Definition, Execution, Analysis, and Optimization of Composite E-Services

Fabio Casati and Ming-Chien Shan
Hewlett-Packard Laboratories
1501 Page Mill Road, 1U-4
Palo Alto, CA, 94304
casati,shan@hpl.hp.com

Abstract

The Internet is becoming the mean through which services are delivered to businesses and customers. Standard bodies as well as individual software vendors are rushing to define languages, protocols, and tools that support the development and deployment of e-services. The shift to the e-services paradigm provides the opportunity and the need for defining value-added, composite *services by orchestrating other basic or composite ones. In this paper we present our efforts towards the development of a service composition facility. We first illustrate the characteristics of composite services, and then describe the service composition model and language we have developed, by reusing concepts from the workflow research community where possible and extending them when needed. We next introduce the prototype implementation of a composite service engine, and we conclude the paper by detailing our work in progress on composite service analysis and optimization.*

1 Introduction

The next chapter of the Internet story is about the mass proliferation of *e-services*. Indeed, more and more companies are rushing to provide all sorts of services on the Web, ranging from "traditional" on-line travel reservations and directory services to real-time traffic reports and even outsourcing of entire business functions of an organization. As Web technologies continue to improve, allowing for smaller and more powerful web servers, and as more and more appliances become web-enabled, the number and type of services that can be made available through the Internet is likely to increase at an even higher rate.

A few years ago, HP decided to invest heavily in the e-service area. Efforts were focused on both the development of specific e-services as well as *infrastructures* for supporting the design, development, and delivery of e-services. In particular, one of the main results of this effort has been the development of an e-services platform called *e-speak* [6]. E-Services Platforms (ESPs) are software tools that facilitate users in deploying applications (typically Java objects) as e-services and in securely delivering the services offered by these applications to authorized users.

Bulletin of the IEEE Computer Society Technical Committee on Data Engineering

ESPs allow service provider to *register* service descriptions and to *advertise* them in web directories (such as LDAP or UDDI directories). Service providers are also offered features to *monitor* and *manage* service executions. Once a service has been registered and possibly advertised, it can be discovered by customers. Typically, customers query the service repository associated with the ESP or web directories for services of interest. As the result of the query, clients get references (handles) to e-services that can fulfill their needs. Then, clients can use these references to get more information about the services or the service providers, or they can immediately invoke a service. Access to a service is restricted to authorized users, according to the security features provided by the platform.

The e-service economics create the business opportunity for providing value-added, integrated services, delivered by composing existing e-services, possibly offered by different providers. For instance, an *eMove* composite service could support customers that need to relocate, by composing truck rental, furniture shipments, address change notification, and airline reservation services, according to the customer's requirements [1].

Service *composition* (or *orchestration*) is seen by many ESP vendors, (including HP) as a strategic area that can provide a competitive advantage within the ESP arena. In this paper, we provide an overview of our work on service composition, both with e-speak and with ESPs in general, and then we give pointers to our technical reports for details. We begin by describing our work on the service composition model and present the prototype we have developed. Then, we introduce our work in progress on composite service analysis and optimization.

2 Service composition model and language

A composite service is similar to a workflow. In fact, it represents a complex activity specified by composing other basic or complex activities. Like in workflows, a composite service definition must include the specification of the component services and their execution dependencies, as well as their input and output parameters. However, composite services have several characteristics that makes them different from workflows (see [3] for a detailed discussion):

- Services in ESPs are typically more complex than workflow activities (that are often modeled as "single-function" black boxes). In particular, in most e-services model (including e-speak) a service can offer several operations to be invoked. For instance, a car rental service may offer operations to *browse* available cars, and then *book* a car or *cancel* a reservation. Interacting with an e-service requires operations to be performed at the service level (e.g., search and authentication) and operations to be performed at the method level (e.g., method invocations). These aspects of e-services must be taken into account by a service composition model. In addition, services typically interact by exchanging XML documents, while XML support is very limited in "traditional" workflow models and systems.

- Composite services need to be more adaptive and dynamic, consistently with the adaptive and dynamic characteristics of the e-services environment, where new services are created every day.

- Security is a key issue, since services do not reside within the protected intranet, but can be anywhere on the Internet. Hence, mechanisms to specify and enforce security requirements must be supported.

