IEIP: an Inter-Enterprise Integration Platform for e-Commerce Based on Web Service Mediation

Suo Cong
University of Zürich
Switzerland
congsuo@ifi.unizh.ch

Ela Hunt
ETH Zürich
Switzerland
elahunt@inf.ethz.ch

Klaus R. Dittrich
University of Zürich
Switzerland
dittrich@ifi.unizh.ch

Abstract

Cross-platform interoperability of web services makes it possible to wrap and combine information assets from heterogeneous systems to create a consolidated application. Here we propose a new pragmatic architectural model using web services as building blocks, to form an inter-enterprise integration platform for e-commerce. The key idea is to separate the view of a web service defined by its provider from the views of this web service employed by consumers. Based on this principle, we explain the use of web service mediators which act on behalf of both service providers and consumers. We advance the notion of requester-based views of a web service, which are published by a mediator and used by service providers to construct services that suit the consumer. To complement those, we use customer-oriented global views which express the information need of a customer. IEIP architecture improves on traditional data and enterprise service integration systems. It achieves the following two goals: a customer can use its own (private) view to access a number of different web services as it is using a global view; and a provider can provide one view for a service implementation that can be accessed by different requesters using different global views. Leveraging this model, we report on a prototype supporting a European-wide parking services platform that integrates data and services from a large number of parking providers.

1 Introduction

Electronic applications, such as e-commerce, e-government, and e-healthcare, are increasingly required to integrate a variety of autonomous and disparate information systems across organizational boundaries [24, 26, 28, 29, 33]. Such systems deliver new value-added services or improve the quality of existing services. It is noteworthy that industry is pursuing new integration approaches based on the convergence of EII and EAI [24]. We believe a web services-based approach is an optimal candidate platform able to deliver significant improvements in both data and application integration. Since the emergence of web services, the services themselves and the related technologies have been embraced quickly by most software vendors and have gained remarkable success. Consequently, web services are increasingly used by industry to share enterprise-level services for conducting business across the internet using XML-based lightweight protocols [10, 16, 21, 28, 33]. Furthermore, databases are also expected to be encapsulated as and accessed via web services [23]. We believe this trend is a significant step towards real world applications which go beyond traditional data integration systems which mainly focus on schema mapping/matching and data exchange but ignore the issue of how enterprise data can be actually retrieved across heterogeneous information systems.

We observe rapid growth in both numbers and types of web services. Within our projects either the industrial partners have already implemented some web services, or they are planning to do so in order to integrate their systems. Our contribution is a web services based inter-enterprise integration platform (IEIP). IEIP is a foundation on which service providers and requesters can easily cooperate across organizational and geographic boundaries. Our inspiration comes from the traditional data integration systems [27] that provide every client with one global view of and unified access interface to data residing at distributed data sources. Beyond the one-view-for-all approach, we introduce the concept of a customer-oriented global view which provides a private global view for each distinct client. Of course, in real applications, it is very likely that some clients can share one global view. By using private global views, both the service providers and requesters gain further flexibility to offer services and consume them.

The paper is organized as follows. In Section 2 we present a brief survey of enabling technologies and discuss

the standards that underpin our IEIP implementation. The architecture of the IEIP is presented in Section 3 and an implementation is described in Section 4. Finally, we discuss our contribution and conclude in Section 5.

2 Overview of IEIP

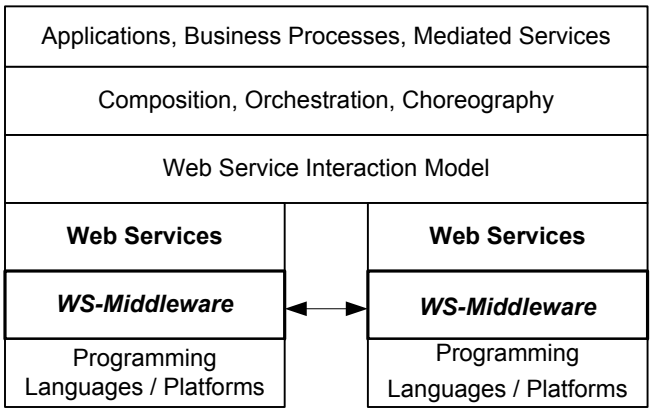

Figure 1. IEIP architecture uses WS-Middleware

The primary goal of IEIP (Inter-Enterprise Integration Platform) is clear and simple: to manage the increasing complexity of a large number of interfaces for both service providers and requesters. The targets to be managed by an IEIP are both enterprise data stored in backend databases or semi-structured storage, and business functionality implemented using different programming languages and running on different platforms. Traditionally, the first integration activity has been implemented by EII, based on data integration techniques and the second integration activity has been achieved by EAI, EDI, and middleware such as CORBA and EJB. Due to the intrinsic and significant heterogeneity of enterprise information systems, none of the above solutions can be successful in supporting inter-enterprise integration, although they are appropriate in intra-enterprise integration.

Our IEIP architecture is shown in Figure 1. Web services are the core of inter-enterprise integration, and WS-Middleware is the component which allows us to flexibly combine a number of web services via view composition and asynchronous execution of service requests. In the following subsections we survey the enabling technologies supporting this model and focus on the widely used technologies that underlie the IEIP architecture.

