

Using a Rigorous Approach for Engineering Web Service Compositions: A Case Study

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

Michael Hu
Police Information Technology Organisation,
22 Upper Ground
London SE1 9QY, UK
michael.hu@pito.pnn.police.uk

Abstract

In this paper we discuss a case study for the UK Police IT Organisation (PITO) on using a model-based approach to verifying web service composition interactions for a coordinated service-oriented architecture. The move towards implementing web service compositions by multiple interested parties as a form of distributed system architecture promotes the ability to support 1) early verification of service implementations against design specifications and 2) that compositions are built with compatible interfaces for differing scenarios in such a collaborative environment. The approach uses finite state machine representations of web service orchestrations and distributed process interactions. The described approach is supported by an integrated tool environment for providing verification and validation results from checking designated properties of service models.

1. Introduction

Our case study is taken from a national development currently underway by the Police IT Organisation (PITO) in the United Kingdom. PITO provides information technology and communication systems to the police service and criminal justice organisations in the UK. PITO's vision is "to be a trusted and valued partner in the delivery and operation of information and communication solutions to meet the needs of the police service and its partners and stakeholders". In this way, one of the highest priorities for PITO is its ability to provide secure, reliable, and available services on demand and to provide highly accurate information as a part of the processes carried out to fulfill service requests. In this project the view is to consolidate distributed national police services and to form a set of core processes by which the national police force may use without directly connecting to separate data sources in the process. Our work runs alongside reported findings so far in the progress of the project, detailing the consideration of moving to a service-oriented architecture and its quality of service provision and expectations [1, 2]. Our contribution is to assist in analysing the initial development of web service

compositions, to support a series of different police enquiry types.

We present here a study of some scenarios described within the scope of interacting police enquiry services. We apply the approach described in our earlier work [3, 4] to concisely model the interaction behavior of the compositions built from the scenarios gathered as part of a business requirements building exercise. The scenarios given in this example case study are representative of the interactions used in providing solutions to the business requirements gained, however, due to the nature of the business of PITO and sensitivity in the detail of systems, these examples remain representative and may not illustrate exact developments.

2. Background

A series of services, such as those scoped in the PITO project, clearly requires management and coordination. If this coordination is implemented as a process in a standard, such as the Business Process Execution Language (BPEL4WS) [5] or other web service orchestration language, then the implementation needs to be constructed for a series of differing scenarios and verified and validated thoroughly for desirable execution paths in each. The use of web technology for services provides an example of how flexible distributed system computing has become. From a specification perspective, the focus is on appropriate service interactions, yet it is equally important to build the web service compositions correctly for all service actors and more importantly, verify this process before deployment is undertaken. This issue is particularly important in sensitive and critical system domains such as civil, emergency and other national infrastructure services.

Our approach, fully described in [3, 4] is undertaken as follows. A designer, given a set of web service requirements, specifies a series of Message Sequence Charts (MSCs) to describe how the services will be used and to model how each service requests or receives a reply in a series of service scenarios. The resulting set of scenarios is synthesized to generate a behavioral model in the form of a series of Finite State Processes (FSPs) [6].

The service implementation is undertaken by a BPEL4WS engineer, who builds the BPEL4WS process from either specification or requirements. The BPEL4WS specification is used to generate a second behavioural model (by way of mapping BPEL4WS to FSP) through a process of abstracting the BPEL4WS, with respect to data, to yield a model of interactions.

Verification consists of comparing and observing states of these two transition systems, whereas validation can assist in determining whether the implementation contains all the specified scenarios and whether any additional scenarios implied 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 user is in the form of UML style MSCs. The approach is to hide the underlying Labelled Transition System (LTS) representations and let the user view only the BPEL4WS implementations and the MSCs as a simple intuitive and visual formalism accessible to most engineers (Uchitel and Kramer 2001). The approach described is illustrated in Figure 1.

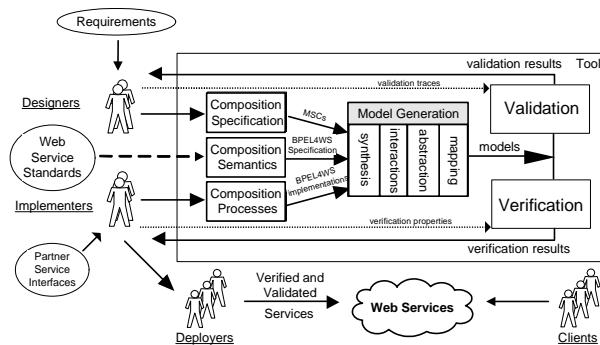


Figure 1 An Approach to Rigorous Web Service Composition Engineering

The activities of the approach can be summarized into a series of steps undertaken by the designer, the composition engineer and a tool we have constructed to assist in development. These are outlined in Table 1. Our work in the approach builds upon earlier reported work for MSC synthesis [7], FSPs [6] and the BPEL4WS translation to FSP given previously. We discuss how our approach differs from other work in the same area in the evaluation section of this paper.

3. Scope and Requirements

The scope of the compositions to date consists of an orchestration of a number of web services implemented to support some basic enquiry types of PITO systems. The position of PITO is to have core enquiry processes running on a central business process architecture (for the

Table 1 Activities in the Approach

Step	Product	Performed by
MSC Design Specification	MSC model	Designer specifies MSCs. Tool generates model.
BPEL4WS Source	BPEL4WS model	BPEL4WS Engineer builds composition process. Tool translates process to FSP model.
Abstraction of interactions	Refined model, for interactions	Tool mechanically abstracts from BPEL4WS to provide refined model of interactions.
Mappings for model refinement	Composition interactions mapped to design model	BPEL4WS engineer assigns interactions to design specification using tool mapping functionality.
Properties for Verification and Validation	Equivalence property added to as additional process model	Tool option to run trace equivalence of BPEL4WS implementation against MSC specification.

deployed web services) which interact with other services provided by local police force system owners. These central *service enquiry compositions* form key interactions to providing a distributed yet consolidated view of the data representations and business processes spread throughout the organisation and its associated forces.

