Federated Information Management for virtual enterprises

Garita Rodriguez, C.O.

Publication date
2001

Citation for published version (APA):
Chapter 5

Extension and Application of the DIMS Federated Architecture to Support Service-oriented VEs

5.1 Introduction

In previous chapters of this dissertation, the requirement analysis and internal design of the Distributed Information Management System (DIMS) to support VEs have been presented, and it was demonstrated how the DIMS can properly support different VE collaborative scenarios, with special focus on Small- and Medium-size Enterprises (SMEs) in the industrial manufacturing sector. However, other application domains, though may have some commonalities, will certainly pose different information management requirements in terms of for instance, data modeling, data access functionalities, and use of specific IT standards and tools. For instance, some of the federated information management requirements that have been identified for the support of VEs in the manufacturing field, may have different characteristics when considering service-oriented VEs. Furthermore, some kinds of VEs may be best supported by the extensive application of Web-based technology and standards in all the components of the VCL infrastructure, including also the DIMS. These considerations will have an impact on the potential application of the DIMS architecture to different VE environments. Nevertheless, the fundamental need for the federated information management architecture is proven to be present in a large variety of VE applications within both production and service-oriented domains.

Therefore, the main objective of this chapter is to demonstrate and validate how the general DIMS federated architecture and implementation platform that has been presented in Chapters 3 and 4 mainly focused on the industrial manufacturing sector,
can be tailored, extended, and applied in order to cope with the requirements found in other VE service-industry application domains, taking as an example the tourism sector. Therefore, this chapter focuses on the description of an extended DIMS for proper support of the tourism service industry. Only partial implementation of the aspects introduced in this chapter are achieved within the European 5FP project FETISH, as will be described in Section 5.2 of this chapter. Furthermore, it must be also mentioned that the detailed design and development of some of the extensions described in this chapter is still in progress.

The structure of this chapter is organized as follows. Section 5.2 focuses on the description of the VE paradigm in the tourism sector, and the major structural components that are required within a reference VE support platform for this domain. Furthermore, Section 5.3 describes the diverse set of information management requirements that have been identified for VEs in the tourism sector. Section 5.4 shows how the design and implementation of the DIMS can be extended and applied to support VEs in this particular domain. Finally, Section 5.5 summarizes the main conclusions of this chapter.

### 5.2 A Reference Infrastructure for Tourism VEs

This section introduces some generalities about the application of the VE paradigm in the tourism domain, and proposes a general structure for a VE support platform in this sector. A part of the work described in this chapter has been carried out within the EC-funded 5FP project FETISH [8].

#### 5.2.1 The VE Paradigm in Tourism

In order to remain competitive in the aggressive global market, 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 through 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 [73, 75].

Namely, even though these Web-enabled tourism services are widely spread and have substantially broadened the market-opportunity horizon for these companies, there is still a need to support much more advanced collaboration scenarios in which these enterprises can for instance, dynamically look for partners to fulfil new business
opportunities, and offer aggregated or *value-added* services based on a combination of resources provided by a number of different enterprises.

In other words, services offered by these enterprises via Internet are currently provided on quite independent basis, and there is a poor support for end-user scenarios involving the booking of "complete journeys", in which different tourism services from various companies need to be integrated and monitored along a given period of time. For example, a complete journey for a person attending to a conference in a given location can include booking of flights, different hotel reservations, train ticket reservation, car rental, local excursions, etc. For this case, a *smart* tourism agency may want to offer some kind of "generic conference plan" to potential customers, who can then progressively customize this initial plan according to their own itinerary, personal preferences and other specific requirements [73]. The customers may be able to define their own instance of this plan by having access to an integrated *federation* of services that are made available for specific purposes (and under specific circumstances) by many different organizations (see Figure 5.1).

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 con-
figuration, execution and follow up of these value-added services involving different enterprises.

Therefore, the concept of Virtual Enterprise 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. In particular, a Virtual Enterprise in the tourism sector can be defined as a temporary consortium of different service provider organizations (e.g. travel 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, and whose cooperation is supported by computer networks [8]. In this way, enterprises can offer new tourism services that represent tailored and more specific 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.

Furthermore, it is clear that the support for this kind of complex VE collaborations also 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. In the next section, a proposed structure for a VE support platform incorporating some of these technologies is presented.

5.2.2 A Support Infrastructure for Tourism VEs

This section presents some general design considerations regarding the support of the VE paradigm in the tourism sector. Furthermore, this section presents a reference "system structure" for the target VE platform in the tourism domain.