2.1 Building block: web services

The primary strength of Web Services is the internet-wide interoperability across different programming languages and platforms. This is achieved by enforcing simple and flexible standards which allow us to define and discover services that support loosely-coupled and interoperable program-to-program interaction. The service-oriented interaction paradigm leads to a design strategy in which all software components and their interfaces could be exposed and used as a service [9, 12]. We classify the standards and proposals for web services into two levels: the core standards which define the minimum essential web service model and are widely used by current commercial applications in the real world; and the enhancing standards which are extensions to the core and are mainly used in research. The core of the basic web services model consists of two standards [12, 42, 4]: WSDL (Web Services Description Language [7]) and SOAP [3]. Currently both are supported by all major software vendors and widely used by industry. There are numerous enhancements to the core web services model, especially to introduce semantics into web service descriptions. Typical examples include standard specifications such as OWL-S [30]. However, the use of such enhancements could cause severe interoperability problems in e-commerce applications.

2.2 Web services middleware and wrappers

The term ”Web Services Middleware” is not used consistently. In some contexts, it refers to middleware implemented using web services. Here WS-Middleware refers to software tools (e.g. Apache Axis *ws.apache.org/axis*) mediating between the layer of business logic implemented in general-purpose programming languages (Java, C++/C#, PHP, Perl) and the layer of standard web services, as shown in Figure 1. This layer has often been invisible in research prototypes. However, it is not a trivial component that can be ignored by real industrial applications. Indeed, it is crucial to the success of internet-wide interoperability among heterogeneous systems because it determines the development strategy of web services.

Although a number of efficient XML processing tools are available, it is generally not a good idea, at this stage, to generate complex SOAP messages (e.g. assemble a SOAP request) and the business documents carried by the message (e.g. parse an XML document to get the input parameters) directly in the layer of business logic, because this strategy results in a bulk of custom code to handle the complexity of SOAP messages. Furthermore, such custom code can hardly be reused by different applications. As a result, most current commercial applications providing or consuming

web services have been implemented using general-purpose languages, and are using web services middleware to transform the web services into relevant programming language-specific constructs like distributed objects. Based on our experience in developing web service applications, we explicitly include web service middleware in the IEIP architecture not only because many applications nowadays are using it to provide and consume web services, but also because we consider it as a layer improving the reusability of components processing SOAP message or other types of web service messages in the future. In that way, we obtain "thinner" applications.

One closely related concept is the Web Service Mediator (e.g. the mediation service in the WSMF [20]) which is often defined variously in different contexts. In this paper we use the term "Web Service Mediator" to refer to a component of the IEIP platform, which is designated to transform a SOAP request message issued by a requester to a SOAP request message expected by a provider, and a SOAP response message created by a provider to a SOAP response message expected by a requester.

2.3 Interaction model: messaging

The interaction model of web services is based on messaging. XML-based messages carry business documents between services and requesters [12, 40, 41, 42, 4]. This approach achieves internet-wide interoperability and loose coupling. Its success is largely due to the simplicity of the open standards for web services, which all software vendors can easily support. SOAP specifies messages using XML in a structured manner and defines bindings to transport protocols such as HTTP or SMTP [3]. Web services can be invoked simply by sending XML documents encapsulated into SOAP messages. However, such internet-wide vendor-independent interoperability is still fragile to some extent in real world applications, because the processing of SOAP messages can be implemented differently by different vendors, especially when heterogeneous web services middleware is employed to transform SOAP messages into language-specific constructs.

2.4 Composition, orchestration, and choreography

In order to support inter-enterprise cooperation, more sophisticated interaction models are required to bind web services together. Application developers need flexible mechanisms to make use of a variety of web services which may operate in autonomous and heterogeneous environments. These mechanisms are often referred to as service composition, orchestration, and choreography [9, 12, 2, 20, 31, 32, 38, 5]. Web services composition refers to combining a number of web services into one processing entity at a higher level, to provide some new functionality. Web services orchestration and choreography refer to creating an executable business process by specifying the execution orders of a set of web services.

2.5 Mediated web services: reusable components

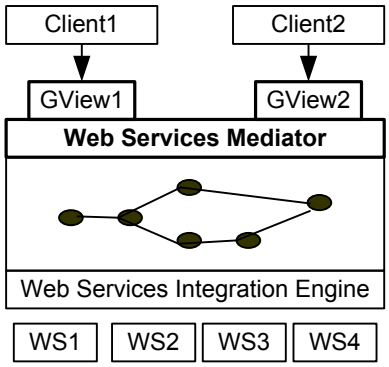

Figure 2. A web service composed by a WS-Mediator

Each web service has an interface defined by its provider. To invoke a web service, requesters need to discover the interface of this web service and understand its semantics. When the number of web services increases, the complexity of discovering and binding to each individual service interface increases significantly [41]. To overcome this problem, we delegate the management of web service usage complexity to a mediator. As shown in Figure 2, in IEIP each client has its own private global view which is dynamically created by composing and orchestrating the underlying enterprise web services into one processing entity. By using a web service mediator, client applications are released from handling the complexity of interacting with each individual web service. And the service providers can also be released from supporting multiple views used by different clients.