The initial pilot project consists of a series of web services providing functionality for; “Vehicle Enquiry” – matches vehicle details based upon enquiry search criteria, “Motor Insurance Enquiry” linking vehicle details with motor insurance details, “Nominal Enquiry” – matches person details based upon search criteria, “ANPR Enquiry” – provides primary vehicle identification given a Automotive Number Plate or recognition image pattern, and “Finger Print Enquiry” – provides DNA or Finger Print matches to Nominal details. Collectively the scope of the pilot architecture is as illustrated in Figure 2.

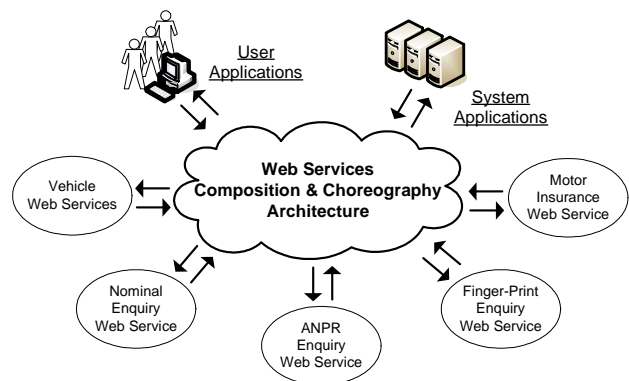


Figure 2 PITO Web Services Architecture Scope

Within this scope is an aggregation of data services and the pilot project considers how these services can be combined, what the required behaviour of such compositions should be, and how these compositions fulfill one or more goals in a series of web service choreography specifications. The pilot project’s initial

requirements were based upon the core function of facilitating a police officer’s enquiry. The scope of this sample is to address vehicle checks with that of the owner’s collective *state* in the police IT network. The basis for this scenario is that when a one officer request is made, a series of requests to various services is undertaken to facilitate building an overall view of required data to assist the officer make a informed decision quickly and on demand.

Initially, the context of a request is built with a variety of linked information sources, with each request supplying key information for the next request. A textual representation of the requirements is given in Figure 3.

A suspicious vehicle with the number plate “xxxxxxx” has been identified by a police officer in Northern England. The officer launches a formal police enquiry about the vehicle including its registration record, insurance details, the registered owner’s criminal records (if any) and DNA/fingerprint of the owner, as well as checking the vehicle’s movement in the last 24 hours at key points in Scotland.

Figure 3 Initial Pilot Project Scenario for Web Service Composition in PITO

4. Interaction Specifications

A first attempt specification of this scenario is built by abstracting the interaction components from the requirements. In this case, the officer makes a request through some *device* (whether it is a Personal Data Assistant, Internet enabled Phone or locally via a Personal Computer). The interactions are then added to support the steps described in the scenario. The composition service in this scenario is formed from a single “police enquiry” component. Initially, the only client in the scenario, the officer’s device component, simply has two interactions, for that of making a request and receiving a reply from the composition service. For each enquiry the police enquiry composition makes a request using key search criteria (such as vehicle registration no.). A Basic Message Sequence Chart (bMSC) view of this specification is given in Figure 4, and illustrates the interaction invocations and replies from given services in the pilot project. The specification illustrated is quite simple, in that it assumes that each enquiry is performed sequentially from the composition enquiry process (i.e. the central service) and that alternative scenarios are not possible. Studying this sample however, highlights possible areas of composition improvement through concurrent service behaviour (which is also a “speed” goal for quality of service in this case study), for example both vehicle records and vehicle insurance enquiries use the vehicle registration details concurrently between the vehicle enquiry and insurance enquiry services.

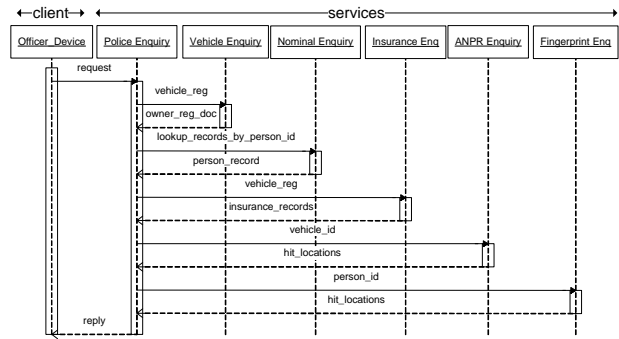


Figure 4 Initial Specification for a Police Enquiry Web Service Composition

This elaboration of requirements yields additional scenarios for the specification. The amendment to the original specification is focused on providing additional scenarios of permissible sequences of interactions following the initial enquiry request. The concurrency requirements poses a series of further questions over that of the behavior constructed initially. Firstly, how does the introduction of this concurrency effect the remaining interactions of the composition? Clearly, by introducing the possibility that two service requests can be performed concurrently suggests that a series of actions may or may not occur within the duration of these initial service requests. Furthermore, can the response from each of the service requests based upon vehicle registration exist over the entire duration of the composition? A partial answer to this question can be found by revisiting the initial specification and identifying that a vehicle id is used in another service request, that is, to the ANPR Enquiry service. Therefore, it is evident that one or both of the two vehicle based service requests must be linked to provide the composition with sufficient detail to pass as parameters to the ANPR service call. Considering this leads the designer to enhance the specification to include the possibility of either the Vehicle Enquiry or Vehicle Insurance Enquiry replying and subsequently the ANPR Enquiry service being called.

