A SERVICE INTERFACE DEFINITIONS CATALOGUE FOR VIRTUAL ENTERPRISES IN TOURISM

César Garita, Ersin C. Kaletas, Hamideh Afsarmanesh, L.O. Hertzberger
University of Amsterdam, Kruislaan 403, 1098 SJ Amsterdam, The Netherlands
{cesar,kaletas,hamideh,bob}@science.uva.nl

A Virtual Enterprise (VE) in the tourism sector can be defined as a temporary consortium of service provider organizations (e.g. traveling agencies, accommodation providers, etc.), that join their skills and resources in order to offer integrated value-added services. Value-added services are composed of basic services and/or other value-added services that are interconnected to create new higher-level tourism services. In order to support the required level of inter-operability among basic and value-added services offered by different enterprises, it is necessary to define standard semantic models regarding specific service interface definitions and data type structures. These standard service interface definitions can be made available through a common web-based catalogue. This paper focuses on the design and implementation of a Service Interface Definitions Catalogue component aimed at the support of interoperability of value-added services in tourism VEs.

1. INTRODUCTION

With the on-going rise of commercial globalization and market aggressiveness, tourism industries and service providers face the need to strongly collaborate and share their expertise and resources, as well as their costs and risks. In fact, basic collaborations among tourism enterprises based on web information technologies have taken place for several years, since many of them already offer different services through individual or common web sites that allow for instance: gathering of general information about tourism facilities, hotel search and room reservations in a certain geographical area, purchase of plane tickets, selection of organized excursion packages, etc. Typically, the information is presented to end users through static or dynamically generated html pages, which interface with the local or remote company systems using communication mechanisms such as CGI, email, fax, etc. In general, although these services are well established and have a wide acceptance among tourists, the level of integration and interoperability among these services is still quite fragmented and rigid (Afsarmanesh, 2000).

Thus, even though these Web-enabled tourism services are widespread and have substantially broadened the market opportunity horizon for the companies in the tourism sector, there is still a prominent need to support more advanced

collaboration scenarios among these companies. The support for *value-added services* (VASS) is an example of these advanced collaboration scenarios, through which travel agencies would offer aggregated or *value-added-services* composed of components supplied by a number of different organizations. For instance, a value-added service such as “booking complete journeys”, may include booking several means of traveling, arrangement of hotel reservations, leisure tour bookings, etc (see also (Garita, 2001)).

In order to provide *on-line value-added-services* in the tourism sector, both service provider and service requester enterprises must have access to an integrated *federation of services* that are made available for specific purposes (and under specific circumstances) by many different organizations (see Figure 1). This federation of services logically represents a group of distributed and heterogeneous tourism services and resources that are presented to end users through a single infrastructure. This infrastructure would provide an easy and common access to a massive collection of distributed resources and services that would be able to interoperate and be combined in order to build new higher-level value-added services. Clearly, advanced ICT models and mechanisms must be applied in order to standardize and integrate the disparate models and service implementations that are internally used by the tourism enterprises world-wide, and to allow the flexible configuration, execution and follow up of these value-added services involving different enterprises.

Therefore, the concept of Virtual Enterprise (VE) can be applied to the tourism domain in order to support the proper interaction and cooperation of alliances of existing enterprises towards the accomplishment of a common goal (see also (Camarinha-Matos, 2000), (Goranson, 1999)). In particular, a Virtual Enterprise in the tourism sector can be defined as a temporary consortium of different service provider organizations (e.g. traveling agencies, accommodation providers, organizers of leisure programs, public tourism organizations, etc.), that join their skills and resources in order to offer an integrated and aggregated service

Figure 1 - Federation of heterogeneous and distributed tourism services.

(Afsarmanesh, 2000). In this way, enterprises can offer new tourism services that represent tailored solutions to customers, and they would also be able to participate more actively in certain business scenarios that are currently not well supported by existing information technology infrastructures in the tourism area.