2.6 Dynamic binding

A very significant promise of web service technologies is to enable a service requester to determine the actual web service to be invoked at run time [9, 4]. This is referred to as dynamic binding in which a requester dynamically discovers, selects, and invokes a web service at run time. In the standard SOA (Service Oriented Architecture) paradigm, this is realized by employing service registries such as UDDI [1] or some form of discovery agencies [4]. Although, theoretically, dynamic binding is possible, it is

very difficult in practice to develop commercial applications without knowing precisely which particular web services or operations will be invoked [9].

One of the primary objectives of IEIP is to support dynamic binding to web services. Our WS-Mediator resolves the binding in a common layer which can be shared by other applications and shifts the matching and processing difficulty from each application to IEIP.

2.7 Management of web services

Both service providers and requesters need to manage web services. Providers use management functions to improve service provision and requesters use management functions to select the best service dynamically. Service management in the IEIP manages both web services themselves and their execution environment through a set of capabilities for monitoring, controlling, and reporting service qualities and service usage [15, 4]. IEIP defines a two-layer management scheme: enterprise layer and inter-enterprise layer. The enterprise layer is used to manage and monitor the provision of web services provided by every enterprise information system. The inter-enterprise layer is provided to manage the IEIP mediated web services which are defined by orchestrating a number of enterprise web services, and are made visible to external users.

2.8 Related work

We briefly survey a number of enabling technologies that underpin IEIP. The purpose of IEIP, the provision of a set of unified access interfaces to a variety of underlying enterprise web services, is not new. There are a number of similar efforts in the domains of data integration [27] and EII/ EAI [24]. Considerable research effort has been devoted to B2B integration [8, 14, 28, 37] which mainly focuses on interoperable business process modelling. Recently, Kajan [25] defined e-commerce as a third wave of B2B where ad-hoc integration is to be supported, with the help of web services. He focuses on the future developments where ontologies will be used [28] within a mediator architecture to match service supply and demand. Similarly, Vetere and Lenzerini [39] advocate the use of ontologies to enable SOAP message translation between providers and customers [17]. This vision aligns B2B with the domain of web services, where research has been focused on the enhancements and extensions of web service standards of description, discovery, and composition. Recent industrial offerings, like IBM's Enterprise Service Bus (ESB) [36], share the vision of automated service discovery and mediation-enabled binding (called "links") and propose to use a separate mediation service which will enable semantic mappings, without actually specifying how mappings will be computed. ESB specification reflects the current status of the semantic web.

Our contribution, the IEIP architecture, enables collaboration between B2B and web services, supported by a web service mediator. IEIP distinguishes itself from previous work by: 1) customer-oriented (private) global views that manage the increasing complexity of interfaces for both providers and requesters; 2) publishing of requester views to the potential service providers; 3) focus on the macro level to create a systematic model of using web services to integrate information assets rather than on the micro level to invent protocols and algorithms for producing and consuming web services; 4) a component-based architecture that makes IEIP an open platform in which different modules, which implement different algorithms, can be plugged in to acquire different processing capabilities. The core of IEIP platform depends on its capability to transform SOAP messages. The component-based architecture makes it easy to plug in a variety of XML transformation algorithms [10, 11, 13].

3 Web services-IEIP architecture

The aim of the IEIP architecture is to manage distributed enterprise computing resources (data and business functionality) at the inter-enterprise level, and to provide operational support to a number of commercial applications in order to simplify the integration of enterprise services. One of the major functions an IEIP must provide is the reconciliation of the enterprise service heterogeneity. We identify four levels of heterogeneity of enterprise services: platform, language, syntactic, and semantic levels. Web services solve the problems of platform and language heterogeneity. They offer greater and more widespread pragmatic internet-scale interoperability than any previous distributed technologies such as CORBA, DCOM and RMI. Interoperability is based on messaging/document-oriented computing rather than distributed object technologies [42]. A web service is described by an XML document (a WSDL document), is designed to process an XML document (a SOAP message) as input, and produce an XML document as output. A client (service request) is able to discover a web service by parsing its description schema automatically or semi-automatically and to interact with a web service by composing and sending an XML document.

Considerable extensions to the core standards of web services have been proposed to resolve the syntactic and semantic heterogeneity. However, none of those extensions have been widely adopted yet. For the purpose of creating a practical solution for real world applications, we have adopted the most popular standards: WSDL for description, and SOAP for interaction, and have enhanced those with mediation.