We illustrate this in further scenarios, such as that for a Vehicle Enquiry request followed by a Vehicle Insurance Request, then a Vehicle Enquiry reply, then an ANPR Enquiry request and a Vehicle Insurance reply etc. This is illustrated in Figure 5. Further consideration of the initial specification highlights that there is another constraint required for other possible concurrent interactions. The Nominal Enquiry is requested with person identification as part of its required parameters. This identification is also taken from the result of either Vehicle or Insurance Enquiry service request. The two services based upon a person, being the nominal and DNA/fingerprint enquiries must be linked with this initial request if an improvement to have these performed concurrently is achievable.

We can therefore, equally define this constraint as being that the Nominal Enquiry request can not be invoked unless the Vehicle Enquiry has been completed. The revised specification consists of a series of individual scenarios linked together with a scenario composition diagram specified in a hMSC. Starting with a series of basic interactions and formulating an elaboration through consideration of a series of scenarios yields a higher level sequence chart to compose these scenarios together (Figure 6). The interactions can be concurrent (where permissible) and composed in one or many specifications. At each point in constructing the bMSCs or hMSCs, the designer can synthesize the specification to a behaviour model and compile this into an LTS using the LTSA tool [6]. An example LTS for the hMSC shown, is illustrated in Figure 7. We now consider how the case study would implement such a web service composition in one or more collaborating BPEL4WS processes.

5. Implementations

The implementations in this case study take the form of BPEL4WS processes. The requirements are built as processes by a BPEL4WS engineer who interprets the necessary composition and choreography forms of the services involved. We consider how the pilot would undertake implementing and verifying these compositions in our approach.

5.1. Web Service Compositions

The BPEL4WS process requirements are focused on the core Police Enquiry service (the central composition component in the design specifications).

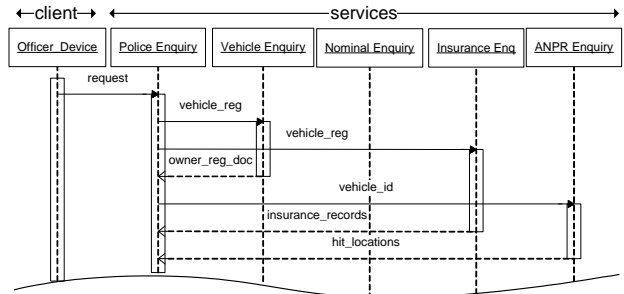


Figure 5 Partial View of a Scenario for Concurrent Requests as part of Specification Elaboration

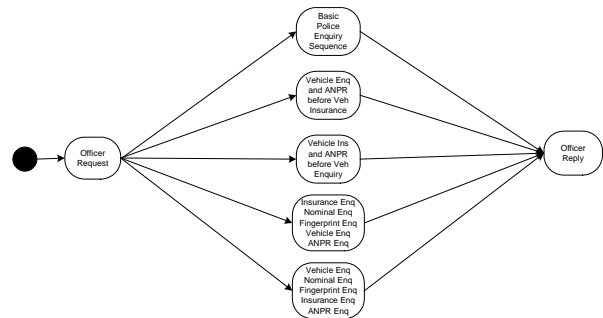


Figure 6 hMSC for Police Enquiry Composition

The process consists of a series of BPEL4WS process statements, specifying the interactions between services and local logic to manipulate message data for these interactions.

To re-illustrate the iterative development in the project using our approach, the implementation begins by building the initial process, where all the interactions are sequenced, as illustrated in Figure 4. The sequence consists of an initial “receive” followed by five invocations, one for each of the enquiry services involved.

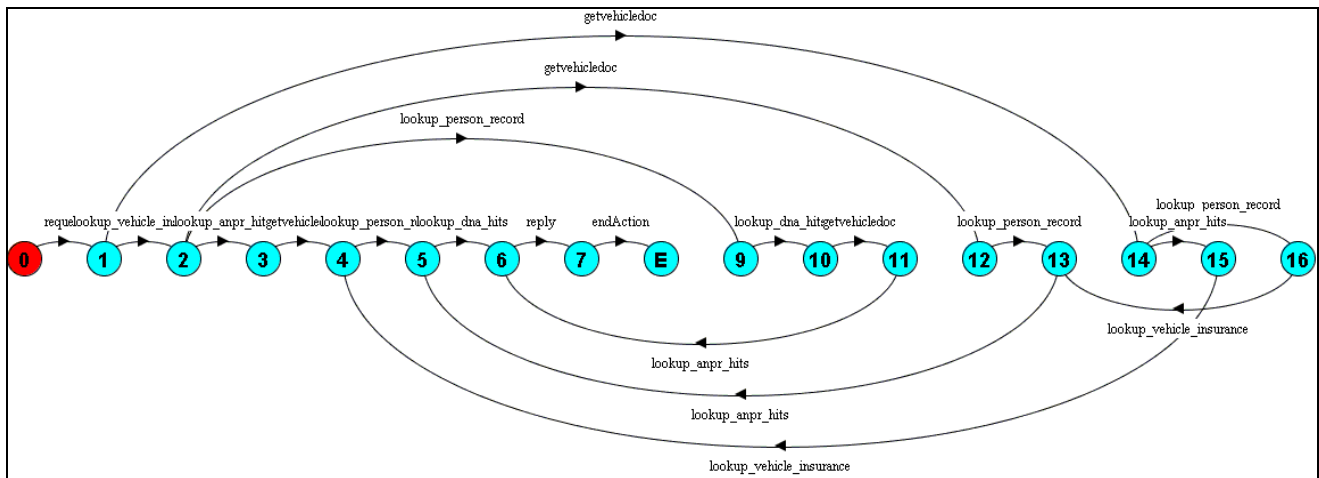


Figure 7 LTS of invocations from PITO Web Service Composition Specification

The form of this process is illustrated in Figure 8, without the BPEL4WS variable assignment statements to manipulate message data for the different interactions.



