Design of BPEL Compositions based on Data Properties of Service Interfaces

George Feuerlicht
Faculty of Information Technology,
University of Technology, Sydney,
PO Box 123 Broadway Sydney NSW 2007 Australia
jiri@it.uts.edu.au

Abstract
The problem of composition of Web Services into business processes that represent business-to-business interactions has recently received extensive attention by the research community. Service composition methods range from industry standard approaches based on Web Services and BPEL to Semantic Web approaches that rely on AI techniques to automate service discovery and composition. BPEL is gaining wide industry support as a standard for specifying the implementation of inter-enterprise e-business processes. In this paper we view service composition as a design-time concern and propose a unified method for the design of services and service compositions focusing entirely on data properties of inbound and outbound messages that implement data flows between services. We use a flight booking scenario example to illustrate how a service composition model can be translated into a corresponding BPEL implementation.

Keywords: Web Services, BPEL, service composition

1. Introduction
The problem of composition of Web Services into business processes that represent business-to-business interactions has recently received extensive attention by the research community (Srivastava, 2003; Juric, 2006). Service composition methods range from industry standard approaches based on Web Services and BPEL (Business Process Execution Language) that focus on defining the workflow of Web Services execution, to Semantic Web approaches that employ AI techniques to automate service discovery and composition. BPEL (WS-BPEL) (OASIS, 2006) is gaining wide industry support as a standard language for specifying the implementation of inter-enterprise e-business processes by controlling the flow of execution of Web Services (Leymann 2001; Kloppmann, et al. 2004). BPEL applications typically use previously defined Web Services made available by business partners via partner links and coordinate business process execution passing messages between services. When developing BPEL applications the underlying assumption is that composition involves already existing Web Services and the design focus is on specifying the execution workflows using Web Services orchestration and choreography techniques. However, an equally important aspect of BPEL compositions is the mapping of the results of service invocations between outbound and inbound messages of individual Web Service operations as the execution progresses through the BPEL process. Consequently, the design of inbound and outbound message structures is of paramount importance as it determines the compatibility of service interfaces, and the composability of services into higher level business processes.

In previous publications we have described a methodological framework for the design of services that uses top-down decomposition based on the data properties of interface parameters to maximize cohesion and minimize coupling of service operations (Feuerlicht and Meesathit,
2004; Feuerlicht, 2005a; Feuerlicht, 2005b). In this paper we focus on the problem of service composition and extend the original design framework to include considerations of BPEL compositions based on data properties of service interfaces. Unlike the proponents of the Semantic Web Services approach (http://www.daml.org/services/) we regard service composition in the context of e-business applications as a design-time activity with the objective to ensure that service interfaces are mutually compatible (and composable), and at the same time exhibit a high degree of independence. In the following section (sections 2) we briefly describe our framework for service design and composition illustrating on an example how high-level business functions can be decomposed into elementary service operations with normalized interfaces and then mapped onto BPEL processes. In section 3 we show how the resulting process description can be implemented with BPEL. In section 4 we briefly review related literature, discuss the benefits of the proposed approach and identify potential for further work.

2. Service Interface Design
In this section we describe our approach to the design of BPEL compositions and illustrate the method using a simplified Flight Booking example. The method presented here is an extension of the service design methodology described in previous publications (Feuerlicht, 2005a; Feuerlicht, 2005b). The design method consists of three main design stages: the first stage (section 2.1) involves top-down decomposition with the objective of identifying elementary, reusable service components (i.e. service operations). The second stage (section 2.2) involves service aggregation with the aim of adjusting service granularity with respect to the requirements of a particular message interchange scenario (e.g. airline travel booking dialogue). This stage could also include consideration of performance, state management and other related issues; such issues are not considered in this paper. The final design stage (section 2.3) involves mapping the resulting service operations to BPEL processes that implement a specific business function. We illustrate the design method in the following sections using an example of the Flight Booking business process based on the OTA flight availability request/response messages: OTA_Air_AvailRQ/OTA_Air_AvailRS. For the purpose of this discussion we limit the scope of the example to the flight availability enquiry component of the Flight Booking business process, make a number of simplifying assumptions and use a subset of the OTA message data elements to populate the service interfaces.

2.1 Service Decomposition
During this design stage complex business functions are progressively decomposed into elementary functions and then mapped to corresponding candidate service operations. This approach is consistent with maximizing cohesion as elementary business functions typically accomplish a single conceptual task and exhibit high levels of cohesion. Decomposition of the Flight Booking business function can be achieved by modeling the interaction between a travel agent and an airline using a Sequence Diagram. Each step in the Sequence Diagram dialog produces a Request/Response message pair that corresponds to an elementary business function. Alternatively, elementary business functions can be identified as leaf functions in a business function hierarchy. Given the initial set of candidate service operations, further decomposition can be achieved by applying data normalization to the interface data parameters. Normalization of service interfaces eliminates redundant input and output parameters and improves the compatibility of service interfaces. This has a corresponding effect on service compositability as
composite service operation can be constructed based on common interface parameters (Feuerlicht, 2005a; Feuerlicht, 2005b).

