Issues in the formalization of Web Service Orchestrations[1]

Javier Cámara[1], Carlos Canal[2], Javier Cubo[1]

[1] CITIC, Andalusian ICT Centre, Málaga (Spain)
{jcamara, jcubo}@citic.es
[2] Dept. of Computer Science, University of Málaga (Spain)
canal@lcc.uma.es

Abstract. In this paper we outline an approach to the formalization of Web Service composition using WSBPEL (formerly BPEL4WS). Complementing current specification by adding protocol information, and making use of process algebra to model both Web Service and business process dynamic behaviour, we highlight key issues regarding the analysis of Web Service compatibility in the context of a business process managing their interaction.

1 Introduction

Recently, Web Services are increasingly being used by organizations and enterprises in a wide range of applications such as travel planners or financial services. The integration of these services facilitates cooperation across organizational boundaries, giving way to a new scenario of collaborations and opportunities in many fields. Unfortunately, the technology to model, search, compose, and access them is still far from mature, and the development of integrated Web Services is still a time consuming and expensive task. The need to achieve a higher degree of automation in the composition of Web Services has generated an important research effort by the Web Services community in order to address this issue. The software industry is also devoting more and more resources to solve interoperability problems, through organizations like W3C or WS-I [16]. These organizations promote the development and deployment of applications and services able to interact among them in a simple and efficient way through the Internet, independently of their platforms or languages.

Currently, description languages such as WSDL are capturing only static information about the signature and direction of the operations supported by a given Web Service. In order to raise the level of expressiveness of Web Service descriptions, two major conceptual approaches have been taken to Web Service composition [13]: the Standard based syntactic approach and the Ontology based semantic approach.

The standard based syntactic approach basically describes the order in which messages are exchanged among services. The two major flow description languages are: the Web Services Business Process Execution Language or WSBPEL [4]

[1] This work has been partly funded by the IST-2-004349-NOE AOSD-Europe network.

(formerly known as BPEL4WS), largely supported by the industry and that has currently become the *de facto* standard, and the Web Service Choreography Description Language or WS-CDL [15], the latest proposal of the W3C after WSCI.

On the other hand, the ontology based semantic approach developed by the semantic Web community, uses pre-agreed ontologies which explicitly define resources, preconditions, and effects of processes. OWL-S (formerly DAML-S) is a Web Service ontology based on OWL which provides the ability of describing properties and capabilities of Web Services unambiguously in a machine-interpretable way [11]. This opens new possibilities, allowing the use of reasoning techniques traditionally used in AI oriented towards automatic Web Service composition. Although a very promising and powerful approach, the semantic Web community is basing most of its research on techniques grounded on knowledge representation, goal-oriented planning, and logic [10]. These approximations, although with a great potential, still need further development and are in the way of solving several important issues, such as the representation of parallelism between processes in a natural way, a key aspect to software systems, and in particular, to Web Service composition.

Instead, we are focused in WSBPEL, since it is becoming the industrial standard, and we believe that it is medium term realistic approach towards automation. Adding protocol information to interacting services, and using additional adaptation techniques based in formalization through a process algebra, will establish solid foundations for seamless integration taking advantage of previous research made in this field [14]. This idea will be detailed in the next sections.

2 Formalizing WSBPEL

In this section we give a brief introduction to WSBPEL and justify the use of a process algebra to formalize it. In particular, we will use CCS [8], for which we will briefly describe how to use it for representing the behaviour of Web Services. This will enable us to reason about characteristics such as compatibility and replaceability, described in the following section, as well as to generate adaptors for mismatching behaviour whenever required and possible, just as it was done in [2] with WSCI, where a formal methodology for the automatic adaptation of software components described in [1] is applied.

WSBPEL is an XML based specification language used to describe business processes which manage (orchestrate) the interaction of different Web Services. A WSBPEL specification has four different parts:

Partner links: they identify relationships between the business process and the rest of the partners (Web Services). They basically provide WSDL port type definitions for process/web-service interactions.

Variables: they can carry data in messages, and define the state of each instance of the process. These may contain partner links, that is, abstract references to other processes. Thereby, they may be used to dynamically connect structures to each other.

Correlation sets: they identify interactions relevant for a given process instance, being used to dispatch messages correctly among different sessions.
Activities: describe the behaviour of the business process. They can be either basic or structured (see Table 1).

Table 1. WSPBEL relevant primitive and structured activities.