Figure 8 PITO Police Enquiry Basic BPEL4WS Process structure (interactions only)

Although the assignments are hidden in the illustration, these form the remainder of the process interaction structure, and provide the basis for passing search criteria between service invocations and replies. The initial request message part of vehicle registration is the first to be assigned, to both message variables for input to Vehicle and Insurance Enquires. This basic process is modeled in FSP as a simple sequence of activities. A guide to mapping BPEL4WS to FSP is given in [3]. Each receive or invoke activity, for example the initial request by the officer, is undertaken in the order specified in the sequence. The begin and end of the process is marked by the receive, and a final activity of a reply, before the process terminates. Each new request by an officer creates a new instance of the process, signified by the *createinstance* attribute on the initial receive activity. In addition to the interactions, the process composition is also modelled with variable assignments.

Whilst we could attempt to perform full trace equivalence against the specification at this point, the sequence is perhaps too trivial to expect any relationship of implementation against design. Yet even at this stage in implementation, this implementation may be partially fulfilling the specification through one such scenario (in this case the basic sequenced scenario). The composition engineer then specifies the mappings between the interactions modeled in the composition with that of the design specification to be verified against. The mappings are selected against a list of activities presented in our tool. We therefore use the trace equivalence verification method to perform such a check at this stage. The preparation of the implementation process is also discussed in [3]. To perform trace equivalence the non-interaction activities are marked as being non-observable in the implementation model. In this case, we use the FSP hide operator of these activities. A first check trace that is listed in Figure 9, has highlighted that the BPEL4WS does not exhibit the behaviour to support a Vehicle Insurance Enquiry immediately after a request (i.e. the expected *lookup_vehicle_insurance* transition).

The engineer may wish to discuss this with the designer, however, with our initial knowledge of the specification's concurrency scenarios, the project is keen that either service may be initiated first so that concurrency of activities may be utilised to increase performance of the composition.

```

/* Trace run of equivalence property check
of BPEL4WS process over MSC */
Composition:
CheckMSC = BPEL4WSModel || MSCModel
Trace to property violation in
BPEL4WSModel:
    request
    lookup_vehicle_insurance
Analysed in: 10ms

```

Figure 9 Results of Trace Equivalence Check for Scenarios not covered in BPEL4WS Composition

The BPEL4WS engineer therefore revisits the composition again and studies how this requirement can be implemented. Further concurrency analysis suggests that linking of activities is required to fulfil the remaining set of scenarios, for example, detailing the constraint that the ANPR Enquiry may follow Vehicle Insurance or Vehicle Enquiry service invocations. The engineer therefore adds a *source link* to the Vehicle Enquiry and a target link to the ANPR Enquiry service invocation activity. Another link is considered against the Fingerprint/DNA Enquiry. This can only occur if the Nominal Enquiry has completed. The engineer completes the link constraints by placing a *source link* on the Nominal Enquiry invocation and related *target link* on the Fingerprint Enquiry invocation. The process is completed with a concurrent (FLOW) wrapper around the invoked services and links. The updated, potentially a first release composition, is illustrated in Figure 10.

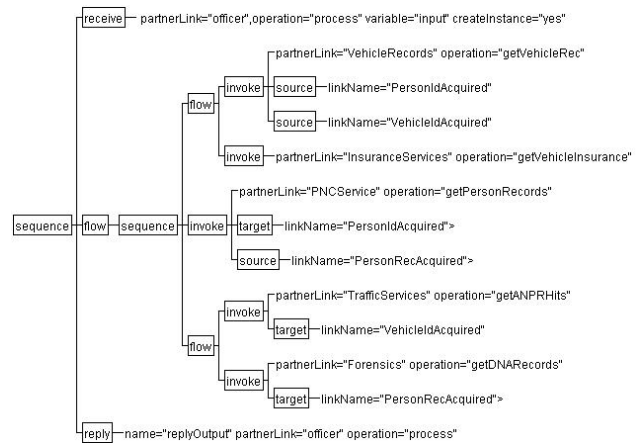


Figure 10 BPEL4WS Process with Links

A compiled LTS model of this process is also given in Figure 11.