General Considerations

In order to develop a VE platform supporting the concept of creation and execution of value-added services in the tourism sector, it is necessary to provide the existing enterprises (e.g. service providers) with a mechanism that allows them to plug/unplug their actual service implementations into a global service federation (represented through for instance, distributed directory nodes), from where they can be used by other service requester enterprises. Here, it is important to clarify the difference between two main kinds of services:

1. **Basic Services (BASs)**, represent the basic "atomic" end-point services offered by tourism organizations. Examples of services that fall in the BAS category include: hotel_booking, flight_booking, telechecking, car_rental, etc.

2. **Value-Added Services (VASs)**, are composed of a set of basic services and/or other value-added services that are interconnected together to offer new higher-level tourism services. For example, a holiday package service is composed of a number of basic services, such as arrangement of flights, accommodations,
museum visits, local tours, etc. The event attendance, group tours and general information provision services in Figure 5.1 are also examples of VASs. Please notice that the definition and execution of a value-added service itself involves several activities, what in fact leads to the definition and execution of a business process (BP), i.e. a VAS is always implemented as a BP.

In order to plug/unplug and use BASs and VASs, the actual tourism services, which can be already implemented in heterogeneous hardware/software platform, must be extended with a common “interoperability layer” to become part of the general federation of available services. Following a typical object request broker architecture, this interoperable layer can be defined as composed of two main parts: the service wrapper and the service proxy (see Figure 5.2). Namely, the actual service implementation can be extended with a wrapper that provides a common interface to it, and processes the requests coming from remote proxies. A proxy represents a pointer or a remote reference to a wrapper, and acts as a front-end for other systems or applications that ultimately need to access the tourism service implementation. In this way, 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), via the locally developed service wrappers.

Also, there are some important considerations that must be taken into account in the design and implementation of the common service federation layer above:

- The common layer must use a common platform-independent interface definition language. This is necessary in order to define the wrappers for the heterogeneous systems and platforms used in tourism enterprises located worldwide.

- 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

![Figure 5.2: Interoperability approach for common access to tourism services.](image)
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 Definitions Catalogue, which could be accessed via Web interfaces. Similarly, the data type structures needed by the service interfaces, could be made available through an Object Interface Definitions Catalogue. The service and object definitions catalogues will be further described later in this chapter.

- A service proxy can be submitted by a service provider to be registered in service directory nodes, from where other service requesters can search and look up proxies that comply with the interfaces and specifications defined in the Service Interface Definitions Catalogue, and that are suitable for their purpose. The service requester can have access to any of these directory nodes, issue queries on the directory information, retrieve a set of proxies that comply with specified conditions or parameters, and finally, select and fetch the desired BAS and/or VAS proxies to be used with its own internal applications or value-added services.

- During the process of registration of service proxies in the directory nodes, the submitter enterprise (service provider) must be able to specify certain access rights on its own service wrappers for other enterprises. For instance, some services may be offered to public while the use of other special tourism services could be restricted to certain enterprises or end users depending on criteria such as geographical location, user or enterprise profiles, mutual agreements and contracts, etc. This issue will be analyzed later in this chapter.

The service proxies must be able to seamlessly interoperate with other proxies in order to support the combination of simple services (BASs) into more complex value-added services (VASs). Namely, some proxies can be embedded in the implementation of other proxies, and the input/output parameters of a given service must be properly interpreted by other services or internal systems that intend to use it.

In terms of the computational technology to be used to implement the architecture supporting such a service federation, several features of the Java language can be well exploited, including platform portability, object orientation, and wide industry acceptance [27]. Also, Jini connection technology can be conveniently applied as the base approach in order to facilitate the construction and deployment of the components of the interoperable service layer described above [8, 59, 64]. The Jini architecture has been specifically designed for deploying and using generic services in a distributed network [24]. These 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. The system architecture analyzed in this chapter is actually based on Java/Jini technologies, and its main components are described in the next section.
A Support Infrastructure for Tourism VEs

In order to develop a VE infrastructure in the tourism sector addressing the considerations introduced in the previous section, a common system "structure" is introduced here as a reference model. The diagram presented in Figure 5.3 depicts the main structural components of the proposed system. Each component of the figure is briefly described in the paragraphs below.

Let us first identify the main kinds of end users described in Figure 5.3:

- **Service provider.** The service provider represents a tourism enterprise that offers a service or resource to the global federation. At the service provider site, the actual tourism service implementation is extended by a service wrapper and a service proxy, which represent the Java/Jini-based code that needs to be developed in order to plug the actual service to the global federation.

- **End user.** The end user in the figure represents a person who may interact directly with different system components in order to for instance, browse public catalogue information and eventually request a service.