Primitive activities	
Receive	Accepts a message through the invocation of a specified operation by a partner
Reply	Sends a message as a response to a request previously accepted through a receive activity
Throw	Used when the process needs to signal a fault explicitly
Invoke	Used for invocation of a web service operation offered by a partner
Link	Defines a link of a flow; an activity within the flow can act as the *source* of a link or the *target* of a link
Structured activities	
Flow	Provides concurrency and synchronization (concurrent composition)
While	Supports repeated execution of a specified interative activity; execution continues until the specified boolean condition no longer holds true
Sequence	Includes one or more activities to be executed sequentially, in the order in which the appear under this activity
Pick	Waits the appearance of one or more events and executes the activity associated with the event that emerged. Messages incoming or timer pass form the posible events
Switch	Supports conditional behavior by enabling specification of one or more case branches whose execution depends on a specified condition, and an optional else branch which gets executed if all cases fail their checks

The nature and features of WSBPEL suggest the use of a process algebra to formalize it. Although the π-calculus, for instance, would be a good candidate, in an initial approach we will not model mobility (references to other processes expressed in variables containing partner links). This will make the expressiveness level provided by CCS enough to verify compatibility, reducing the level of complexity if we compare it with the option of the π-calculus, which will probably be used in a second deeper approach to the problem.

3 Reasoning about Web Service behaviour

Web Services pursue the achievement of the highest possible level of interoperability between software systems. Ideally, the composition of these services should be something like connecting pieces of a jigsaw game knowing beforehand that they are going to fit perfectly in every possible aspect with the rest of them. Unfortunately this is still far from the current situation. However there are a number of aspects that we can try to analyze in order to move in that direction when we have a given set of Web Services. The two key aspects to interoperability are compatibility and replaceability.

3.1 Compatibility and replaceability

For our purposes, we will consider that a software system, formed by the composition of several entities specified in a process algebra, is compatible when it terminates without requiring any interaction with its environment. However, this definition must be extended as detailed in [2] since client/server systems do not terminate, and we must consider infinite sequences of silent actions.

A formal notion of behavioural compatibility has been developed throughout several works such as [3] for CORBA components. These notions can be directly applied to Web Services. In [5] a model-based approach is proposed for verifying Web Services composition, using Message Sequence Charts (MSCs) and WSBPEL.

Replaceability refers to the ability of a software system to substitute another, in such a way that the change is transparent to external clients. In stateless Web Services, replaceability is fairly easy to check. We only have to test that the WSDL description of the new service contains all the operations of the replaced service. However, the situation is different at the behavioural level. First, we need to check that the dependencies of the new service when implementing the methods of the old one, are a subset of the dependencies of the old service. Second, we have to check that the relative order of incoming and outgoing messages of the old service is preserved by the messages of the new one..

Being able to express with WSBPEL dynamic behaviour has been an important step ahead in Web Service description, but once we have that information, what can we do with it? Which kind of properties can we infer from WSBPEL descriptions? How can we prove those properties? In the case that two Web Services are not compatible, can we solve this situation adapting them somehow?

3.2 Issues on behavioural analysis

There are two different approaches to the standards based syntactic composition [12]. In Web Service *choreographies* we have a description of the observable behaviour of each of the services participating in the interaction. For instance, WSCI defines interfaces for each of the interacting services. The alternative to choreographies is called *orchestration*, where a single business process coordinating the interaction among the different services is defined, as in WSBPEL specifications (see Fig. 1).