Please notice that to become widely accepted, the design of the common federation layer must follow standardized *semantic models* regarding the service interface definitions and their associated *data type structures*. Namely, the meaning of the service interface definitions and their parameters must be documented and commonly agreed by tourism enterprises in the VE within the target geographical region of the system. Here, the use of already existing *ontologies* and *data types definitions* in the tourism areas is strongly needed. The standard service interface definitions can be made available through a common *Service Interface Definition Catalogue*, which could be accessed via web interfaces. The data type structures needed by the service interfaces, could also be made available through this catalogue.

In this context, this paper focuses on the internal system design and final implementation of a Service Interface Definitions Catalogue (SERV-CAT) for the support of VE in tourism. Part of the system described in this paper was implemented within the European 5FP project FETISH. In short, the FETISH project aims at the integration of the fragmented tourism information systems and their IT-based services into a federation of distributed resources that are presented through a single infrastructure to end users and other service provider enterprises.

The rest of this paper is organized as follows. Section 2 describes a reference structure of a support infrastructure for VEs in tourism. Section 3 presents the detailed design and implementation of the SERV-CAT system as one of the main components of this reference infrastructure. Finally, Section 4 summarizes the achieved results after the implementation of the SERV-CAT component.

2. A SUPPORT INFRASTRUCTURE FOR TOURISM VES

It is clear that the support for complex VE collaborations involves the application of advanced information models and technologies, including distributed business processes, workflow management, ontology definitions, standard data models, Internet facilities, middleware components, multi-agent approaches, and advanced distributed information management techniques among others (Garita, 2002). In this section, a proposed structure for a VE support platform in tourism incorporating some of these technologies is presented.

The diagram presented in Figure 2 depicts the main structural components of the proposed system. The system architecture presented here is internally based on Java/Jini technologies. In short, the Jini architecture has been specifically designed for deploying and using generic services in a distributed network (Edwards, 1999), (Freeman, 1999), (Arnold, 1999). In this way, services can be plugged/unplugged into network directories, and specific lookup mechanisms allow the retrieval of services to be used and executed by client applications or end users. In the Jini approach, a service proxy must be defined for each actual tourism service (which can be already implemented in heterogeneous hardware/software platforms). A

Figure 2 -General architecture components for a VE platform for tourism sector.

proxy represents a pointer or a remote reference to an actual service (or a wrapper to it), and acts as a front-end for other systems or applications that ultimately need to access the tourism service implementation. Thus, service requesters must be able to retrieve these proxies and integrate them into their own systems. In other words, the actual tourism services will be accessed and executed through proxies located in other enterprises.

Each major component of Figure 2 is briefly described in the paragraphs below.

First of all, please notice that the end users and service providers have access to several system toolkit components, which can be downloaded via a web site. Namely, the system toolkit includes all the tools and applications that are made available to service providers and end-users in order to facilitate their access to services and/or adaptation of the basic services (and value-added-services) that they provide in compliance with the global federation specifications. The tools and applications in the system toolkit include four main components:

- The *Service Interface Definitions Catalogue* (SERV-CAT) tools. The main objective of the SERV-CAT tools is to allow service requesters and providers to

access the precise service and object definitions for the tourism services that are handled within the federation.

- The *VAS Process Management System* (VAS/BP Manager). This system allows the definition and execution of business processes (BPs) representing value-added services in the context of Virtual Enterprises. In the remaining of this paper we refer to this toolkit component as PROMAN (Camarinha-Matos, 2001).
- *Programming Model* (PM). This component represents a set of APIs and documents that can assist service providers that wish to extend their service implementations with wrappers and proxies to be plugged into the global service federation. For instance, this “model” can include templates to facilitate the generation of service wrappers and proxies complying with the JINI technology.
- *Access Manager System* (ACMS). This module serves as the entry gateway for the end users and service providers to the system, and supports interaction scenarios in which some kind of “human intervention” may be required.

