Perspectives on Data Flow-based Validation of Web Services Compositions

Antonia Bertolino, Eda Marchetti
ISTI-CNR
Via Moruzzi 1 - 56124 Pisa
{antonia.bertolino, eda.marchetti}@isti.cnr.it

Ioannis Parissis
Laboratoire d’Informatique de Grenoble
BP 53 - 38041 Grenoble Cedex 9
ioannis.parissis@imag.fr

ABSTRACT

Composition of Web Services (WSs) is anticipated as the future usual way to dynamically build distributed applications, and hence their verification and validation is attracting great attention. The standardization of BPEL as a composition language and of WSDL as a WS interface definition language has led researchers to investigate verification and validation techniques mainly focusing on the sequence of events in the composition, while minor attention has been paid to the validation of the data flow exchange. In this position paper we schematically settle some future research issues on the perspectives opened by data flow-based validation of composite web services.

1. INTRODUCTION

The emerging architectural approach to develop and execute applications distributed over the network is Service Oriented Architecture (SOA). By facilitating loose coupling between independently developed components, SOA provides a very flexible and cost-effective software paradigm, allowing for the reuse of existing applications and for the integration between heterogeneous systems and technologies. As known, SOA applications are built based on services and this is mainly done by the composition, via standard interfaces and protocols, of existing services into new service compositions (or composite services), so giving rise to new, more powerful functionalities.

Compositionality is together one of the most promising aspects of this architecture and a major issue. How to ensure the compliance of the composite services with the initial requirements is a compelling and challenging research problem. In fact, unlike traditional component integration, which is carried on at assembling time, service composition can be performed at run time, so that dynamic discovery and run-time integration become first level concepts of the SOA paradigm. A concrete instantiation of the SOA are Web Services (WSs), which are implemented and composed by using established Web-based open standards, like WSDL, UDDI, HTTP, SOAP, and others.

The verification and validation of composite WSs, which would correspond to the integration stage of the more conventional testing process, in this new architectural standard raises several issues [3]. One major issue is that the implementation of the WSs involved in the composition is generally unknown. Their interfaces are specified by the WSDL standard description language, but there is no behavioral specification available. Testing can only be done under assumptions on this behavior. The dynamic binding of services makes it impractical to test in advance all the concrete service combinations that can be involved in a workflow. Heuristics must be used to reduce the amount of test executions. Furthermore, testing a service composition increases the workload of the involved services (that may also be used in the same time in other compositions) and also the cost of testing (using the services may not be free of charges). The same kind of problems occur for testing non-functional attributes of an architecture, such as quality of service (QoS).

In the recent years, several attempts have been made to adapt existing validation approaches to the validation of WSs and WSs composition. They mainly use behavioral specifications, extracted from the BPEL description of the composition, as an input to formal verification. However, proving the behavioral correctness of the system may not be sufficient to guarantee that specific data properties are satisfied [13]. But so far little attention has been paid to data validation or even modelling for WSs. Most of the times data-flow requirements and data properties are just informally expressed in natural language, and consequently cannot be adequately verified.

Data modeling represents an important aspect to be considered during the implementation of a composition, as data-flow relationships and requirements provide an alternative view of the composition problem with respect to the functional-oriented view, which should be taken in consideration during both the implementation and the testing phase.

Of course the usage of data information for verification purposes is not a novelty. Several data-flow oriented test adequacy criteria have been proposed in the past. The purpose of this paper is to explore the possible ways of exploiting data information for the validation before and during the implementation of a WSs composition.

Permission to make digital or hard copies of all or part of this work for personal or classroom use is granted without fee provided that copies are not made or distributed for profit or commercial advantage and that copies bear this notice and the full citation on the first page. To copy otherwise, to republish, to post on servers or to redistribute to lists, requires prior specific permission and/or a fee.

Copyright 200X ACM X-XXXXXX-XX-X/XX/XX ...$5.00.

99This research work has been partially supported by ART DECO (Adaptive InRasTructure for DECentralized Organizations), an Italian FIRB (Fondo per gli Investimenti della Ricerca di Base) 2005 Project.
2. RELATED WORK

This section is a brief overview on recent investigations on WSs validation either regardless of composition issues or addressing specifically WSs composition or focusing on fault or failure models for WSs compositions. In the Coyote framework [16], test data are selected among monitored data and other manually produced test data, according to their fault detection ability. The latter is assessed on WSs validation either regardless of composition issues or on the structural coverage of the composition specification, considering that it is provided in BPEL [6], the standard language for programming WSs compositions. In [9], a transformation is proposed from BPEL to PROMELA (similarly to [4]). The resulting abstract model is used with the SPIN model-checker to generate tests guided by structural coverage criteria (such as transition coverage).

Compositions of WSs can also be formally verified as soon as formal models of the composition and of required properties are provided. For instance, in [12] workflows are described as Petri Nets and then simulated to verify properties such as reachability. Similarly, a transformation of BPEL processes in Colored Petri Nets (CPN) has been proposed [20]. Another formal approach is proposed in [7]. The workflow is specified in BPEL and an additional functional specification is provided as a set ofMSCs. These specifications are translated in the Finite State Processes (FSP) notation and model-checking is performed. The final goal is to detect execution scenarios allowed in the MSC description and that are not executable in the workflow and, conversely.

