transactional Workflows Utilizing CORBA 2.0," presented at 10th ERCIM Database Research Group Workshop on Heterogeneous Information Management, Prague, 1996.

[9] R. J. Hall, "Open Modeling in Multi-stakeholder Distributed Systems: Model-based Requirements Engineering for the 21st Century," presented at Proc. First Workshop on the State of the Art in Automated Software Engineering, U.C. Irvine Institute for Software Research, 2003.

[10] X. Fu, T. Bultan, and J. Su, "WSAT: A tool for Formal Analysis of Web Services," presented at 16th International Conference on Computer Aided Verification (CAV), Boston, MA, 2004.

[11] S. Nakajima, "Model-Checking Verification for Reliable Web Service," presented at OOPSLA 2002 Workshop on Object-Oriented Web Services, Seattle, Washington, 2002.

[12] S. Uchitel and J. Kramer, "A Workbench for Synthesising Behaviour Models from Scenarios," presented at the 23rd IEEE International Conference on Software Engineering (ICSE'01), Toronto, Canada, 2001.

[13] J. Magee and J. Kramer, *Concurrency - State Models and Java Programs*: John Wiley, 1999.

[14] H. Foster, "A Guide to Mapping BPEL4WS to FSP," Department of Computing, Imperial College London, Technical Report 2003b.