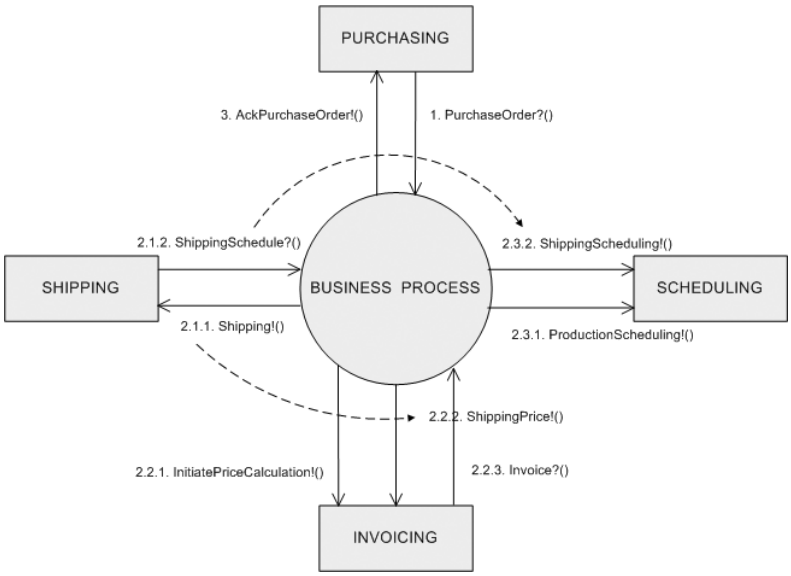

Fig. 1. Basic WSBPEL business process example: On receiving the purchase order from a customer (1), the process initiates three tasks: selecting a shipper (2.1), calculating the final price for the order (2.2), and scheduling the production and shipment for the order (2.3). While some of the processing can proceed concurrently (section 2), there are dependencies between tasks (represented by dashed lines). In particular, the shipping price is required to finalize the price calculation, and the shipping date is required for the complete fulfilment schedule. When the three tasks are completed, invoice processing can proceed, and the invoice is sent to the customer (3).

While the choreography approach provides a description of the dynamic behaviour of each of the services participating in the conversation through their interfaces, there is an inherent problem to orchestration, and it is that we have only a dynamic model of the process coordinating the interaction among services. WSBPEL provides the description of a business process which interacts with several Web Services through WSDL ports, each containing only static information about the signature of supported operations. The immediate consequence is that we have to assume either that we are interacting with stateless services, or that the Web Services the process is interacting with are going to behave as expected. This puts all the responsibility in the hands of the engineer, who has to perform the task of composition manually, considering all the possible issues in the interaction without any guide.

This lack of information about partner dynamic behaviour hampers the analysis of characteristics such as compatibility and replaceability, as well as the use of automatic or semiautomatic adaptation mechanisms between services. In order to overcome this situation, we need to complement the description of the services participating in the conversation with protocol information. Although different alternatives may be considered here, we have to choose carefully the most appropriate

taking into account the differences between Web Services orchestration and choreography: While in choreography interaction occurs between any pair of services arbitrarily, in orchestration all interactions have a single Web Service and the coordination process as endpoints. For this reason, we can choose between two alternatives when analyzing service compatibility and replaceability (see Table 2). In both of them we would model dependencies between operations by means of synchronization messages.

Global: Analysis of the interaction of the process with all the services involved in the conversation. While this alternative makes more information available and allows deeper analysis, it makes the process much more complex as well.

Partitioned: Analysis over the different projections of the business process in terms of observable behaviour or message exchange, on each of the services which interact with it. As an advantage, the analysis is substantially simplified, although several issues have to be addressed in depth, such as the detection of deadlocks.

Table 2. Global and partitioned behavioural specifications expressed in CCS for the purchasing process example.

Global behaviour of the business process with the participating services.

```
purchasingProcess = PurchaseOrder?().
                    ((Shipping!().
                      synch211-222!().
                      ShippingSchedule?().
                      synch212-232!()) ||
                     (initiatePriceCalculation!().
                      synch211-222?().
                      ShippingPrice!().
                      Invoice?()) ||
                     (ProductionScheduling!().
                      Synch212-232?().
                      ShippingScheduling!())).
                    AckPurchaseOrder!()
```

Projections of the business process with the services that interact with it.

```
purchasingService = PurchaseOrder?().
                    AckPurchaseOrder!()

shippingService   = Shipping!().
                    synch211-222!().
                    ShippingSchedule?().
                    synch212-232!()

invoicingService  = initiatePriceCalculation!().
                    synch211-222?().
                    ShippingPrice!().
                    Invoice?()

schedulingService = ProductionScheduling!().
                    synch212-232?().
                    ShippingScheduling!()
```

3.3 Adaptation

In case of detecting a mismatching or incompatible behaviour between a Web Service and the business process, we could attempt to perform adaptation between the two of them. The formalization previously described would allow adaptation of the services at the behavioural level, rather than dealing with the more traditional issues of message naming unification.

In our opinion, the semantic approach would be a more appropriate option to deal with message and parameter naming. By establishing sets of domain-specific common ontologies, inference techniques could be used in order to provide meaningful message exchange between different Web Services.

On the other hand, dynamic behaviour adaptation has to be performed as well, where message order has to be carefully considered in order to avoid deadlocks. Every possible execution path has to be considered. Generating an adaptor between the business process and a Web Service would require a mapping relating actions and data from the two software entities. In a first approach, this mapping would have to be confectioned manually, although semantic technology could substitute domain specific knowledge in order to provide a higher degree of automation for this task.

In conclusion, our opinion is that adaptation is not a trivial issue, and we have to use every available tool in order to solve interoperability problems. A hybrid approach would provide enough expressive power, both at the semantic and the behavioural level, to describe and adapt business processes and Web Services.