3.1 Mediator-oriented SOA

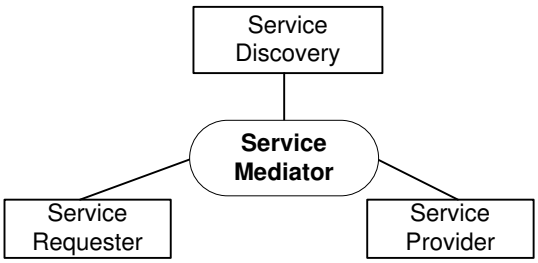

Figure 3. A mediator-oriented SOA architecture, MOSOA

SOA is a very successful solution in both data integration and data exchange [19, 27, 35]. We extend this paradigm and propose a mediator-oriented SOA (MOSOA) model to present *a unified view* or *multiple views* of a specific web service. We refer to the description of a web service as its view. As shown in Figure 3, the modification to the standard SOA model looks trivial, but in fact it resolves the syntactic and semantic heterogeneity. The key idea is the separation of the view of a web service, which is defined and published by its provider, from the view of this web service visible (for discovery and dynamic binding) to service requesters. The objective of using a mediator is similar to the adapter (or wrapper) design pattern that enables a client to access a class in a client-specific way, by transforming the interface of one class into an interface which is expected by the client [22]. However, the implementation of such a scheme is different. In the adapter design pattern, the interfaces are often defined by the service provider. But in the MOSOA model, the mediator defines the required views. In the IEIP architecture, a mediator uses an XML-based schema/message/data mapping [13] to transform SOAP messages between the provider view and the mediator view.

3.1.1 Requester-based view

The first contribution of the IEIP architecture is the introduction of the concept of *requester-based* view of a web service. Normally, the view of a web service is defined by its provider and the requester must use this view precisely to interact with the web service.

Although web services hide the implementation of a service from its requesters, the requesters still have to discover the view (interface) of the service to invoke. Based on the current web services infrastructure, major research efforts have been devoted to the introduction of additional levels of semantics into the description of web services, and to the invention of new dynamic searching/matching approaches to intelligent and automated web service discovery. Most of previous research contributions have been based on the views defined by providers.

The most crucial issue in service discovery is to match service requests to service descriptions, to find the most appropriate service or a list of services. However, the state-of-the-art of web service discovery is still far from sufficient to support fully automated dynamic service binding. Dynamic binding is not a trivial requirement in e-commerce applications. It is common that a service requester is implemented before a service provider releases a service.

In the light of such a request-before-service scenario, we think it is important to support *requester-based views* of a web service. A service requester can either define / register a view of an expected service to be used in a service mediator or acquire a common view of a service from a mediator. As a result, a service requester can make use of a service which will be available later by using the mediator to map its requested view to the actual view of a web service.

3.1.2 Multiple views

The second contribution of IEIP model is the support for *multiple views of a web service*, analogous to the multiple interfaces of a class in the object-oriented programming model [18]. When a service provider publishes/registers one web service, the pertinent mediator can create multiple views (descriptions) of this web service, based on the provider's view and on the requests registered by service requesters. Of course, a service provider can also publish multiple views of a web service implementation.

3.1.3 Customer-oriented global views

Because the MOSOA model enables the separation of views defined by the providers from the views used by the requesters, it can define a global view of a number of sibling web services, or a virtual view of a composite service created by orchestrating multiple web services. IEIP develops the concept of global view further and introduces the concept of *customer-oriented global view*. As a consequence, each requester (customer) can acquire *its own global view* over a number of selected web services. In a real world application, a client often needs to access a number of web services which implement the same or similar business functionality but are operated by different providers. Currently, such a client has to compose a different XML document for each distinct web service and to handle different XML documents returned from different web services. By using the MOSOA model, a mediator maintains a global schema as the description of a virtual web service representing a number of local web services. The mappings between the global schema and each local schema are defined and used

to specify the translation of web service requests and responses. Consequently, a client does not need to compose a distinct request for each web service. Instead, it poses a request against the virtual representative web service according to its global schema. The mediator will translate this global request into a local request based on a local schema. If a web service produces a response, the mediator translates this local response into a global response which can be processed by the specified client according to the client's global schema.

3.2 Web service schema/view management

A view of a web service is a description of its interface and semantics, which is published as an XML document [4]. We refer to such a document as the web service schema (WSS). In the MOSOA model a web service can be described by one or more schemas: one provider-defined schema, and, possibly, several mediated schemas created and managed by the mediators. One or several mediators then constitute a mediated web service. The core of web services mediation is the management of WSSs: (1) creation of mediated schemas; 2) creation of mappings between a mediated schema and a local schema.

From the mediator's perspective there are two ways of creating such a mediated schema: push and pull. The first way, push, defines a mediated schema and pushes this description to web service providers or registries to search for similar web services. A provider can use this (pushed) mediated schema as a reference to develop its own web service. It is not mandatory that a provider must implement its web service exactly as the mediated schema. Instead, it allows the service provider to specify mappings between its own schema and the mediated schema. And the mappings can be implemented either at the mediator-side or at the local service-side. The second way, pull, requests descriptions from web service providers or service registries and classifies web services into categories or clusters using similarity searching [12]. After a number of sibling services are identified, schema matching techniques can be employed to generate a mediated schema and related mappings. The push method is more appropriate for application startup scenarios in which few web services exist, while the pull method is more appropriate for scenarios in which a great number of web services have been developed and are running. In the real world, a hybrid approach would be employed. A mediator initiates itself by searching and pulling schemas of web services to construct a mediated schema to serve a multitude of clients. By learning the requests issued by different clients to a global schema, the mediator is able to adapt to serve clients better and pushes this global schema to service providers who offer their web services via this