Furthermore, the set of service proxies that are available within the federation are organized and managed within “Service Directory Nodes” of the system architecture. The main components of this node are described below:

- *Distributed Service Lookup Manager* - DSLM. The DSLM component represents an advanced look-up service manager facility that extends the basic JINI look-up mechanism in order to allow the distributed access and management of service proxies, that are physically distributed and located at different nodes. The management of local proxies at each node is carried out by the SLM (Service Lookup Manager) component using built-in JINI look-up services. In general, the directory nodes are logically inter-linked following an acyclic graph or a hierarchical tree-like organization, which represents the distributed service directory of the global service federation.
- *Federated Access Rights Manager* (FARM). This module allows the definition and validation of access rights to services defined at the level of service proxy in the general context of the global federation, as well as services defined within the Virtual Enterprises. For instance, service providers can specify that their given service proxies can be available for lookup only to certain specific sets of service requesters or only to specific VE members. Such access rights provision is extremely important for the support of VE infrastructures.

Besides the system toolkit, DSLM node and FARM components, there are two other important elements addressing different aspects of the data and meta-data that need to be managed in order to support the operation of the tourism VE infrastructure that are described below:

- *Ontology Manager*. This component basically represents the knowledge base repository in which the definition of all concepts belonging to the tourism specific domain are represented and stored. New service / object catalogue definitions that are generated here, are pushed into the federation repository
- *Federation Repository*. The Federation Repository represents a database containing all the information that is required to support the operation of the global service federation, and that needs to be accessed by different modules and end users. The main components of the repository contain the data related to: service interface catalogue, object interface catalogue, service providers' information, registered end-users information, and VE-related information.

Finally, the General Standardization Committee represents a group of people who must decide if a new tourism service-interface definition is accepted to become a part of the interface definition specifications for generic federation services.

Based on the system structure defined in Figure 2, it is possible to illustrate different use-case scenarios, such as the procedures for definition of a new federation service, registration of a basic service in the federation, distributed service look-up, and user access to an existing service. Furthermore, the system structure can also support more complicated cases, such as the definition, registration, and access to value-added services within the global federation. Some of these scenarios and functionalities have been addressed in previous publications (Afsarmanesh, 2000),(Camarinha-Matos, 2001). In the next section, the internal design and implementation of the SERV-CAT component of the presented reference infrastructure is described in details.

3. SERV-CAT DESIGN AND DEVELOPMENT

As mentioned previously, SERV-CAT applications allow end users to access the Service Interface Definitions Catalogue. The SERV-CAT architecture embodies a catalogue database (part of the Federation Repository) containing the service interface definitions, software components to access the catalogue, application servers to manage different information management requests, and several graphical and programming interfaces for end users and applications.

The end-users of the service federation should be able to browse service interface definitions information via for instance, a web interface. This information also needs to be accessed by other internal components of the tourism infrastructure to perform some validation actions that may be necessary for instance, when defining a business process for a value-added service.

In the following subsections, the SERV-CAT architecture and its components are described in details.

3.1 General SERV-CAT Architecture

In general, multi-threaded web-based applications can be defined and modeled by a three-tier architecture, also called client-agent-server architecture. In this architecture the clients are only concerned with presentation services, such as the user interface design specificities. As such, the agent, referred to as application server in this paper, processes the application logic for the client, hiding the underlying implementation and access details of the server tier, and adding higher-level support functionalities for the client. The SERV-CAT architecture has been designed following such a multi-tier approach (see Figure 3). By making a clear separation between the client and application server tiers, the client components are designed as relatively light processes, which are mostly concerned with user-interface details, and representation and formatting of the data. The functionality itself is supported in the middle tier (Application Server), which in turn gets relieved of all the specific details of the end-user and module interfaces.

Figure 3 - General SERV-CAT 3-tier architecture.

Following this approach, the tiers in the SERV-CAT architecture and the components in each tier are described in the next sub-sections.

3.2 SERV-CAT Client Tier

The client tier of SERV-CAT architecture contains the components that directly interact either with the end users or with other system components e.g. PROMAN system. From a general point of view, the SERV-CAT Client Tier is composed of two types of interfaces: graphical interfaces for end-users, and programming interfaces for other system components.