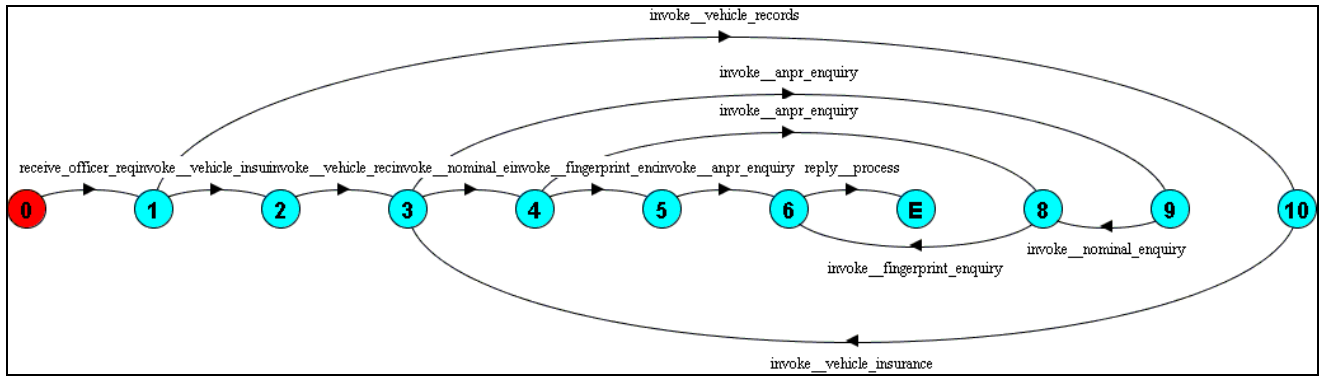


Figure 11 LTS of BPEL4WS Implementation for a Police Enquiry Composition

5.2. Web Service Choreography

In essence, web service choreography defines a kind of policy for “rules of engagement” in conversations implemented between partners in web services architecture [8]. It defines the global interactions necessary to be controlled and enforced when two or more services are interacting to fulfill a common goal in a process. The police enquiry composition interacts with other services, themselves potentially compositions. The choreography aspect of this pilot project therefore focuses on the underlying engagements between compositions and expands on the original requirements described earlier. Here we consider how these compositions can be verified together to form a view for service choreography policy verification.

Addressing choreography takes us back to the designer, who may reuse existing composition scenarios to act as a source for interactions observed in other compositions. For example in the ANPR Enquiry (used as part of the vehicle movement checking requirement) the ANPR Enquiry service may consist of other traffic related service enquiries. Initially, the only assumption is that the service will eventually reply to the Police Enquiry. However, as we highlighted, choreography provides a global view of requirements for one or more scenarios – such as in this case, that the ANPR does eventually reply to the Police Enquiry, but also that its own partnered services equally complete the interaction cycle. Hence, the engineer requires greater confidence in the composition working alongside other partnered compositions. As choreography describes one or more global goals and an elaborated understanding of global interaction state, we introduce a third service into the PITO Police Enquiry requirements. The requirements are expanded to include an *authorisation* service which holds state of enquiry requests and provides a control on which services may be accessed in an enquiry type. We introduce the “Authorisation Service” composition into the choreography, by which each enquiry service must request authorisation before proceeding to the next in the

officer’s request. Reusing the composition for Police Enquiry and ANPR (Vehicle Movement) compositions, our domain of interest is depicted as in Figure 12.

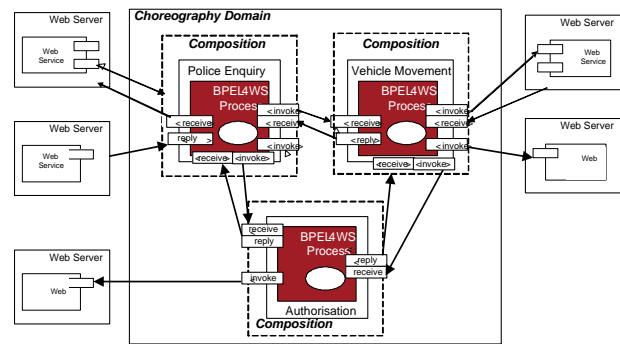


Figure 12 Overview of choreography in elaborated composition scenario

Now the designer specifies this new requirement in another scenario. The choreography is spread across components in the specification, with interactions occurring between multiple-parties, such that the police enquiry is not the single focus. The design specification for this choreography scenario is illustrated in Figure 13. Notice that for each enquiry request, an invocation of the authorisation service is undertaken. The reply result from the authorisation is taken that the request has been granted in the current request’s state. For simplicity, the designer has specified only one scenario for this choreography, and as such the behaviour of all compositions and services within the implementation of this choreography must exhibit behaviour suitable for this interaction sequence. As with the composition implementation we build the compositions supporting this requirement in BPEL4WS and then use verification, specifically “compatibility verification” (which we discuss later), to ensure that the behaviour of these collaborating compositions is suitable to fulfil the requirements in the design specification.

The BPEL4WS compositions now consist of a Police Enquiry process, a Vehicle Enquiry process, ANPR

Enquiry process and Authorisation process. Again, we have simplified the processes to support this scenario as a base for creating extended processes supporting other enquiry types. The choreography scenario used here is a subset of the interactions built in the process used previously.

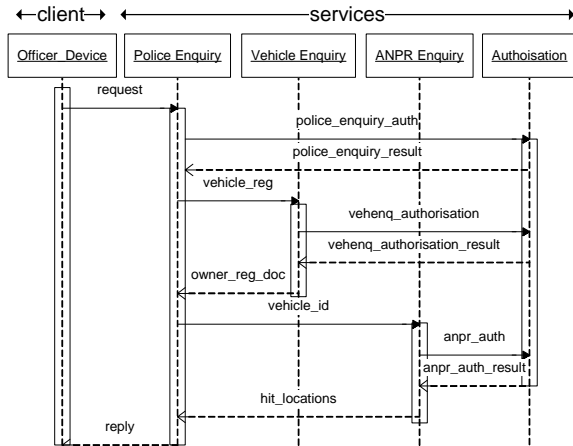


Figure 13 A Specification for Choreography of Police, Vehicle, ANPR and Authorisation Compositions

It is the case that this additional scenario may be simply included in the current process, with the use of a BPEL4WS SWITCH statement to distinguish which type of Police Enquiry is undertaken. However, for clarity we build a new Police Enquiry composition process outlining just the interactions in this scenario as in Figure 14, with the distinct additions of *invoke_enquiry_auth* and *invoke_enquiry_result* activities for each service invoked.