Table 2 we provides a classification of the above presented approaches.

The above investigations use models of the composition behavior and of properties or scenarios expressing the user expectations (MSCs or state based properties such as the absence of deadlock). A different characterization of failures of WS compositions is proposed in [17, 18] where failures are considered as interactions between WSs, similarly to feature interactions in telecommunication services, and classified as goal conflict, resource contention, deployment-ownership decisions related problem, assumption violation, information hiding, policy conflict or wrong invocation order.

The underlying models of all the above approaches - workflows, scenarios, user goals, state-based properties - focus on control. The verification of the data transformations involved in the WSs composition execution does not seem to have been explored so far. From the modelling point of view, this lack has been outlined in [11] where dependencies between data exchanged during the execution of a WSs composition are explicitly modeled by means of an ad hoc notation. There is no literature on fault models based on data for WSs compositions. However, we could mention a proposition of data fault model for workflows [13]. According to this model, data can be redundant if they are produced by an activity, but not used by any other activity. They can be lost, if the outputs of two concurrently executed activities are assigned to a single variable in a non deterministic order (so, one of the outputs may be lost) or missing, if an input activity expects data that are not specified as outputs of another activity. They can be mismatched, if the expected input data do not match with the actual data sent to an activity. Inconsistent data correspond to corrupted variables. Data can be misdirected if an activity A expects data from an activity B while A is prior to B in the workflow. Finally, data may be insufficient to complete the workflow goals (this is mainly a specification problem).

3. USING DATA-FLOW FOR TESTING WEB SERVICES COMPOSITION

3.1 Data related models

To illustrate the usefulness of data flow modelling for testing purposes, we refer to a simplified version of the Vir-
tual Travel Agency (VTA) example used in [11]. A VTA service offers travel packages to customers, by combining two independent existing services: a flight booking service (FBS), and a hotel booking service (HBS). HBS receives the date and the location using the ports $H.request.date$ and $H.request.loc$, respectively. The ports $H.offer.cost$ and $H.offer.hotel$ are used for returning the cost and other hotel information.

In a similar way, FBS uses the ports $F.request.date$ and $F.request.loc$ for receiving flight booking requests for a given time period and location, while $F.offer.cost$, $F.offer.schedule$, $F.booked.info$ for returning the cost, the schedule and other flight information. In both cases, the offer can be accepted or canceled through the customer interface, which is a service provided by the VTA and invoking HBS and FBS.

Customers can ask for a travel package providing the dates and location (output ports $C.request.date$, $C.request.loc$) and be informed about the proposed flight ($C.f.offer.schedule$, $C.f.offer.cost$), hotel ($C.h.offer.hotel$, $C.h.offer.cost$) and the package cost ($C.booked.cost$). Figure 1 provides an abstract data-flow model (DFM) expressing dependencies between the flight and hotel offered costs and the cost proposed to the customer. According to this model, the hotel cost offered to the customer ($C.h.offer.cost$) must be equal to the cost returned by the HBS ($H.offer.cost$) and similarly for the flight cost. The whole package cost ($C.booked.cost$) is computed from the selected hotel and flight costs by means of the function $A$ (this function collects costs and may add various fees).

DFMs can be useful in the specification of WSs compositions, since they highlight the goal of the composition from the data point of view. Building a DFM forces modelling the implicit knowledge on data, avoiding loss of information or misdirection of data flow and highlighting the most critical data flow paths. A DFM can also be used, notably, to define test coverage criteria and test strategies. Furthermore, data-flow relationships and requirements provide an alternative view of the problem with respect to the functional-oriented view. Assuming that a DFM is provided for the WSs composition, we explore in the rest of this section the applicable verification and validation approaches that could use this model, and identify challenges for research in this field.

Data fault models, as the classification mentioned in the previous section [13], could also be useful in the validation process. In the context of WSs composition, some of the identified problems in this classification do not apply when the standard languages BPEL and WSDL are used, since these languages ensure that the data exchanged between services conform to a mutual accepted specification (see Table 3). In order to be able to formally identify and to automate the verification of such data problems, a model of the data used within the WSs composition is needed. Hence, data modeling is an important issue in the design, the implementation and the validation of a composition.

### 3.2 Kinds of models referred to

Performing data-flow based validation may involve several kinds of models than can be combined in various ways. In the rest of this section we consider that one or more of the following models may be available:

- A DFM, defined before the implementation of the composition and independently from any behavioral specification of the composition. It expresses dependencies between the data handled or exchanged during the execution of the WSs composition.
- A behavioral model of the WSs composition. It is possible to derive, from this model, a DFM explicitly focusing on the service data interactions. An example WSs composition for the VTA is given in Figure 2 (alternatively, a BPEL process could also be provided).
- A model defining classes of faults related to data (data fault model). Table 3 is an example of such a classification that could be used as a fault model.
- In addition to these models, we can consider properties focusing on data, written in a formal language. While a DFM focuses on dependencies between data,
properties may restrict the domain of the computed values or express relations between them (for instance, \(C.h\_offer.cost + C.f\_offer.cost \leq C.booked.cost\)).