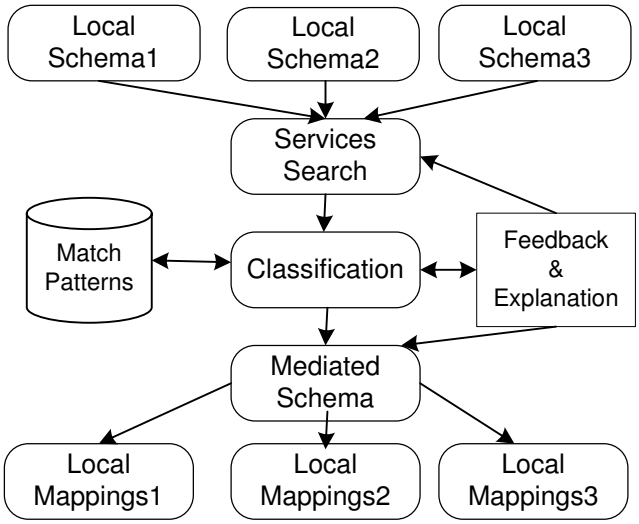

Figure 4. Schema matching and mapping

mediated schema. Once the mediated schema is defined, various approaches can be combined to produce mappings between the mediated schema and each local schema. The overall process of mapping construction is summarised in Figure 4.

Independently of the concrete schema description language, a minimal description of a web service is abstracted as $\mathbf{WSS} = (\mathbf{OP}, \mathbf{IOP}, \mathbf{OOP})$, where OP is a set of operations defined by web services, IOP is the set of input messages (requests), and OOP is the set of output messages (responses). Consequently, web service schema matching and mapping can be defined as $\mathbf{WSSM} = (\mathbf{G}, \mathbf{L}, \mathbf{GL}, \mathbf{LG})$, where G is a global schema, L is the set of local schemas, GL are the mappings from the global schema to local schemas, and LG are the mappings from the local schemas to the global schema.

The primitive elements in a web service schema which are meaningful in the context of schema matching are web service operations. Since web service operations can be grouped into different types of web services, a global schema may have different matching levels: operation-level, service-level (a service consists of multiple operations), and process-level (a process consists of multiple services).

3.3 Mediated web service model

A mediated web service provides virtual interfaces to a number of web services. As shown in Figure 5, a service requester does not need to write specific custom code to handle each distinct local web service. A client sends a web service request composed according to the mediated

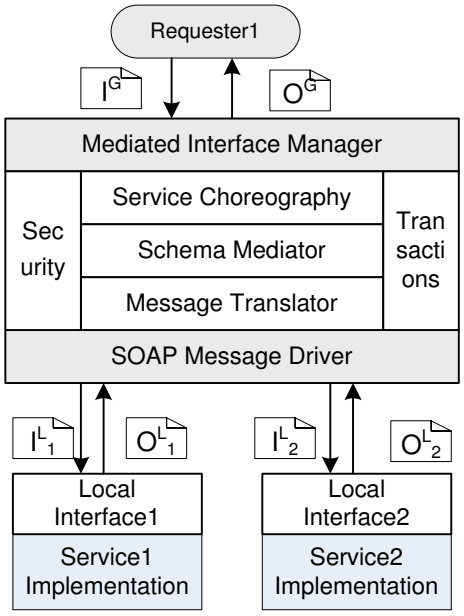

Figure 5. Mediation for Web Services

schema. This request, however, is not processed directly by a local web service. The mediator will translate this global (mediated) request into the specified local request(s). Similarly, a local web service produces a response document which has to be translated by the mediator for the service requester.

When a mediator receives a service request from a client, the first task is to determine and select local web services to process this request. Service selection can be specified either explicitly by the client in the request or implicitly, according to the constraints specified by the client. A service request can be served by multiple services, for example, to retrieve flight information from different airline information services. When local services are selected, the mediator rewrites the original incoming request for each selected local service using global to local mappings (GL). In Figure 5, an incoming request I^G is translated into two local requests, I_1^L and I_2^L, which are delivered to local web services 1 and 2. Local web services process the incoming requests, I_1^L and I_2^L, and produce response documents, O_1^L and O_2^L. These response documents are sent to the mediator. The mediator can either merge the two documents or consolidate them (to remove duplicates), and transforms the consolidated document into the global response document O^G.

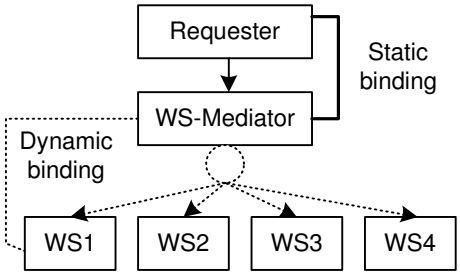

Figure 6. Asynchronous dynamic binding

3.4 Asynchronous dynamic binding

One primary objective of the IEIP is to implement the dynamic binding mechanism. Instead of implementing dynamic binding directly at the requester side, the IEIP uses a mediated web service to implement dynamic binding. As shown in Figure 6, the IEIP supports dynamic binding in an asynchronous manner. The requester binds to a pertinent mediated web service statically. And the web service mediator inside a mediated web service binds to a number of local web services dynamically. The key idea is to separate the binding of a requester to a mediated web service from the binding of a mediated web service to a number of local web services.