The starting point of our analysis is a set of normalized service interfaces. Assuming functional dependencies FD1-FD5 between data parameters of the relevant service interfaces the corresponding set of service operations with normalized interfaces is shown in Table 1. The interfaces in Table 1 conform to the BCNF (Boyce-Codd Normal Form) criteria as the determinants (i.e. left-hand sides of the functional dependencies) form the data parameters of inbound messages (i.e keys of the interface relations) (Date 1992).

FD1: OriginLocation, DestinationLocation, DepartureDate → FlightNumber
FD2: FlightNumber → DepartureAirport, DepartureTime, ArrivalAirport, ArrivalTime
FD3: FlightNumber, DepartureDate → ArrivalDate
FD4: FlightNumber, DepartureDate, CabinType → Quantity
FD5: FlightNumber, DepartureDate, CabinType → BasicFare, BasicFareCode

<table>
<thead>
<tr>
<th>Business Function</th>
<th>Operation</th>
<th>Inbound Message</th>
<th>Outbound Message</th>
</tr>
</thead>
<tbody>
<tr>
<td>Requests for available flights for a pair of origin and destination cities on a given departure date.</td>
<td>FlightEnquiry</td>
<td>OriginLocation, DestinationLocation, DepartureDate</td>
<td>FlightNumber</td>
</tr>
<tr>
<td>Request for flight schedule information for a given flight number.</td>
<td>ScheduleEnquiry</td>
<td>FlightNumber</td>
<td>DepartureAirport DepartureTime, ArrivalAirport, ArrivalTime</td>
</tr>
<tr>
<td>Request for arrival information for a given flight.</td>
<td>ArrivalEnquiry</td>
<td>FlightNumber, DepartureDate</td>
<td>ArrivalDate</td>
</tr>
<tr>
<td>Request for seat availability information for a given flight and cabin type.</td>
<td>SeatEnquiry</td>
<td>FlightNumber, DepartureDate, CabinType</td>
<td>Quantity</td>
</tr>
<tr>
<td>Request for pricing information for a given flight and cabin type.</td>
<td>PriceEnquiry</td>
<td>FlightNumber, DepartureDate, CabinType</td>
<td>FareBasicCode, BasicFare</td>
</tr>
</tbody>
</table>

Table 1. Normalized interfaces for the Flight Booking Service.

2.2. Service Aggregation

The flight availability business process could be implemented directly using the above fully normalized service interfaces as a dialogue between a travel agent and an airline. However this fine-granularity solution results in an excessively complex dialogue characterized by a large number of runtime calls and does not represent a practical solution. Let us now consider aggregating the elementary operations shown in Table 1 based on interface parameters. Inherent property of normalized service interfaces is the correspondence between interface parameters (i.e. parameters of inbound and outbound messages) of individual operations. For example, the ScheduleEnquiry and ArrivalEnquiry interfaces share a common parameter FlightNumber and can be combined into a new operation TimeTable based on the FlightNumber parameter (Feuerlicht 2005a):

**TimeTable**

IN: FlightNumber, DepartureDate, 
OUT:DepartureAirport, DepartureTime, ArrivalAirport, ArrivalDate, ArrivalTime)
Similarly, we can consider combining operations SeatEnquiry and PriceEnquiry using the common attributes (FlightNumber, DepartureDate, Cabin-Type) producing a new operation SeatPriceEnquiry:

\[
\text{SeatPriceEnquiry(in: FlightNumber, DepartureDate, CabinType, out: Quantity, FareBasisCode, BaseFare)}
\]

The above aggregations produce operations that are no longer strictly atomic. However, the loss of functional cohesion can be justified on the basis that the operations closely correspond to the conversational requirements of the flight availability business process, and that the benefits of reduced number of runtime procedure calls (from 5 to 3) outweighs the relative loss of cohesion.