- **Administrator Users.** These are authorized system administrators with special access rights, which are able to modify the data objects or internal structures of the system architectural components.
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 mainly four components:

- The Service/Object Interface Definitions Catalogue Managers (SERV-CAT/OBT-CAT). The objective of the SERV-CAT and OBT-CAT tools, is to allow service requesters and providers to browse the precise service and object definitions for the tourism services that are handled within the federation. These tools will be described in details in Section 5.4 of this document.

- The VAS Process Management System (VAS/BP Manager). This system allows the definition and execution of business processes representing value-added services in the context of Virtual Enterprises. Within the context of the 5FP FETISH project, this component is actually being designed and developed by the UNINOVA research center, in Lisbon, Portugal, and is called PROMAN. Therefore, in the remaining of this chapter we refer to this toolkit component as PROMAN. For example, PROMAN allows the definition of “generic conference plans” described in Section 5.2.1, based on a combination of the service definitions that are accessed via the SERV-CAT tool. For a specific instance of that generic plan, the end user will be able to select specific service proxies that would match the service definitions specified in the generic plan. This scenario will also be described later in details in Section 5.4 of this chapter.

- 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. In this case, ACMS can present an integrated user interface for certain offered capabilities of the system, such as “quality control and service operation maintenance” requests by users.

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. As will be proposed in Section 5.4, the DSLM component can become an extension to the DIMS introduced in this thesis.

- **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 set* of service requesters or only to specific VE members. Such access rights provision is extremely important for support of the VE infrastructures. Therefore, the FARM module must support the proper configuration of the access rights. Furthermore, it must work in close cooperation with the SLM (at each DSLM node), in order to accomplish the proper filtering and validation of access rights *at the moment* in which the lookup process is being performed.

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 definitions 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. If the new service interface is accepted, then the next step is to create the corresponding knowledge base entries in the tourism ontology, and then these definitions must be entered into the service and object interface definitions catalogues. Only afterwards, the new information can be accessed by end-users and service providers through the SERV-CAT and OBT-CAT tools.
Based on the system structure defined in Figure 5.3, 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, and user access to an existing service. An example of the application of the distributed service look-up facility will be given in Section 5.4 of this Chapter. Furthermore, the system structure can also support more complicated cases, such as the definition, registration, and access to value-added services (also using the PROMAN module) within the global federation.

The definition of the proposed system structure introduced in this section, is necessary in order to fully understand the analysis and design phases for the extended DIMS presented in the next sections of this chapter.

5.3 Federated Information Management Requirements for Tourism VEs

The distributed information management approach to support VEs in the tourism sector shares many common characteristics with the approach in other VE areas, such as the industrial manufacturing. For instance, the fact that enterprises involved in different aspects of tourism need to: share/exchange information spread over geographically distributed areas; co-operate in order to perform their specific tasks within the general common goal of the VE; and act in a coordinated manner to guarantee a consistent global behavior. Furthermore, enterprises involved in tourism businesses typically employ heterogeneous legacy systems as their internal information systems, which use disparate data models and hardware/software platforms.

Therefore, any mechanism addressing the management of distributed information among enterprises in tourism must consider the sharing/exchange of distributed information in a highly heterogeneous environment. At the same time, it must be taken into account that these enterprises represent autonomous nodes. Namely, these enterprises run independently of each other, and they may evolve independently in terms of their applications and data models. Moreover, the level of autonomy in these enterprises implies that when accessing new distributed services, each enterprise must be able to independently decide to share a part of their local resources or services with certain other specific companies, based on their bilateral or VE-related agreements.

In this context, the support for protected and selective access to distributed tourism enterprise services and resources must be provided by the corresponding information management system of the VE support platform. For instance, when an enterprise joins the service federation, the enterprise may decide to make some of its internal services available (or not) to other enterprises based on available enterprise profile information. In particular, the use of a specific service provided by a given company may not be valid for other companies or users outside a determined geographical area, such as a particular country or even the European union. In other words, the legislation of a given country in relation to others may contribute to the validation of this kind of visibility levels in order to either grant or deny access to a given service or set of services.

In effect, it is clear that the need to support access rights and visibility levels
on internal tourism services and resources at every company must be taken carefully into account by the information management system of the VE support platform, since not every company or end-users can have the same kind of access to services or resources that are internally managed by a given company.

Please notice that this sort of *visibility levels and access rights* defined on distributed tourism “services” and resources, can be well represented, supported and reinforced by a federated information management system architecture, such as the DIMS system architecture introduced in Chapter 4 of this thesis. In fact, the design and implementation of the FARM module introduced in the system structure presented in Section 5.1, is fully based on the Export Schema Manager (ESM) and associated end-user tool (ESMT) that were described in Chapter 4.