3.5 Web services management

IEIP employs a light-weight management scheme to monitor the underlying enterprise web services, based on a message interception mechanism, which intercepts the SOAP message and records it into a log database for analysis. The implementation of IEIP management is based on the web services conformance testing tools developed by the WS-I [6]. At this stage, the IEIP management facility has two tasks: 1) to check the availability of every enterprise web service; 2) to conduct a statistical analysis of the usage of every web service by parsing the message log.

4 IPARK: proof of IEIP

We now outline the implementation of our prototype iPARK, an application providing European-wide parking services by integrating a variety of different parking providers, based on the IEIP architecture.

4.1 Motivation: parking service integration

In most European cities parking spaces are very limited and the physical distribution of car parks is often unrelated to

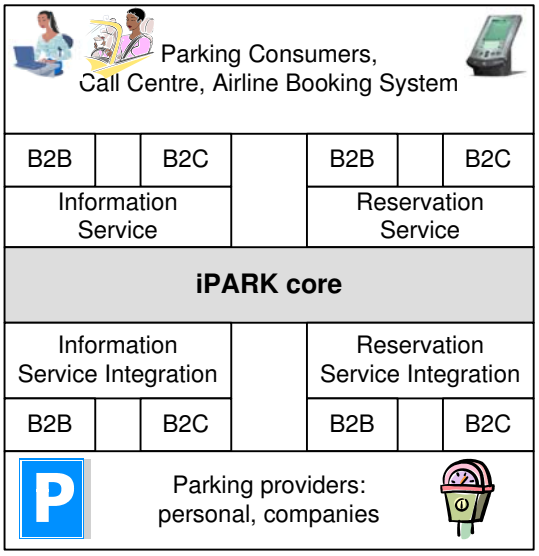

Figure 7. iPARK service delivery

the requirements of the public. The objective of developing iPARK is to optimize the usage of parking spaces and reduce the parking search and the resulting traffic. Parking operators increasingly offer online information systems that allow the customers to find, reserve, and access parking spaces. In order to optimize the usage of these information systems, the operators have to implement their online services in such a way that any potential client (a human user or a software agent) can find and access the desired service flexibly via various network-enabled devices. Parking operators want their services to reach as many customers as possible.

However, due to the intrinsic heterogeneity hidden in different parking operators, end users face many information silos: each operator develops its own information system independently and thus often serves the customers in a way distinct from others. While these services might be consumed in a limited way by human users who crawl through the web to find the access point of a desired service from a multitude of distinct web pages, it is very difficult to access these diverse services from other applications (e.g. hotels, restaurants, festival events) which represent a large number of indirect users. An undesirable consequence is that system heterogeneity hinders potential users from reaching parking space providers who want to deliver their services to more customers by building online services.

Another critical issue we faced is the order in which the relevant internet system components are developed. Our application and the parking service integration framework had to be developed first. Many parking operators will develop new or update existing local parking services later. Thus we need to be able to integrate services which do not exist yet. We started the development of an application aimed at delivering European-wide parking services in the EU IST project e-Parking, and followed in the Swiss CTI project PORAS. It has been well recognized in both projects that iPARK will benefit public transportation, city administration, drivers, and parking operators, by building a pool of parking spaces and integrating services from all categories of parking operators. The long-term objective of iPARK is twofold. It aims to establish a nation-wide (or European-wide) e-Commerce application serving drivers directly or indirectly to find and reserve parking spaces. It also allows businesses to offer their services more flexibly to a larger number of customers. Such an application needs to be based on a service-oriented B2B collaboration platform and capable of interconnecting a large number of parking services provided by a variety of operators.

As shown in Figure 7, iPARK provides unified parking services to end users, based on both B2C (to end users directly) and B2B (to end users via another application, e.g. a theatre ticketing system). On the other hand, iPARK integrates local parking services from a variety of operators also by using both B2B (for parking services companies) and B2C (for personal parking providers).

4.2 Database web services

iPARK is based on a web service-oriented N-tier architecture. It consists of four tiers: presentation, business service, data service, and data. An important feature of iPARK is the creation of a data service tier which is used to provide the unification of data and algorithms. With the introduction of this tier, we achieved three goals: 1) a clean data tier where the code and data are still separated; 2) a meditation tier where the code and data are combined by wrapping access to the underlying database with a set of web services; 3) separation of the underlying database schema and several public virtual schemas defined in terms of web services. Leveraging web services to encapsulate data manipulations, iPARK achieves high interoperability and scalability. Because the actual business logic accesses the data via a set of web services, loose coupling between the database and business logic is realized. Therefore, privacy is protected, because the underlying database schema can be hidden from the (external) business logic.

Complementing traditional data integration, the use of web services to wrap a database facilitates inter-enterprise data integration. Based on current technologies, we believe, the best way to connect an enterprise database to internet-based commercial applications is to wrap it as a set of web services. Such a set defines public views of the underlying database. An enterprise can define different wrapping policies and then expose different sets of web services to

different partners.