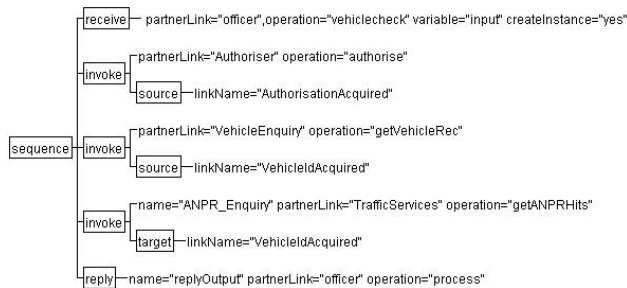


Figure 14 Partial Police Enquiry Composition within Choreography

The choreography model is generated by employing the use of port connector models for modeling the *conversations* of web services (linking invoke with a receive and reply). A generic port connector is illustrated in Figure 15. For every composition process selected for modelling we extract all the interaction activities in this process. As mentioned previously, interaction activities are service operation invocations (requests), receiving operation requests and replying to operation requests. In addition to an invocation request, we also add an

invocation reply to synchronise the reply from a partner process with that of the requesting client process. The list is then analysed for invocation requests, and for each one found a partner/port lookup is undertaken to gather the actual partner that is specified in a partnerlink declaration.

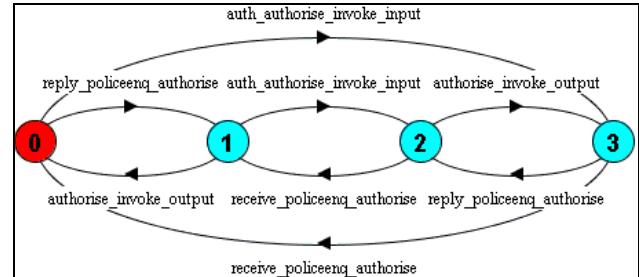


Figure 15 Example Mapped Port Connector for Composition Interaction Compatibility Checking

To achieve this, a partner list is used and the partner referenced in invocation request is linked back to a partnerlink reference. The partnerlink specifies the porttype to link operation and partner with an actual interface definition. To complete the partner match, all interface definitions (in the form of WSDL documents) used in composition analysis are searched and matched on porttype and operation of requesting client process. This concludes the partner match. A port connector bridge is then built to support either a simple request invocation (with no reply expected) or in “rendezvous” style, building both *invoke*→*receive* and *reply*→*invokeoutput* models. This supports the synchronous model mapping. The sequence is then repeated for all other invocations in the selected composition process, and then looped again for any other composition processes to analyse. To check compatibility of these compositions against the referenced ANPR and Authorisation compositions we perform a safety analysis of the choreography model, analysing it for deadlock freedom. The result of such verification is illustrated in Figure 16. The reason for this deadlock is suggested in the trace of two invoke actions without a reply from the first – in this case the engineer refers to the Vehicle Enquiry composition and observes that there is no reply action specified and amends.

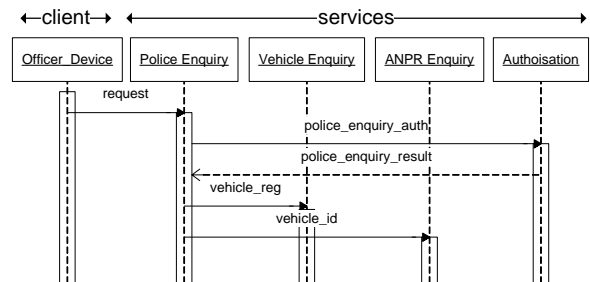


Figure 16 Deadlock Example of Compatibility Verification BPEL4WS and Partnered Compositions

Consequently, the port connector models cannot be synchronised and a trace to deadlock is observed in the verification. To solve this issue, the engineer can add a reply and complete the model. A subsequent compatibility verification of the choreography provides the engineer with a suitable “no deadlocks/errors found” successful result. Again, we can also perform the trace equivalence of this composed model against the specification by repeating the method described in the composition analysis example previously.

Note that we have not given an example of mapping port activities to the actual service invocation cycle activities, yet the cycle of invoke, receive and reply are taken from each invocation and relabeled appropriately. The results obtained from both composition and choreography checking highlight errors in synchronization. We now consider how this case study was undertaken and evaluate the approach.

6. Analysis and Evaluation of Approach

6.1. Review of Tasks and Results

The progress of this case study has resulted in four iterations, for both designer and BPEL4WS composition engineer. We specifically managed this progress to illustrate the concepts of compositions and the effort required to address issues that arise when building for a service-oriented distributed set of processes. The results of using this approach have provided benefit in terms of verifying that implementation interactions, specified either completely or partially, fulfill a set of specification scenarios and that where a mismatch occurs, the engineer follow traces to readdress how compositions can fulfill requirements. There is also potential to measure the impact of service changes, and affected compositions that interact with these services. The case study will likely be expanded to consider maintenance at a later stage in the project.