Therefore, the next subsections describe the particular functional and information modeling requirements that have been identified for the DIMS in the tourism sector.

### 5.3.1 Functional Information Management Requirements

In this section, the analysis of the “functional requirements for the DIMS” is presented in order to support tourism VEs within the general context described in Section 5.2. Please notice that the analysis of DIMS information management requirements for tourism VEs, is also based on the requirement analysis results described in previous chapters of this thesis. Bearing this consideration in mind, the information management requirements for VEs in the tourism application domain can be divided in three categories:

- **“Generic”** fundamental information management support for VEs. This category includes generic functionalities for VE instantiation and operation support such as: VE topology initialization, management of information related to VE contract clauses, status monitoring of VE distributed business processes, and definition and maintenance of access rights for each particular VE partner at every VE node. These functionalities were already identified for the DIMS as described in Chapters 3 and 4 within the context of the industrial manufacturing VEs, and are generic for VE information management support. In particular, they need to be applied to tourism VEs, taking into account that certain specific functionalities still need to be developed and adapted to meet the particular requirements of tourism enterprises.

- **“Specific”** tourism VE requirements. There are certain VE-related information management functionalities that are specific to the tourism domain. As examples of support functionalities in this category we can mention: catalogues for tourism services and service providers; support for tourism value-added services; and incorporation and adaptation of existing minimum data sets in the tourism sector that need to be recognized by the tourism VE infrastructure.

- **Advanced VE features and mechanisms.** Some of the identified VE information management requirements in the tourism sector represent extended functionalities that do not fall under the “fundamental” category and that are not specific
to tourism sector. Examples here include: automatic definition of information access rights for VE partners based on VE contracts and clauses, as proposed in Chapter 4; extended Internet-based access to VE federated database schema; and information management support mechanisms for VE dissolution.

Consequently, the information management requirements described in the next sub-sections do not exclusively belong to the tourism-specific category. Rather, they also include requirements that have been identified based on previous development experiences such as PRODNET, as well as potential advanced extensions associated with the VE information management system.

Some DIMS requirements for the first two categories addressed above are analyzed in more details in the next sections below. The DIMS requirements related to the support of other advanced VE features and mechanisms will be addressed in Chapter 6 of this thesis.

“Generic” fundamental information management support for VEs

In order to support a VE environment for tourism enterprises as described in Section 5.2.2, the following “generic” DIMS information management requirements can be identified, mostly to support the VE operation, but also some of them can apply to other specific cases:

1. Management of information related to tourism company profiles (enterprise catalogue). An enterprise catalogue containing profile (descriptive) information about potential and real VE partner companies is an important data repository that has different applications during the entire life cycle of the VE (e.g. creation, initialization, operation and dissolution phases [42]). This catalogue can be used to for instance:

   - Search for best-fit partners for the creation of specific ventures.
   - Keep track and provide the service quality information, related to the performance of the services offered by an enterprise.
   - Provide general profile information about the enterprises (name, address, objectives, focus areas, location, list of supported services, etc.).
   - Store communication parameters (e.g. network address, contact personnel’s email addresses, public encryption keys, ftp server addresses, etc.) for each given company so that others can communicate with it.

2. Data management functionalities to support the creation, initialization, and operation life-cycle phases of a VE. The information regarding the VE topology, namely the VE composition in terms of partners and relationships among the VE partners, needs to be stored and managed in the DIMS in order to support the operation of the VE itself, and of other functionalities such as the Distributed Business Process (DBP) monitoring.
3. **Information management support for VE agreements/supervision clauses.**
For the VE to operate properly, there must be some sort of agreement in terms of for instance, the contracts among its partners and the VE supervision clauses for its coordinator, in which the responsibilities of each partner providing a service to another partner are clearly stated, made available for access at any point of the VE execution phase, and enabled to be monitored. These agreements are the on-line counterparts of the other physical legal contract or arrangement that could be settled and signed by the involved companies outside the context of the VE support infrastructure. The VE agreement information must be modeled and accessed as on-line data in the database. The agreement information will be used to both define the proper access rights regarding the exchange of information among every two partners in a VE, and to provide a base for monitoring of task progress status.

4. **Generic federated database management facilities to support the secure data access/exchange among autonomous VE enterprises.** These facilities include the application of the generic Federated Query Processing (FQP) and the Export Schema Manager (ESM) modules of DIMS described in Chapter 4 (see for example [74, 65]). There could be many scenario cases in which the support for secure access to distributed/shared information among tourism VE enterprises is required. For example, let us consider the case where VE enterprises need to exchange information related to for example, commonly used Minimum Data Sets (MDS) [91]. In the most general case, enterprises will need to exchange and share information through any ad-hoc queries about the MDS. For this point, a federated database system based on FQP and ESM components would support this scenario by providing seamless access to distributed information at different sources such as tourism agencies. At the same time, specific access rights bilaterally agreed and defined among companies at different levels of a VE are preserved and enforced.