4.3 iPARK IEIP implementation

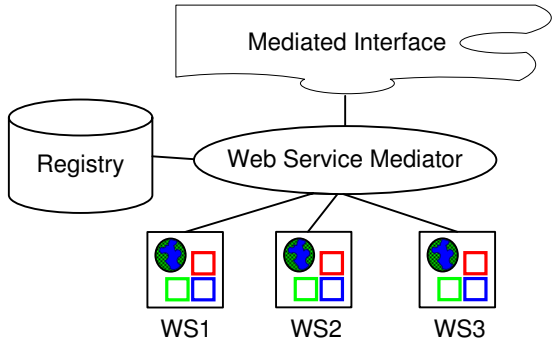

Figure 8. One to many mediation

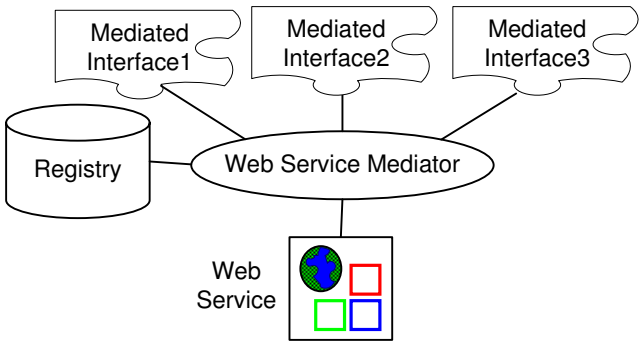

Figure 9. Many to one mediation

iPARK IEIP implements two types of mediators. The first type is a one-to-many mediator, see Figure 8, in which a mediated web service provides one mediated interface to a number of web services. In the second service type, see Figure 9, a mediated web service provides multiple interfaces to access the same web service. Based on these two types, complex mediated web services can be created easily by using web services orchestration and choreography. For example, iPARK generic reservation service uses the one-to-many mediation pattern to implement integrated access to all registered local reservation services provided by different partners. In addition, it uses the many-to-one mediation pattern to provide two different mediated interfaces for iPARK service requesters.

5 SOAP mediation

The mediator uses Intersystems Caché database (www.intersystems.com), and could as well be supported by Apache/Axis. In iPARK clients reserve parking spaces and providers offer reservation services. Clients 1 sends the following simplified message to the mediator.

```
<SOAP-ENV:Body>
<Reservation xmlns="http://serve:9000/ipark">
<DriverID>0986432</DriverID>
<Start>2006-07-01 9:00</Start>
<ReservedTime>120</ReservedTime>
</Reservation>
</SOAP-ENV:Body>
```

The mediator executes the queries used in message translation to convert requests received from the clients to the request format expected by the providers. In this case, the mediator transforms the message and forwards it to two providers. Provider 1 receives:

```
<SOAP-ENV:Body>
<Reserve xmlns="http://vpark:9000/ipark">
<DriverName>Suo Cong</DriverName>
<Billing>Wallisstr.  473, Zurich</Billing>
<StartDate>2006-07-01</StartDate>
<StartTime>9:00</StartTime>
<EndDate>2006-07-01</EndDate>
<EndTime>11:00</EndTime>
<NumberOfSpace>1</NumberOfSpace>
</Reserve>
</SOAP-ENV:Body>
```

and Provider 2 is sent the following.

```
<SOAP-ENV:Body>
<ResSpace xmlns="http://vpark:9000/ipark">
<IPARKUserID>suocong</IPARKUserID>
<StartDate>2006-07-01</StartDate>
<EntranceTime>9:00</EntranceTime>
<RequestedTime>120</RequestedTime>
</ResSpace>
</SOAP-ENV:Body>
```

When a response is produced by any of the providers, the mediator transforms it to match the client's global view and routes the reply back to the client. In the mediated scenario a client uses one global view of the available web services, and can take advantage of services offered by several providers. In fact, iPARK has a registry of several requestor views, and several provider views, and executes *m:n mappings* between clients and providers. The iPARK database supports the translation of SOAP mappings which are expressed as XML queries.

6 Discussion and conclusions

We presented a novel platform for loosely-coupled integration of enterprise services at inter-enterprise level, IEIP, which uses a mediator-oriented SOA model. The key idea of our work was to separate the view of a web service, as defined by its provider, from the views created by a medi-

ator, and used by a service consumer. Based on the mediated views of a web service, requesters are freed from direct binding to the underlying enterprise web services, and can use the web service infrastructure more flexibly.

The implementation of mediated web services is based on the techniques of schema mapping and data exchange. Most research in this area has been devoted to relational schema mapping and data exchange [17, 19, 34, 35]. Research on XML-based schema mapping and data exchange [10, 11, 13] is still a hot area, and our work extends this paradigm by making the issue of mediation explicit. Our main interest lies in exploiting XML techniques to implement the transformation of SOAP request/response messages between (mediated) web services. Our current IEIP implementation only supports limited semi-automated generation of mediated schemas and schema mappings. Our solution provides some of the functionality envisaged for ESB [36], but focuses on the tasks and techniques used by a mediator, which lie outside ESB focus.

In the future we will work on improving the mechanisms used in schema mediation. We require more flexible support for authentication and security, and ad-hoc service composition. The long-term aim is to support web service usage scenarios which offer flexibility, fit-for-purpose, as well as quality assurance. Adaptation of mappings to environmental change is another concern that needs to be addressed.