We have taken a number of assumptions in both composition and choreography modeling. Key assumptions taken are that we model from a static analysis perspective, taking less focus on the message values and how they affect composition interaction behavior. Associated with this is the ability to perform dynamic checking of choreography policies, perhaps written in the Web Services Choreography Description Language (WS-CDL) standard [9], with particular emphasis on obligations necessary by a service to contribute to a choreography role. We model the collective behavior by synchronizing on basic interactions such that a complete model may be analysed for interacting service compositions.

In terms of performance and resources used to undertake modeling and verification, we recorded a final

composition implementation model consisting of 41 states and 101 transitions. We noticed an expected increase in size of design specification models from the start until the end iteration, and equally when implementations were evolved to support an increasing number of scenarios implied by the initial requirements (although this may not necessarily be the case where further scenarios add more constraints to the composition behaviour). The models used for tracing are generally smaller than those without synchronization and label mappings used to restrict the number of states and transitions. We have also performed the approach using a number of theoretical composition examples provided for in IBM’s Alphaworks BPWS4J (a BPEL4WS process engine). Amongst the examples, is a more complex process for an Automated Teller Machine (ATM) which defines a set of interactions for logon, disconnect, logoff, withdraw and credit operations with the process itself as the ATM. A modeling approach resulted in a machine of approx. 157 states and 428 possible transitions. We are actively pursuing larger and more complex processes to test the approach and tool. We believe that the process is limited however to performance of the state- of-the-art model checking tools.

6.2. Ease of Learning

Our approach and the tool built to support it, aims at providing the following criteria for its ease of learning and carrying out verification and validation. Firstly, the design specifications are based on the scenario approach, and the use of bMSC and hMSC sequence charts is widely undertaken and understood in industry. Where other work has concentrated on specifying formal algebraic notations for specifications, we provide a graphical interface so that the user does not have to learn these sometimes complex notations. The implementations are constructed either directly in the tool or through a third-party tool such that the BPEL4WS engineers are not restricted in a particular editor implementation or feature list to use our approach. Indeed, there is already evidence of several BPEL4WS editors in the Eclipse development environment. Verification properties can be specified by reusing the approach for building design specifications. For example, the designer, in addition to building a complete set of service scenarios for equivalence verification, may also submit further safety or liveness properties by way of constructing bMSCs that specify these individual requirements. Validation is undertaken through an animated label transition system interface. Validation can be undertaken with two views. Firstly, the designer can animate their design specification and make initial assessment of what composition interactions should occur, in which order and in relation to other compositions for choreography scenarios. Secondly, the BPEL4WS engineer can verify implementations of

composition process interaction through the same interface. Either party can examine counterexamples generated by the verification steps in our approach through this interface.

6.3. Early Payback

Early payback is a key objective of the approach. Through the motivation for this work, our aim is to support answering questions highlighted by the distributed nature of web service compositions and on consideration of the pattern by which these compositions may interact. Clearly, the benefits of using such an approach will require early feedback to the developers so that greater assurance can be given, in both design and implementation activities, as earlier as possible. We achieve this by separating the tool between design and implementation, and consolidate their output to provide another view for analysis. The iterative development does not necessarily suggest that service compositions will be designed and implemented in isolation; moreover, we believe that compositions will be part of much wider developments cross-enterprise and between several development teams. An example of this is from our Police Enquiry case study. The initial specification and composition design suggested a simple sequence of interactions between one compositions interacting with up to five partnered services. Further elaboration of the scenarios possible from that composition illustrated that the other partnered services may also be compositions. Indeed, one of the last elaborations in the case study suggested that several compositions all communicated with an “Authoriser” composition. Clearly, it can be seen that several design specifications and implementation models may be used in this case study. By providing early feedback, in terms of verification and validation, these types of projects can resolve local differences and yet at the same time consolidate global requirements for choreography scenarios.

6.4. Orientation towards error detection

The essence of our approach is to highlight inconsistencies between interactions specified in composition implementations against that of those given in design specifications. We do not aim to clarify notational and specification semantic correctness, although that can be achieved to a degree by user validation through animation. We assume that correctness of implementations is a given attribute of the inputs submitted for observing errors against design specification scenarios, and as such, this provides orientation of our approach towards error detection rather than correctness of these artifacts.

6.5. Integrated use

From a technical implementation perspective, we wished to provide the tool as much as a reusable service as that of which it is used to verify. In this way, we have scoped the architecture for the tool to be extendable and integrated without a presumption of which interfaces would be used to build the inputs to the tool core. In other words, the BPEL4WS engineers are free to build the compositions in any supporting editor, yet on the one condition that the output from these editors conforms to the same specification supported by our tool. The tool was originally written as a prototype plug-in to the existing LTSA tool suite [6]. 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[7]. FSP code is generated for the architecture, trace and constraint models described previously. LTSA model checking capabilities (safety and deadlock freedom checks) are then used to detect implied scenarios. The LTSA-WS plug-in for Eclipse (Figure 17) is built in the commonly known model-view-controller pattern.

We also do not believe in forcing a new methodology upon developers by way of the tool, but aid in various methodologies for the tasks that must be undertaken regardless of the actual steps of a methodology e.g. verification and validation can be undertaken in either analysis, design, implementation or maintenance.

7. Conclusions

We plan to expand the approach to describe properties by way of the Web Services Choreography Description Language (WS-CDL). The use of such standards provides an example of how distributed system computing using web services will be specified for web service choreography invocations, yet it is important to compose the service workflow correctly for all service actors and more importantly, verify this flow before actual implementation and deployment is undertaken. Further work is also required with respect to transactional modeling and to verify this against compensation routines between processes, as related faults must explicitly be mapped to known fault naming conventions.