Please also notice that this list of generic requirements has been identified based on the requirement analysis phase described in Chapter 3, and partially based on the studies performed within the 5FP FETISH project.

**"Specific" tourism VE requirements**

This section focuses on some DIMS requirements that are specific to the tourism VE application domain, including: support for selective access to enterprise services, management of service interface definitions catalogue, management of tourism distributed business processes, and some general requirements considering the system platform described in Section 5.2.

**Selective access to enterprise services.** As described previously in Section 5.3, when an enterprise joins a VE, it must indicate which services it will make available to which other specific enterprises, based on for instance, the enterprises profile information or special agreements. For example, a given service of a company may not
be offered to companies outside the European Union. This sort of visibility levels and access rights definitions regarding enterprise “services”, must be represented and managed by the DIMS through the FARM component (see Figure 5.3). Please notice that these access rights need to be supported both within the VE, in order to provide the access rights among the partners, but as well for other enterprises that are not presently involved in a VE. To support the previous point, the proper access rights need to be stored, as well as a **catalogue of services** (proxies) being offered by every service provider in the network. Further information about these proxies can be associated to this catalogue in order to be used to keep track of enterprise’s service execution performance, service quality assurance issues, etc.

**DIMS requirements for Service Interface Definitions Catalogue.** An important requirement for the DIMS is the need for the design and development of a “Catalogue Manager for Services’ Definitions” for the tourism VE infrastructure, as described in Section 5.2.1. Typically, when an enterprise in the network wants to either access an existing service or to provide a new service proxy, that member needs to know, among many other issues:

- What are the interfaces being offered for this service?
- What is the precise functionality required by a specific service interface?
- What are the parameters (and datatypes) associated with a service?

This information must be handled independently of both the existing service proxies and the service implementations offered by other VE member enterprises. In other words, it is necessary to maintain this **Service Interface Definitions Catalogue** as a data repository containing all the required **static** information about the “service-interfaces”. As such, this catalogue needs to maintain the different formal descriptions of the federation services. The end-users of the VE infrastructure should be able to browse this information via for instance, a Web interface. This catalogue may also need to be accessed by other internal components of the VE infrastructure to perform some validation actions, that may be necessary for instance, when a new entry is added in a service directory node.

The Service Interface Definitions Catalogue must also contain the precise definition of the parameters for the listed service interfaces. Namely, the data structures being used for input/output parameters of services need to be described and documented. The parameters will typically correspond to data subsets of different standards, or commonly accepted data definitions handled in the tourism sector.

Please notice that in principle, it is not the aim of the tourism VE member enterprises to define and propose new data set definitions but rather to support existing ones. In practice, there exist many tourism standards or initiatives that can be considered, including [93]: ETC (European Travel Commission), HEDNA (Hotel Distribution Electronic Network Association), OTA (Open Travel Alliance), TTI (Travel Technology Initiative), HITIS (Hospitality Industry Technology Integration Standards), IATA (International Air Transport Association), UN/EDIFACT TT&L
Federated Information Management Requirements for Tourism VEAs

(Travel Tourism & Leisure), WATA (World Association of Travel Agencies), and IFITT (International Federation for Information Technology and Tourism).

Furthermore, it may be necessary to manage information about different versions or extensions of service interfaces and data types. For instance, it may be useful to handle information about the correspondence between different versions of data set items that may be defined in the future as the data standards evolve.

Security aspects related to the permissions for special users/administrators to have read/write access to different parts of the catalogue also need to be carefully analyzed.

For the representation and management of the Object Definitions Catalogue information (i.e., the catalogue describing the specification of the parameters and return types of the interfaces stored in the Service Interface Definitions Catalogue), the use of XML and Java type definitions definitely plays an important role. Here we should consider that the service requests among VE enterprises can be done by means of Java/Jini functions or by means of the exchange of XML messages. This latter possibility has the advantage that some end-point users and service requester enterprises would not necessarily need to rely on Java/Jini technology if they can process XML documents representing the service request and its corresponding parameters. Therefore, it is possible to define an XML representation of a Java object whose structure is defined in the Object Interface Definitions Catalogue. In this case, the VE infrastructure can provide the parsers that convert XML documents to Java objects (and vice versa), where both documents and objects comply with the required data structure definitions (e.g., Java classes and probably XML schemas) that are stored and managed by the Object Catalogue associated with the Service Interface Definitions Catalogue. For this task, the application of XML Data Binding technology for the Java platform can be applied, as will be explained in Section 5.4.3.