The graphical interfaces support functionalities required by ordinary end users and administrator users. Currently, there is one main client interface for ordinary end users: the Service Interface Definitions Catalogue Browser (SERV-BROWSER). SERV-BROWSER allows a regular user, which can be a service provider, a travel agency, etc., to access the contents of the Service Interface Definitions Catalogue. The functionalities provided by the SERV-BROWSER to end users include:

- Downloading a jar file (Java archive file containing compiled Java code), which contains the compiled Java codes of the data types in SERV-CAT.
- Viewing the service packages in the catalogue and displaying their corresponding descriptions. Service interface definitions in SERV-CAT are logically grouped into packages, which correspond to packages in Java.
- Viewing the types in each package. Service interfaces in SERV-CAT are represented as types in Java programming language, which can be either a Java class or a Java interface. Detailed descriptions of each type can be obtained via SERV-BROWSER, including: name of the type, super classes, package name, access modifiers of this type, textual description, and a mark indicating whether this type is representing a value-added service.
- Viewing the members of each type. The activities or events associated to a service are represented as the members of a type in the Java programming language. A member can be either a field or a method. The detailed description of each member can also be obtained, including for instance: name of the

member, type of a field or return type of a method, access and other modifiers of this member, and the parameter list if the member is a method. Details of the type definitions (i.e. type of a field, return type of a method, or parameter types of a method) can be displayed in a separate window. Using this window, users can both browse and download the Java source code if the type is a complex type, or the XML description if the type is a primitive type.

Please notice that ordinary users can only view the catalogue contents; they are not allowed to modify the contents of the catalogue.

A snapshot of the browsing interface is shown in Figure 4. As can be seen in this figure, the interface that allows the user to browse the service catalogue information is divided in three main areas. In the first area (left), the user can browse the service packages that are available. In the second area (middle), the user can see the service types e.g. Java classes and interfaces, that are defined within a given package. Finally, for a given type, the user can see which methods and field members are defined (right area). When the user double-clicks on a selected item, the detailed information associated with that specific item is displayed. For instance, in the case of the methods, the user can see the parameters, modifiers, return type, etc. that are defined according to a given method (see Figure 4).

The SERV-BROWSER is a Web-enabled application. It can run both as a stand-alone Java application and as a Java applet within a Web browser. It communicates with the middle tier of SERV-CAT via RMI.

Besides the regular end-user SERV-BROWSER tool, the client tier of the SERV-CAT application also includes a specific component for system administrators, namely the SERV-ADMIN. This administrator graphical interface supports the extended functionality required by administrator users. In addition to providing all of the SERV-BROWSER functionalities, SERV-ADMIN allows the administrators to modify the contents of the SERV-CAT; e.g. to insert new service interface definitions to SERV-CAT, or edit an already existing service interface definition.

Finally, the SERV-CAT client tier also includes lower-level application programmers interfaces (APIs). These interfaces represent high-level information management functions specifically developed to support other system components, such as PROMAN. Currently, PROMAN and SERV-CAT are integrated through the SERV-CAT programming interfaces. PROMAN uses the SERV-CAT functions to access the Service Interface Definitions Catalogue in order to obtain the available service interface definitions during the definition of a value added service.

3.3 SERV-CAT Application Server Tier

The middle tier of SERV-CAT contains specific application servers, which encapsulate the application logic for the SERV-CAT Client Tier components. Currently, there are four server tier components: SERV-CAT Manager Factory, SERV-CAT Browser Manager, SERV-CAT Admin Manager, and SERV-CAT Data Server, as described below (see also Figure 3):

Figure 4 - End user interface for Service Interface Browser.