In order to meet the above-mentioned requirements, we have designed a Service Composition Model (SCM), by reusing workflow concepts where possible and extending them when needed. We next present SCM and the corresponding language, called *Composite Service Description Language* (CSDL). Details on the model and language are provided in [2, 3]. Like most workflow models (see, e.g., [5]), CSDL describes the composition by means of a directed graph, whose nodes represent interactions with e-services (for instance to authenticate or invoke service methods) and arcs define execution dependencies among services. Special nodes are provided to denote the starting and ending points of the composite service, or to route the execution flow. Each composite service has a set of local data items, that can be used to exchange data with the invoked services and to make routing decisions.

SCM has a two-level business process model, that distinguishes between invocation of *services* and of *methods* within a service, represented by service and method nodes, respectively. Service nodes define service-level parameters, such as the service selection criteria (e.g., a car rental service available in Como, Italy) and the digital certificate to be sent to the service upon invocation, while method nodes model the invocation of a specific operation within a service. In addition, since clients may need to invoke several operations on the same service, SCM allows the designer to represent, within a service node, the flow of method invocations to be performed on a service. For instance, in order to access a car rental service, we may specify that we first *browse* and get details about available cars and their prices, and then *book* a car if the price is below a certain limit.

Method nodes define the methods to be invoked on a service, as well as the input and output data. The name of the method to be invoked can be statically specified, or it can be dynamically taken from the value of a composite service data item. To support the invocation of e-services that exchange information through XML documents (and in particular to model invocation of services defined according XML-based B2B standards, such as RosettaNet), designers can associate *XML templates* to method invocations. XML templates are XML documents in which some parts are parametric and refer to data items of the composite service. When a method is invoked, these parameters are replaced with actual values, and the resulting XML document is sent to the service. If the service output is also an XML document, then it is possible to extract the values of elements or attributes of interest and insert them into data items of the composite service. This is done by defining, for each data item to be updated by the method invocation, an XQL query that processes the output document and obtains a scalar value. Details are provided in [3].

In order to manage authentications, SCM provides a flexible and easy-to-use mechanism for handling certificates. In fact, the definition of a composite service includes information about the certificates to be used when invoking services within the flow, in case the ESP and the invoked e-service support, or even require, the use of digital certificates, as in e-speak. By default, services are invoked with the privileges (i.e., the certificate) of the composite service *user*. However, the designer may specify that services should be invoked with the privileges of the composite service *designer*, or with the privileges specified by the content of a data item (for instance, the certificate to be used may have been received as output parameter during a previous invocation). In addition, defaults at the composite service level can be overridden at the service node level.

To manage and even take advantage of the frequent changes in today's business environment, composite services need to be *adaptive*, i.e., capable of adjusting themselves to changes with minimal or no manual intervention. SCM includes several constructs that allow services to be adaptive: These are *dynamic conversation selection*, *multiservice nodes*, and *generic nodes* (see [4]).

Dynamic conversation selection makes it practically possible to use dynamically discovered e-services. In fact, one of the most advertised advantages of e-services is the ability of dynamically selecting the best available service. Facilities for dynamic selection are also provided in CSDL. However, since different providers may offer different interfaces for the same type of service, if the conversation (i.e., the flow of method invocations) is statically defined at composite service definition time, then the statically defined conversation could be inconsistent with the interface provided by the dynamically selected service. By using dynamic conversation selection, the designer is not required to secify the conversation at definition time, but can instead specify that the conversation should be selected at run time, based on the service returned by the execution of the service selection query. Conversation selection is performed by maintaining a repository of conversations. The designer specifies within a service node that the conversation should be dynamically selected from a repository. At service invocation time, the system will load from the repository the conversation that corresponds to the selected service. Note that if dynamic conversation selection is used, then the designer should make sure that the service selection query restricts the search to those services for which there is an entry in the conversation repository, otherwise an exception is raised.

Multiservice nodes model multiple, parallel activations of the same service. For instance, a composite service may need to invoke several car rental services to get prices and availability. The number of nodes to be activated is determined at run time, based either on the number of service providers able to offer a given service, or on the number of elements in a composite service data item of type *list*. For instance, in order to rent a car, we may want to check the prices of all car rental companies available at a given location. As another example, assume that a composite service must check the credit history of a group of customers. The number of service nodes to be instantiated must be equal to the number of customers, and each node will focus on one customer.