The Service Interface Definitions Catalogue information may not only be restricted strictly to the services and parameter descriptions. In general many other kinds of information could be associated to this catalogue. For instance, information about companies providing implementations for a given service definition, or the quality information related to the provision of services by those companies, may also be related with this catalogue.

Besides the Service Interface Definitions Catalogue described in this point, a Service Catalogue representing and describing the actual service proxies (and their behavior) provided by different companies, is also considered as a requirement for DIMS. Based on this catalogue, it is also necessary to support the selective access to enterprise services as described previously in Section 5.3.1.

Support of Business Process Management. The following information management requirements have been identified for the support of the distributed business processes used for modeling value-added services as explained in Section 5.2. Some of these functionalities could be carried out in the context of VEs while some others could be conceived outside the VE operation:

1. Support for data access/storage of templates that precisely define the VE-related VASs. The VAS definitions used by a tourism company can be expressed
in terms of a business process model representing the sequence and dependencies among different sub-processes. For certain business processes used on regular basis, a kind of “template” can be defined. The objective of these templates is to facilitate the creation and instantiation of commonly used business processes, inside a given company and within the network of tourism enterprises. These templates can be stored and managed by the DIMS database for further usage and eventual exchange of templates with other enterprises.

2. *Data management functionality to support the progress status monitoring of active DBPs.* In the tourism application domain, activities in a running instance of a given distributed business process take different amounts of time to complete, ranging from a few seconds to several weeks [8]. For instance, a ticket reservation can done several months ahead, but the final insurance of the ticket for a package may need to be done four weeks before the trip. Considering the many simultaneous business processes that run at enterprises, and the amount of data that each business process requires to handle, in order to properly execute and support its progress monitoring, it is therefore convenient to store such associated data about the status of the DBPs in the DIMS. In this way, the information can always be dynamically stored, modified, and retrieved at any time for different purposes. This approach has many clear advantages to the storage of this data in the main memory or simple file systems.

**Some general system design considerations.** There are several requirements to be considered regarding the adaptations and extensions of the DIMS architecture, in relation to the general system structure presented in Section 5.2. For instance:

1. *The facilities and advanced features of the Java/Jini application development environments must be used and exploited as much as possible.* For example, the facilities for database management provided by Java should be considered within the DIMS implementation, as well as the Jini distributed services facilities (whenever suitable) available within the system infrastructure.

2. *The DIMS architecture must be compatible with the general architecture designed for the tourism VE support infrastructure.* Namely, the design and implementation of the advanced VE coordination mechanisms and the required federated database management services must follow the common “philosophy” defined in this case for the global service federation architecture, and use the provided infrastructure whenever possible and convenient.

### 5.3.2 Requirement Analysis for Information Modeling

Considering the functional requirement analysis described in the previous section, several pieces of information, concepts and/or entities are identified that also need to be modeled and stored in the DIMS system. Namely, a detailed study of the application domain needs to be performed in order to define the specific database schema that will ultimately represent and encompass each relevant concept or entity.
In this section, a summarized and simplified list of such data entities is provided, as an example of the information modeling requirements that need to be analyzed in DIMS. Some of these entities will be further detailed in Section 5.4.

The identified entities can be initially categorized as follows:

- **VE kernel specific concepts.** This category includes general VE information (VE id, VE name, number of partners), the VE Partners (e.g. partner id, name), VE agreements or contracts, and general VE DBPs.

- **General information catalogues.** This point includes for instance enterprise profiles (service providers and service requesters profiles), Service/Object Interface Definitions Catalogues, and the catalogue of enterprise services proxies.

- **DBP-related information, e.g.** DBP model for DBP templates, and DBP “instance” model to support the monitoring of long-term running DBP instances.

In the following sections, these information categories are described in more details.

**VE Kernel Information**

An extensive analysis and design of the detailed information management models that represent the VE concept has been described in Chapters 3 and 4 of this thesis (see also [12, 11, 71]). Please notice that most of these models can be used as a reference for designing the corresponding VE-related information management functionality in the tourism area.

A basic set of attributes describing the VE and VE partner entities in the tourism sector, is included next based on the DIMS integrated schema described in Chapter 4:

- **For VE-related information:** VE identifier, VE description, number of partners, VE Coordinator identifier, VE contract number.

- **For VE partner information:** Enterprise id, VE id, name, description, business area, regular address, fax, telephone, Web homepage, contact persons, Web server address, IP address, hostname, security certificate number, public encryption key, pop user account, password, smtp address, etc.