- **SERV-CAT Manager Factory.** The application server tier is internally designed following the factory pattern (Gamma, 1995),(Sun-Microsystems, 2001). In this context, the SERV-CAT Manager Factory is a server, to which client applications (SERV-BROWSER and SERV-ADMIN) connect via RMI, and request a manager instance to access the SERV-CAT. The Manager Factory creates a new manager instance for each request coming from the client application; i.e. a Browser Manager for a request from a SERV-BROWSER instance or an Admin Manager for a request from a SERV-ADMIN instance. The remote reference to the newly created manager instance is passed to the client application, and from this point on the manager handles the communication between the client application and the catalogue.
- **SERV-CAT Browser Manager.** The SERV-CAT Browser Manager provides the functionality to support the client applications being executed by ordinary users (such as SERV-BROWSER). The Browser Manager application server allows read-only access to the catalogue. The information retrieval functions are encapsulated in methods, each retrieving a specific piece of information from the catalogue. Below, some of the methods available to client applications are given as examples:
 - `getInterface`: to retrieve a specific interface or all interfaces.
 - `getTypeMembers`: to retrieve the members (methods and fields) in a given type.
 - `getParameter`: to retrieve a set of parameters in a given method.
 - `getInterfaceSourceCode`: to retrieve the Java source code for a given interface.
- **SERV-CAT Admin Manager.** The SERV-CAT Admin Manager provides the functionality to support the administrative components of the client tier (e.g.

SERV-ADMIN). Please notice that besides all SERV-CAT Browser Manager functionality, the SERV-CAT administrator provides the functionality to modify the contents of the SERV-CAT database. The SERV-CAT Admin Manager is password protected.

- **SERV-CAT Data Server.** The SERV-CAT Data Server handles all database-related operations for the Service Interface Definitions Catalogue. The data server is the only component directly accessing the catalogue database. Requests to the remote SERV-CAT Browser/Admin Manager are forwarded to the underlying data server. The database operations are performed using a connection object obtained from the connection pool in the back-end tier. The Data Server provides the functionality to support all of the client tools, such as SERV-BROWSER and SERV-ADMIN.

Besides the application servers described above, the application server tier of SERV-CAT contains an extra component (server launcher) and a Java package, which are used to support the proper operation of the application servers and to provide some miscellaneous functionalities. In particular, the SERV-CAT Server Launcher is a Java program that starts up the SERV-CAT application by initializing and launching a new SERV-CAT Manager Factory. Furthermore, based on the data structures defined for the Service Interface Definitions Catalogue (the catalogue database model), a Java package was created, which contains a set of Java classes that exactly correspond with the data structures defined in the catalogue database model. Other system components that need to interact with SERV-CAT use this package in order to exchange Service Interface Definitions Catalogue information in the common format of Java objects.

3.4 SERV-CAT Database Back-end Tier

The database back-end tier of the SERV-CAT architecture corresponds to the SERV-CAT DB containing the data repositories and the SERV-CAT Connection Manager handling the connections to the database. The Oracle DBMS is currently used for the SERV-CAT DB. Since the DB is accessed exclusively through JDBC, any other JDBC-compliant database engine can be easily used instead of Oracle, if necessary.

The SERV-CAT DB contains the definitions of the service interfaces available in the system. In the catalogue database, the service interfaces are basically modeled following the syntax of the Java programming language. Thus, the catalogue database model describes packages of service interfaces, fields and methods of each interface, types of fields, parameters and return types of methods, and exceptions thrown by the methods.

The relational database model of the SERV-CAT DB is given in Figure 5. A detailed description of this diagram is outside the aim of this paper; instead, a short characterization of each entity is described below. In brief, the entity TFET_TYPES represents a root class for the main data types that are handled in the current system. The entity TFET_FET_TYPES is a generalization of the concepts of both Java classes (TFET_CLASSES) and interfaces (TFET_INTERFACES). Please notice that both classes and interfaces can be defined recursively in terms of other types, and that they are encapsulated in packages (TFET_PACKAGES). Furthermore,

Figure 5 -Entity Relationship Diagram for Service Interface Catalogue Definitions.

both classes and interfaces are associated with class members (TFET_MEMBERS), which in turn can be either methods (TFET_METHOD_MEMBER) or fields (TFET_FIELD_MEMBER). In general, type members can be associated (through TFET_QUALIFIERS) with zero or more specific qualifier types (TFET_QUA_TYPES). The qualifiers types represent for instance private, public and protected declarations of type members. The method members are also associated with certain parameters (TFET_PARAMETERS) of a given type.