Generic nodes enable the parallel activations of multiple instances of *different* services. Unlike ordinary service nodes, which are either statically bound to a service or to a service selection query, generic nodes are placeholders. They only include a configuration parameter that can be set with a list of actual service nodes either at composite service instantiation time (through the composite service input parameters) or at runtime.

While adaptive models reduce the need for human intervention in maintaining composite service definitions, there are still cases in which these definitions need to be modified, or in which actions need to be taken on running instances to modify their course, for instance to handle unexpected situations or to incorporate new business policies. SCM allows two types of service process modifications: ad-hoc and bulk changes [3].

Ad-hoc changes are modifications applied to a single running service process instance. They are used to manage exceptional situations that are not expected to occur again. *Bulk changes* refer to modifications collectively applied to a subset (or to all) the running instances of a composite service. Modifications to running instances are specified by a set of *migration rules*, that define the flow to which each instance should migrate, i.e, specify the future behavior of running instances. A migration rule identifies a subset of the running instances of a given composite service as well as the target service definition. They have the form `IF <condition> THEN MIGRATE TO <flow_def>`. The condition is boolean expression that identifies a subset of the running instances, while `<flow_def>` denotes the destination flow. The rules must define a partition over the set of running instances. Instances that do not satisfy any rule condition are not migrated.

3 Prototype implementation

Service composition facilities can be offered in several different ways: as a development environment targeted to the enterprise IT personnel, as an e-service itself, or as a functionality integrated with the ESP. We decided to structure the composition facility as an e-service (or, rather, a meta-service, since it is a service for developing e-services), since in this way it can be advertised, discovered, delivered, managed, and protected by end-to-end security analogously to any other e-service, thereby exploiting all the advantages and features provided by the ESP. In particular, service composition functionality can be offered to other businesses and customers, thereby relieving them from the need of maintaining a composition system that may be onerous to buy, install, and operate. In addition, with this approach it is easier to provide the same functionality on top of ESPs from different vendors, by making (relatively) simple modifications to the prototype.

In the following we will refer to this service composition facility as *Composition E-Service*, or simply CES. The prototype is built on top of e-speak and HP Process Manager (a commercial workflow engine). CES has a front-end that accepts invocations of CES methods from service providers, in order to register, modify, and delete composite services. For instance, providers can register a composite services by invoking the *register* method of the CES and sending the CSDL description of the composite service as parameter (see Figure 1). The CES then translates CSDL into the language of the selected workflow engine, inserting the appropriate "helper" nodes and data items that enable the correct implementation of the CSDL semantics.

A *gateway* enables the interaction between the workflow engine and the ESP. All the nodes defined in the generated workflow are assigned to the gateway, that invokes the appropriate operations on the ESP or on the selected service, thereby performing the appropriate data mappings and implementing CSDL semantics that could not be supported by the workflow engine.

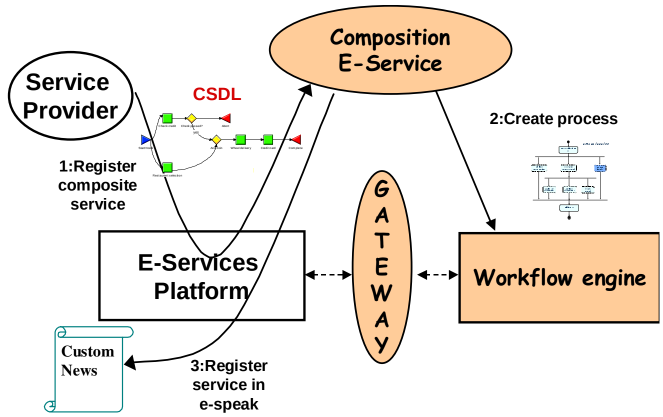

Figure 1: Composite service registration (components of the prototype are shaded).

Once the translation has been performed and the process deployed on the workflow engine, the CES makes the composite service available as an e-service, by registering it with the ESP. The CES also registers itself as being the handler of the service. From this moment, the (composite) service can be searched and invoked just like any other basic service.

When a client invokes a service on the ESP and the service is composite, the ESP calls the CES that, in turn, activates the corresponding process in the workflow system. The workflow engine will then run the process by scheduling nodes in the workflow and assigning them to the gateway. In particular, the gateway receives indication of what to do by the workflow engine as part of the node data items that provide (a) context information about the service on which method calls have to be made and (b) the value of the parameters to be passed as part of the method invocation. When the workflow completes, the CES returns the service results to the ESP, which transfers them to the invoking client.