Table 4 summarizes the above mentioned models and highlights how testing could be performed in presence of one or more of these models. A more detailed description of this Table is provided in the rest of the section.

### 3.3 Using a DFM built without any formal model of the composition

This section corresponds to the row 1 in Table 4 where a DFM is provided modelling the service data interaction during the composition built independently from any other model of the composition (as, for instance, a BPEL process).

#### 3.3.1 Using a DFM alone

Consider the case where the only available specification for the composition is a DFM (column 1 in Table 4). This model can be exploited in two different ways for testing purposes:

- **Base for measuring structural coverage.** Coverage criteria adapted to a DFM can be defined. For instance, considering the Figure 1, all the paths starting from the model input \(H.offer\) and ending to a model output \((C.booked.cost, C.h\_offer)\) could be covered.

- **Test data generation.** Once established a coverage criterion, black-box testing strategies can be adopted to derive the appropriate test cases. The objective of such a test data generation could be to demonstrate concrete executions involving the DFM elements to cover.

#### 3.3.2 Using a DFM together with the BPEL process description

Consider that both a DFM and the specification of the WSS composition (BPEL) are available (Table 4 column 2). This information can be exploited for statically checking both if the defined BPEL composition conforms to the data-flow requirements specified in the DFM and if it properly implements all the data dependencies. In the VTA example, it could be checked that the costs provided by HBS and FBS are used in the PreparePackage activity.

### 3.3.3 Using a DFM and a data fault model

As it has been suggested in Section 3.1, a data fault model could be defined mainly focusing on lost or inconsistent data. Once established, the data fault model can be used with the DFM (Table 4 column 3) to seed faults into the latter (for instance, in Figure 1, the outputs \(C.h\_offer\) and \(C.h\_offer\) could be exchanged). Fault coverage criteria can then be established on the DFM and test data can be selected to detect the presence of these faults in the implementation.

### 3.4 Using a DFM extracted from the BPEL description of the composition

Let us consider that a BPEL description of the composition is available (row 2 in Table 4). In this case, exploiting the behavioral description of the BPEL and the WSDL description of the involved services makes it possible to (automatically) extract a DFM. The peculiarity of such a DFM is that it provides another point of view of the composition, focusing on data interaction. The DFM therefore can be exploited as described in Section 3.3.1, but in this case the criteria, and consequently the derived test cases, intend to check the implementation.

A more interesting possibility consists in exploiting the DFM information to define coverage criteria for the BPEL process description. In that case, data-flow information can be combined with the control flow information to apply common data-flow criteria, such as for instance all-defs or def-use [5], and to guide the test data generation to fill the criteria (Table 4 column 2).

### 3.5 Using properties on data

Overall data properties differ from data-flow modelling in the sense that they can express customer expectations on the composition results regardless of the data dependencies or the process execution. Such properties could state that the result of a query must belong in a given interval (“the price of the travel package must not exceed 500 euros”, “the hotel category must not exceed 3 stars for low cost flights”), or establish a relation between the computed data. For instance, the property \(C.h\_offer\cdot C.f\_offer \leq C.booked\) is not explicitly included in the DFM or in the VTA composition of Figure 2.

Such properties could be used in several ways (Table 4 row 3):

- **Black-box testing.** If there is no information available on the WSS composition (Table 4 column 1), black-box testing techniques, such as Category Partition, can be used taking into account the properties to select relevant test data (for example, a low cost flight could be chosen to check that the offered hotel category is less than 4 stars).

- **Data-flow testing.** The specified set of properties can be used together with the BPEL specification (if available) to apply common data-flow based testing techniques focusing on the data involved in the property expression (Table 4 column 2).

- **Coverage criteria.** The set of formal properties can be used in association with a data-flow model or with a data fault model to define test criteria and adapted test strategies ensuring that any fault in the DFM that has
an impact on the property satisfaction will be detected (Table 4 column 3).

4. CONCLUSIONS

WSs are the most prominent example of the emerging SOA. We have discussed several ways in which, depending on the information available, the flow of data in composite WSs could be usefully referred to for Verification and Validation. Although the verification of WSs compositions assumes strategic value, our impression is that data centered models have not been exploited for their potential.

In a WS composition several services are combined to obtain, from their interaction, a more complex functionality. By considering in explicit way a model of how data are expected to be exchanged between the combined services, we can then check whether the implemented WS composition (which could also be dynamically bound) complies with that model, or whether desired properties are respected.

In this paper we have started with a first classification of possible data flow-based V&V scenarios. The objective is to lay down and share possible future interesting research trends in analysis and testing of Service Oriented Architecture.

We are currently setting a research agenda for influencing and validating some of the mentioned perspectives on the ART DECO project.

5. REFERENCES