**General Information Catalogues**

As mentioned before in this section, the catalogues that need to be modeled and represented in DIMS include: enterprise profiles, catalogue of service interface definitions, data set (object) definitions catalogue, and the catalogue of service proxies offered by enterprises. The complete description of the information modeling requirements associated with these catalogues is outside the scope and the purpose of this chapter. However, in Section 5.4 of this chapter, the details of the database design for the Service Interface Definitions Catalogue will be described.
DBP related information

As mentioned before in Section 5.2, value-added services in tourism can be defined as DBPs as depicted in Figure 5.4, for a “Booking a Journey” example [8].

Furthermore, a DBP, can be represented by a workflow plan, as defined by the Workflow Management Coalition - WfMC [167]. Therefore, the DBP templates can be defined by means of a workflow editor tool, and the corresponding DBP instances will be executed by a workflow management engine module. This engine is responsible for the interpretation of the process definition; control of process instances (creation, activation, suspension, termination, etc.); maintenance of workflow control data and workflow relevant data; and passing workflow relevant data to/from applications or users, among other functionalities.

The DIMS can model and manage the information which is necessary to support the proper operation of the workflow engine [9]. As suggested by the WfMC, an initial meta-model for this information is shown in Figure 5.5. This meta-model can be further refined and tailored according to the specific requirements of the VE support platform and it can be stored and managed by the DIMS.

Finally, in [53] an example of a workflow definition to support a trip reservation scenario is provided. However, there the workflow plans are specified in terms of a first order predicate logic formalism instead of using the WfMC model.

![Diagram showing a workflow plan for the BP template representing a VAS.](image-url)
5.4 Extension and Application of DIMS to Support Tourism VEs

This section focuses on the necessary extensions and modifications to the internal design and implementation of the DIMS presented in Chapter 4, in order to address the information management requirements for VE support identified in Section 5.3. In particular, Section 5.4.1 describes the proposed extensions to the architecture of the DIMS and its internal components. Furthermore, Section 5.4.2 describes the detailed design and implementation of the Service Interface Definitions Catalogue, as one of the main functional extensions to DIMS in order to address some of the peculiarities of the tourism VE in relation to industrial manufacturing VEs. Finally, Section 5.4.3 analyzes in details some of the development standards and tools that have been used in the adaptation of the DIMS for the support of tourism VEs.

5.4.1 Extending the DIMS Architecture

In order to support the functional requirements identified in Section 5.3, the DIMS internal design presented in Chapter 4 can be extended and modified as illustrated in Figure 5.6. As can be seen in this figure, the general DIMS architecture is maintained in this design, although some of the original components need to be modified according to the specificities of the tourism domain, and some other components need to be introduced to cope with new information management functionalities.
Please notice that Figure 5.6 is also consistent with the general VE support infrastructure presented in Figure 5.3. Namely, the extended DIMS architecture encapsulates and ties together the information management functionalities represented by several components in the reference system structure introduced in Section 5.2. The DIMS architecture incorporates these functionalities in such a way that the corresponding modules can be systematically and incrementally developed during the implementation phase. In other words, all the information management requirements identified and described earlier in this chapter, can be addressed by different modules along the DIMS tiers in a comprehensive manner.

The major components that need to be substantially modified or added in relation to the original DIMS architecture are described next. For a detailed description of the other DIMS components, please refer to Chapter 4 of this thesis.

### Extended DIMS Server Agent

The extended Server Agent in Figure 5.6 is composed in turn of a set of specific Application Servers (ASs) and DIMS client tools/APIs as described below.

Each extended DIMS Application Server is designed according to a particular functionality that is needed to be supported for the tourism VE infrastructure, as described in Section 5.3. For instance, the following application servers need to be developed as part of this DIMS tier:
5.4. Extension and Application of DIMS to Support Tourism VEs

- SERV-CAT AS: handles the application logic for the client tier components that have access to the Service Interface Definitions Catalogue.

- DBP AS: this server supports the DIMS functionality that is required in order to support the PROMAN operation.

- FARM AS: this AS handles the application logic for the corresponding clients, and manages the definition of access rights defined on the catalogue of proxies at a given node. The functionality of this FARM is supported by the ESM component of DIMS.

Each of these application servers can interact with other DIMS internal components in order to request a generic data management service, such as the evaluation of a federated query through the FQP component, or the storage of local information through the DIMS Internal Database Manager.