ACKNOWLEDGEMENTS: This work was supported by the Swiss CTI/KTI Project PORAS 6704.2 ENS-ES and by an EU Marie-Curie Fellowship (E.H.). We thank Moira Norrie for her valuable comments.

References

[1] UDDI, 2002. www.uddi.org.

[2] BPEL v. 1.1, 2003. www-128.ibm.com/developerworks/library/specification/wsbpel/.

[3] SOAP, 2003. www.w3.org/2000/xp/Group.

[4] Web Services Architecture, 2004. www.w3.org/TR/ws-arch.

[5] WS Choreography, 2005. www.w3.org/2002/ws/chor.

[6] Web Services Interoperability Org., 2006. www.ws-i.org.

[7] WS Description Language, 2006. www.w3.org/2002/ws/desc.

[8] S. Aissi et al. E-Business Process Modeling: The next big step. *IEEE Computer*, 35(5), 2002.

[9] G. Alonso et al. *Web Services: Concepts, Architectures and Applications*. Springer, 2004.

[10] S. Amer-Yahia and Y. Kotidis. A Web-Services Architecture for Efficient XML Data Exchange. In *ICDE*, 2004.

[11] M. Arenas and L. Libkin. XML Data Exchange: Consistency and Query Answering. In *PODS*, 2005.

[12] T. Berners-Lee. Roadmap for Web Services. www.w3.org/DesignIssues/WebServices.html.

[13] A. Boukottaya et al. Automating XML documents Transformations: A conceptual modeling based approach. In *1st Asian-Pacific Conf. on Concept. Model.*, 2004.

[14] C. Bussler. B2B Integration Technology Architecture. In *WECWIS*, 2002.

[15] F. Casati et al. Business-Oriented Management of Web Services. *CACM*, 46(10), 2003.

[16] F. Curbera et al. The Next Step in Web Services. *CACM*, 46(10), 2003.

[17] R. Dhamankar et al. iMAP: Discovering Complex Semantic Matches between Database Schemas. In *SIGMOD*, 2004.

[18] B. Eckel. *Thinking in Java*. Prentice Hall, 1998.

[19] R. Fagin et al. Data Exchange: Getting to the Core. *TODS*, 30(1), 2005.

[20] D. Fensel and C. Bussler. The Web Service Modeling Framework WSMF. *Electronic Commerce Research and Applications*, 1(2), 2002.

[21] P. Fremantle et al. Enterprise Services. *CACM*, 45(10), 2002.

[22] E. Gamma et al. *Design Patterns*. Addison-Wesley, 1995.

[23] J. Gray and M. Compton. A Call to ARMS. *ACM QUEUE*, 3(3), 2005.

[24] A. Halevy et al. Enterprise Information Integration: Successes, Challenges, and Controversies. In *SIGMOD*, 2005.

[25] E. Kajan. The maturity of open systems for b2b. *SIGecom Exch.*, 5(2), 2004.

[26] J. Lee et al. Enterprise Integration with ERP and EAI. *CACM*, 46(2), 2003.

[27] M. Lenzerini. Data integration: A theoretical perspective. In *PODS*, 2002.

[28] B. Medjahed et al. Business-to-business interactions: issues and enabling technologies. *VLDB Journal*, 12(1), 2003.

[29] B. Medjahed et al. Infrastructure for E-Government Web Services. *IEEE Internet Computing*, 7(1), 2003.

[30] OWL-S. Semantic Markup for Web Services, 2004. www.daml.org/services/owl-s/1.1B/owl-s.pdf.

[31] M. Paolucci et al. Semantic Matching of Web Services Capabilities. In *ISWC*, 2002.

[32] C. Peltz. Web Services Orchestration and Choreography . *IEEE Computer*, 36(10), 2003.

[33] C. Petrie and C. Bussler. Service Agents and Virtual Enterprises: A Survey. *IEEE Internet Computing*, 7(4), 2003.

[34] L. Popa et al. Translating Web Data. In *VLDB*, 2002.

[35] E. Rahm and P. Bernstein. A survey of approaches to automatic schema matching. *VLDB Journal*, 10(4), 2001.

[36] M.-T. Schmidt, B. Hutchison, P. Lambros, and R. Phippen. The Enterprise Service Bus: Making service-oriented architecture real. *IBM Systems Journal*, 44(4), 2005.

[37] S. Shim et al. Business-to-Business E-Commerce Frameworks. *IEEE Computer*, 33(10), 2000.

[38] K. Sycara et al. Automated Discovery, Interaction and Composition of Semantic Web Services. *Journal of Web Semantics*, 1(1), 2003.

[39] G. Vetere and M. Lenzerini. Models for semantic interoperability in service-oriented architectures. *IBM Systems Journal*, 44(4), 2005.

[40] S. Vinoski. Web Services Interaction Models, Part 1. *IEEE Internet Computing*, 6(3), 2002.

[41] S. Vinoski. Web Services Interaction Modes, Part 2. *IEEE Internet Computing*, 6(4), 2002.

[42] W. Vogels. Web Services Are Not Distributed Objects. *IEEE Internet Computing*, 7(6), 2003.