### 2.3 Mapping Services to BPEL Processes

The problem of service composition for BPEL is mostly treated as a problem of coordinating invocations of existing services using workflow features of BPEL to compose sequential and parallel process flows that consist of individual BPEL activities (i.e. steps). Our approach is to focus on the data flows between services and to study the problem of service composition from the perspective of service aggregation based on interface data parameters. In this context service composition takes place by passing messages between individual Web Service operations, rather than directly by aggregating service interfaces, however, the underlying principles used for matching interface parameters are the same. It is not our intention to fully model the BPEL process workflows and we simplify the BPEL process dialogue by excluding alternative execution paths, for example the termination of the booking process due to unavailability of seats on a particular flight. We now map the resulting services FlightEnquiry, TimeTable, and SeatPriceEnquiry to BPEL process steps, in effect continuing the process of service aggregation started in section 2.2, but this time composition involves passing data parameters between individual process steps using messages and would typically involve local and global BPEL variables to store and manipulate intermediate results of service invocations. The resulting Flight Availability Enquiry business process is illustrated in Figure 1 and proceeds as follows:

The travel agent specifies input values for OriginLocation, DestinationLocation, and DepartureDate parameters (step 1). Following the execution of the FlightEnquiry operation (step 2), the travel agent selects a suitable flight (i.e. FlightNumber) (step 3). The travel agent then executes the operation TimeTable (step 4) and supplies the value for CabinType, e.g. Economy (step 5). Finally, the travel agent executes the operations SeatPriceEnquiry (step 6) to obtain the availability and price information for the selected flight.

### 3. BPEL Process Implementation

The diagrammatic model of the Flight Availability Enquiry business process shown in Figure 1 forms the basis for BPEL implementation. The service interfaces for operations FlightEnquiry, TimeTable, and SeatPriceEnquiry can be mapped directly to a WSDL specification (Feuerlicht, 2003), and the corresponding business process can then be implemented in BPEL as interaction between the client application (e.g. executed on behalf of a travel agent) and services executing at the FlightService Partner Link (e.g. Web Service provided by an airline).
Figures 2 below show the BPEL process and the corresponding Request (inbound) and Response (outbound) messages for the FlightEnquiry operation. At the start of the process, BPEL engine receives values for data parameters (OriginLocation, DestinationLocation, DepartureDate) from the client application, invokes the FlightEnquiry operation using these values as the inbound message, and returns the values of (FlightNumber) to the client application.
Next, the client selects a FlightNumber and invokes the TimeTable operation with the values (FlightNumber, DepartureDate) forming the inbound message, returning the parameters (ArrivalDate, ArrivalTime, DepartureAirport, DepartureTime, ArrivalAirport) as the outbound message. Finally, the client supplies a value for CabinType and executes the SeatPriceEnquiry operation with the values (CabinType, FlightNumber, DepartureDate) forming the inbound message, returning the values (Quantity, FareBasisCode, BaseFare) as the outbound message to the client application, and terminating the process. We do not show the BPEL implementation for TimeTable and SeatPriceEnquiry operations here because of the limited scope of this paper.

4. Conclusions and Future Work

Service composition is an active research area with many diverse approaches being currently investigated. Industry based research views Web Services as abstract standardized interfaces to business processes and focuses on describing and implementing service composition using workflow specification languages such as BPEL. The Semantic Web research community takes a different approach and draws on AI planning research and run-time reasoning techniques based on ontological definitions of service semantics. A comprehensive review and comparison of the two approaches to service composition is provided in (Srivastava 2003; Milanovic 2004) with the conclusion that the Web Services compositability problem remains essentially unsolved. In addition to BPEL, other standardization efforts include Web Services Choreography Interface (WSCI) and the Business Process Management Language (BPML) each taking a different approach to orchestration and choreography (Peltz, 2003). However, BPEL is today established as the industry standard language environment for the implementation of business-to-business Web Services applications with WSDL extensions that support the implementation of loosely coupled asynchronous applications that share common XML data types and documents (Pasley, 2005).

The aim of service composition is to assemble higher-level services from component services in order to implement a desired business function. The approach described in this paper complements existing literature on the topic of design and composition of service-oriented applications by viewing service composition as a design-time concern and proposing a methodological framework for the design of BPEL compositions based on data properties of interface parameters. The main contribution of this paper is the proposed unified framework for service aggregation and composition based on combining operations using common interface parameters. We show that both service aggregation and composition can be viewed as design-time activities that combine service operations based on data properties of interface parameters. Further research is needed to understand how service aggregation based on interface parameters can be used to achieve optimal service granularity given a set of application requirements. Another area of research interest concerns the application of this methodology in the more general context of services composition.

I wish to acknowledge the assistance of Mr Sia Minh Hong with implementing the BPEL Flight Enquiry example application.
5. References


5. Feuerlicht, G., Application of Data Engineering Techniques to Design of Messages Structures for Web Services, Proceedings of the First International Workshop on Design of Service-Oriented Applications (WDSOA'05), Amsterdam, The Netherlands, December 12, 2005, IBM Research Report, RC23819 (W0512-29), December 6, 2005b