4 Service composition analysis

This section describes our work in progress in the service composition area, focused on the analysis and optimization of composite service executions. In order to attract and retain customers as well as business partners, organizations need to provide (composite) services with a high, predictable quality. From a service composition perspective, this has several implications: for instance, the composite services should be correctly designed, their execution should be supported by a system that can meet the workload requirements, and the invoked services should be able to perform the work with the appropriate quality. To support providers in offering services that have a high and predictable quality, we are working towards the development of a comprehensive, customizable, and ready-to-go business process intelligence (BPI) solution. This solution enables business and IT users to extract knowledge from composite service execution logs and to be alerted of critical situations or foreseen quality of service degradations. In addition, the BPI tool suite is capable of dynamically affecting composite service execution (with the limitations and constraints allowed by the service designer). The basic idea behind our work, started in the workflow management domain and then extended to composite services, consists in analyzing service execution logs with OLAP and data mining technologies, in order to enable interactive analysis and to extract ”hidden” information.

The overall BPI architecture is shown in Figure 2. Data are periodically extracted from the CES logs and loaded into a warehouse by Extract, Transfer, and Load (ETL) scripts. Having a data warehouse of workflow data is in itself very useful, since it enables OLAP and multidimensional analysis of audit logs, possibly including

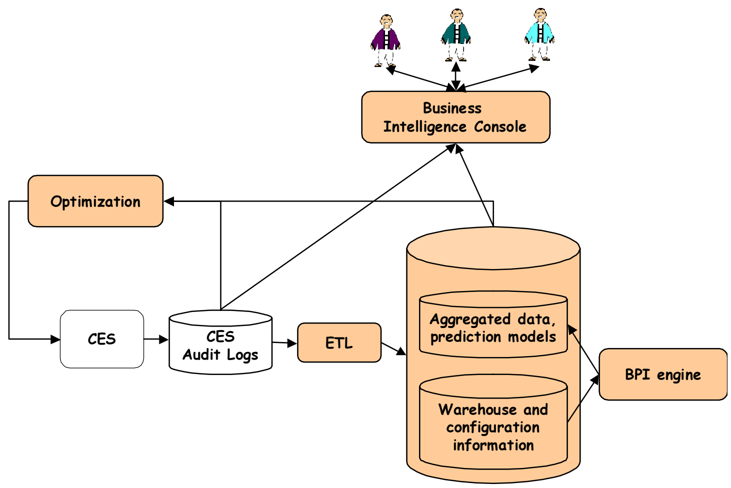

Figure 2: Overall architecture of the Business Process Intelligence application.

aggregation of data coming from different CESs. The BPI engine then processes the content of the warehouse and produces:

- Aggregated data and statistical information about the performance and quality of composite services as well as of individual, component services.
- Explanation of execution behaviors, such as the activation of given paths in the flow, the use of a specific service, the occurrence of an event, or the (in)ability to meet service level agreements.
- Prediction models, i.e., information that can be used to predict the behavior and performances of a composite service instance, of the invoked services, and of the CES.

The initial tests, conducted on internal administrative services for HP employees, have given promising results, identifying bottlenecks and causes of delays in service executions. We now plan to extend this work into a product to be coupled with the HP e-speak and HP Process Manager composition tools.

References

[1] F. Casati, S. Ilnicki, L. Jin, V. Krishnamoorthy, M. Shan. e-Flow: a platform for developing and managing composite e-services. In *Proceedings of the Aiworc'00*, Buffalo, NY, USA, June 2000.

[2] F. Casati, S. Ilnicki, L. Jin, M. Shan. eFlow: An open, flexible, and configurable system for service composition. In *Proceedings of Wecwis'00*, Milpitas, CA, USA, June 2000.

[3] F. Casati, S. Sayal, M. Shan. Developing e-services for composing e-services. 2001. Submitted for publication and available on request.

[4] F. Casati and M. Shan. Dynamic and adaptive composition of e-services. *Information Systems*. To appear.

[5] D. Georgakopoulos, M. Hornick, A. Sheth. An overview of workflow management: From process modeling to workflow automation infrastructure. *Distributed and Parallel Databases*, 3(2):119–153, April 1995.

[6] Hewlett-Packard. e-speak: Architectural Specifications. Available from www.e-speak.hp.org, June 2000.