4 Conclusions and open issues

We have seen throughout this paper that there are mainly two conceptual approaches to Web Service composition [13]. On one hand, we have the ontology based semantic approach, which although very promising [6,7], is still in an early stage to offer short term results in the context of service composition. In [9] and [10] several models for checking the composition of Web Services are proposed. On the other hand, we have the standards based syntactic approach, which uses flow description languages and is currently supported by the industry.

We have attempted to describe a potential approach to the formalization of Web Service orchestration, with a specific interest in WSBPEL, the current industry standard, using a process algebra (CCS).

Several considerations have been made, regarding behavioural analysis of Web Services in the context of a process managing their interaction (orchestration). We proposed two different alternatives to the analysis of compatibility and replaceability of Web Services: global and partitioned, where analysis is performed on projections. Each alternative has to be studied carefully, and its benefits for the analytic process considered as well. For example, finding suitable mechanisms for deadlock detection, or appropriate formalization of WSBPEL link semantics in process algebra, are two important tasks which still have to be tackled.

Another major problem Web Service composition faces is the volatility of standards, in continuous evolution, appearing and becoming obsolete remarkably

soon. Flow description formalization through a process algebra such as CCS in this case, would allow a higher degree of independence between analytic machinery and description language syntax. This would enable the community to reuse analysis techniques as well as to adapt them to new standard specifications whenever required. Even in some cases where similarity between language constructs is remarkable, cross language analysis could be considered based on a common algebraic notation.

References

1. Bracciali, A., Brogi, A., Canal, C.: A formal approach to component adaptation. The Journal of Systems and Software. Special Issue on Automated Component-Based Software Engineering 74 (2005), pp. 45-54.
2. Brogi, A., Canal, C., Pimentel, E., Vallecillo, A.: Formalizing Web Services choreographies. ENTCS 105 (WS-FM'2004), pp. 73-94, Elsevier 2004.
3. Canal, C., Fuentes, L., Pimentel, E., Troya, J.M., Vallecillo, A.: Adding roles to CORBA objects. IEEE Transactions on Software Engineering 29 (2003), pp. 242–260.
4. Curbera, F., et al.: "Updated: Business Process Execution Language for Web Services version 1.1," BEA Systems, IBM, Microsoft, SAP AG and Siebel Systems, May 2003, available at http://www-128.ibm.com/developerworks/library/specification/ws-bpel/.
5. Foster, H., Uchitel, S., Kramer, J., Magee, J.: Model-based verification of web service compositions, in: Proc. of Automated Software Engineering (ASE'2003), 2003.
6. Mandell, D.J., McIlraith, S.A.: Adapting BPEL4WS for the semantic web: The botton-up approach to Web Services interoperation. To appear in the Proceedings of the Second International Semantic Web Conference (ISWC'2003), Sanibel Island, Florida, 2003.
7. McIlraith, S.A., Martin, D.L.: Bringing semantics to Web Services. IEEE Intelligent Systems, 18(1):90-93, January/February, 2003.
8. Milner, R.: "Communication and Concurrency". Prentice Hall, 1989.
9. Nakajima, S.: Model-cheking verification for reliable web service, in: Proc. of OOPSLA'02 Workshop on Object-Oriented Web Services, Seatle (USA), 2002.
10. Narayanan, S., McIlraith, S.A.: Simulation, verification and automated composition of Web Services, in: Proc. of the Eleventh International World Wide Web of Conference (WWW'2002), pp. 77–88.
11. OWL-S Home Page, "OWL-S: Semantic Markup for Web Services", The OWL Services Coalition (2004), available at http://www.daml.org/services.
12. Peltz, C.: Web Services orchestration and choreography, A look at WSCI and BPEL4WS. Web Services – HP Dev Resource Central, July 2003, available at http://devresource.hp.com/drc/technical_articles/wsOrchestration.pdf.
13. Talib, M.A., Zongkai, Y., Ilyas, Q.M.: Modeling the flow in dynamic Web Services composition. Information Technology Journal 3 (2) 184-187, 2004.
14. Viroli, M.: Towards a Formal Foundation to Orchestration Languages. Electronic Notes in Theorical Computer Science 105 (WS-FM'2004), pp. 51-71, Elsevier 2004.
15. W3C, "Web Service Choreography Description Language (WS-CDL) 1.0", World Wide Web Consortium (2004), available at http://www.w3.org/TR/ws-cdl-10/.
16. WS-I Organization, "Interoperability: Ensuring the Success of Web Services", Web Services Interoperability Organization (2004), http://www.ws-i.org/docs/20041130.introduction.ppt.