Besides the catalogue database, a second component is involved in the back-end tier: the SERV-CAT Connection Manager. The Connection Manager handles the connections to the database by using a connection pooling mechanism.

Finally, the SERV-CAT system functionality has also been extensively documented using UML Use Case Diagrams and Use Case Models (Fowler, 1999),(Larman, 1998).

4. CONCLUSIONS AND FUTURE WORK

The Virtual Enterprise paradigm can be applied to the tourism sector in order to support the provision of integrated value-added services as a composition of services provided in turn by different independent organizations. In this context, a reference model for web-based tourism infrastructure to support VE collaborations was proposed and described. One of the main components of the proposed VE infrastructure is the Service Interface Definitions Catalogue (SERV-CAT), through which standard service interfaces and data type can be formally specified.

Therefore, this paper focused on the description of the internal system design and implementation of the SERV-CAT three-tier architecture, which properly supports

the requirements identified for different kinds of end-users and client applications. The SERV-CAT tools provide the necessary facilities for browsing, downloading, and maintaining the common interface specifications. All these functionalities aim at the support of value-added service definition and interoperability within a VE framework.

Finally, future extensions that can be considered for the SERV-CAT architecture described in this paper include: a link with ontology system, manipulation of VE configuration topology and VE members information, application of WSDL, and incorporation of searching capabilities for the SERV-CAT applications.

5. REFERENCES

1. Afsarmanesh, H., and Camarinha-Matos, L. M. "Future Smart-Organisations: a Virtual Tourism Enterprise." *1st International Conference on Web Information System Engineering - WISE'2000*, Hong Kong, China, 2000.
2. Arnold, K., O'Sullivan, B., Scheifler, R., Waldo, J., and Wollrath, A. *The Jini Specification*, Addison-Wesley, 1999.
3. Camarinha-Matos, L., Afsarmanesh, H., Kaletas, E. C., and Cardoso, T. "Service Federation in Virtual Organizations." *11th International PROLAMAT Conference on Digital Enterprise - New Challenges*, Budapest, Hungary, 2001, 305-324.
4. Camarinha-Matos, L. M., Afsarmanesh, H., and Rabelo, R. "E-Business and Virtual Enterprises - Managing Business-to-Business Cooperation.", Kluwer Academic Publishers, Boston, 2000, 552.
5. Edwards, W. K. *Core Jini*, Prentice Hall, 1999.
6. Fowler, M., and Scott, K. *UML Distilled: Applying the Standard Object Modeling Language*, Addison Wesley, 1999.
7. Freeman, E., Hupfer, S., and Arnold, K. *JavaSpaces Principles, Patterns and Practice*, Addison Wesley, 1999.
8. Gamma, E., Helm, R., Johnson, R., and Vlissides, J. *Design Patterns: Elements of Reusable Object-oriented Software*, Addison-Wesley, 1995.
9. Garita, C. "Federated Information Management for Virtual Enterprises," Doctoral Thesis, University of Amsterdam, Amsterdam, The Netherlands, 2001.
10. Garita, C., Afsarmanesh, H., and Hertzberger, L. O. "A Survey of Distributed Information Management Approaches for Virtual Enterprise Infrastructures." *Managing Virtual Web Organizations in the 21st Century: Issues and Challenges*, U. J. Franke, ed., Idea Group Publishing, 2002, 164-183.
11. Goranson, H. T. *The Agile Virtual Enterprise: Cases, Metrics, Tools*, Greenwood Publishing Group, Inc., 1999.
12. Larman, C. *Applying UML and Patterns*, Prentice Hall, 1998.
13. Sun-Microsystems. "Factory Pattern", <http://java.sun.com/j2se/1.3/docs/guide/rmi/Factory.html>, 2001.