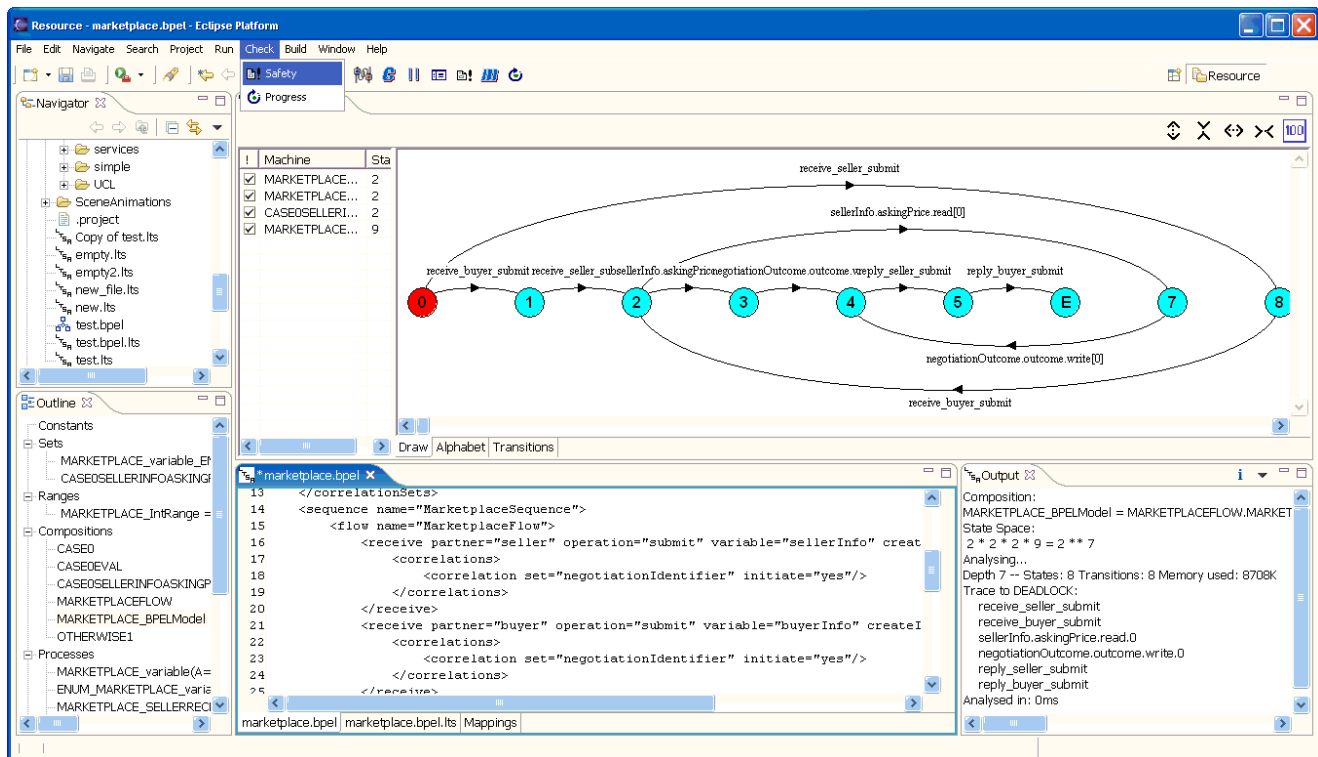


Figure 17 Eclipse plug-in for Web Service Composition (BPEL4WS) Verification

We believe that in parallel with work such as WS-CDL for choreography, WS-Transaction for transaction handling and WS-Coordination/WS-Policy for profiling, roles and service obligations during web service conversations there is sufficient need for a verification process to confirm that designs and implementations will interact correctly once deployed. By combining these models with the complete system model, the additional verification of these routines is incorporated into our approach.

In this paper we presented a case study using an approach aimed towards this goal in the form of MSCs, and an implementation of equal requirements in BPEL. We are still in the process of migrating the complete suite (including MSC editor) to Eclipse. The authors would like to acknowledge that this research was supported, in part, by the STATUS ESPIRIT project (IST-2001-32298), the EPSRC READS project (GR/S03270/01) and by an IBM Innovation Award (2004). The authors would also like to thank the contributions to this work given by the PITO organization and members. The plug-in is available for download at <http://www.doc.ic.ac.uk/itsa>.

8. References

- [1] M. Hu, "Quality of Service Composition and Factoring In Composite Web Services Based Business Process," presented at XML Conference 2004, Washington D.C., USA, 2004.
- [2] M. Hu, "Web Services Composition, Partition, and Quality of Service in Distributed System Integration and Re-engineering," presented at XML Conference 2003, Philadelphia, PA, 2003.
- [3] 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.
- [4] 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.
- [5] T. Andrews, F. Curbera, Y. Golland, J. Klein, F. Leymann, D. Roller, S. Thatte, and S. Weerawarana, "Business Process Execution Language For Web Services, Version 1.1," 2003.
- [6] J. Magee and J. Kramer, *Concurrency - State Models and Java Programs*: John Wiley, 1999.
- [7] 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.
- [8] D. Booth, H. Haas, F. McCabe, E. Newcomer, M. Champion, C. Ferris, and D. Orchard, "Web Services Architecture (WS-A) - W3C Working Group Note 11 February 2004," vol. 2004: W3C Web Services Architecture Group, 2004.
- [9] N. Kavantzias, D. Burdett, G. Ritzinger, T. Fletcher, and Y. Lafon, "Web Services Choreography Description Language Version 1.0 - W3C Working Draft 17 December 2004," 2004.