Furthermore, the extended DIMS client tools and APIs are represented by the Java components that directly interact with either the end users or with other internal VE infrastructure components. From a general point of view, these extended DIMS client components consist of:

1. End-user interfaces. These interfaces include in turn:
   - End-user browser applications, such as the:
     - Service Interface Definitions Catalogue Browser (SERV-BROWSER): application or applet that is used by service providers and end-users to browse the content of the Service Interface Definitions Catalogue.
     - FARM Editor (FARM-EDIT): for edition of access rights on service proxies, so that each enterprise can determine which other enterprises will have access to each specific service that has been plugged into the global federation.
   - End-user Administration Tools, such as the:
     - Service Interface Definitions Catalogue Administrator (SERV-ADMIN): application or applet that supports the maintenance of the information stored in the Service Interface Definitions Catalogue.

2. Java Interfaces. These interfaces represent libraries or packages of high-level information management functions specifically developed to support other system architecture components such as PROMAN. These interfaces basically include:
   - Java applets (DIMS-APP),
   - and Java packages (DIMS-PACK).

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 with the 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.
Extended DIMS "Service-oriented" FQP

As mentioned in Section 5.3, the support for protected and selective access to distributed tourism enterprise services and resources is one of the main requirements that must be addressed by the corresponding information management system of the tourism VE support platform.

Consequently, in the tourism application domain, the functionality of Federated Query Processing described in Chapter 4 can be extended and modified in order to not only consider data as the subject that is being queried and manipulated, but also service proxies. In other words, considering the tourism-related scenario described in Section 5.2, end users and applications interacting with the DIMS need to retrieve "service proxies" that comply with a given service interface definition. To support this functionality, it can be assumed that the DIMS FQP component is extended with embedded Distributed Service Look-Up Manager (DSLM) capabilities that would allow the retrieval of service proxy objects, which can be delivered to end users and would ultimately execute the actual tourism service. More details about the internal operation of the DSLM component of FQP are given next.

Please remember that the DSLM facilities can be built on top of the local service look-up management mechanisms already provided by Jini. Namely, the DSLM must be able to receive a look-up service request specifying the retrieval of a set of proxies complying with specific conditions, and to evaluate it locally using the Jini-based SLM, as described in Section 5.2.2.

Furthermore, considering the fact that the federation of DSLM nodes is organized according to a given global structure such as a hierarchical tree or an acyclic graph of nodes, the initially contacted DSLM must propagate the look-up service request to other succeeding DSLM nodes, which would recursively process the request in the same way, and finally provide the corresponding results. For instance, considering the case in which the DSLM nodes are organized following a tree-like structure, the first DSLM node receiving the original look-up service request, must propagate this request to the parent and children nodes, get the result from their corresponding distributed look-up service invocations, merge the results obtained from these nodes with the results obtained locally, and provide the final list of proxies to the requesting end-user or application.

Furthermore, the DSLM component interacts with the ESM module of each node in order to validate the access rights that a given tourism enterprise has defined on its own service proxies in relation to other enterprises in the context of existing VE collaborations. These access rights must be defined through the FARM-EDIT application at the moment in which the proxy is registered at a given DSLM node, and can be modified later if required.

An example of the described scenario involving the ESM and FQP components of DIMS is depicted in Figure 5.7. First, let us suppose that a user or service provider is using the SERV-CAT browser of DIMS node A, and decides that he/she wants to search for implementations of a service for the HotelSearch Interface (H). The SERV-CAT client application will seamlessly communicate with the corresponding application server of DIMS A, using for instance RMI technology. Then, the SERV-
CAT application server will in turn contact the FQP/DSL M component of DIMS. Here, the DSL M will check if the local SLM contains some proxies for the desired interface. Let us suppose that the SLM discovers a list of proxies \{H1, H2, H3\} complying with the HotelSearch interface H, and passes the list of proxies identifiers, together with the identifier of the service requester (and eventually a VE identifier), to its local ESM component. The local ESM component checks the proxies identifiers against its internal access rights definitions for the requester in the specified VE and returns to SL M a verified list of proxies identifiers that can be accessed from the respective service providers. Let us also assume that the locally verified list is \{H1, H3\}. At this point, the local processing of the look-up service request has been completed. Furthermore, using the service directory information the DSL M at node A knows that one or more proxies can be found if the search is propagated to its parent, DSL M node B, and to its child, DSL M node C. Thus, at the same time, it propagates the search to these nodes. The child and parent DSL M nodes carry out the same process recursively and return the lists of proxies \{H4\} and \{H9, H10\}, respectively. At the end, the partial lists of proxies are merged and the result \{H1, H3, H4, H9, H10\} is returned by the DIMS to the original service requester through the SERV-CAT client. The user or service provider may ultimately decide to use one of these proxies individually, or to make it part of a VAS.

Please notice that according to the described scenario, the sub-query decomposition step of the FQP described in Chapter 4, is actually extended and enhanced by new distributed service look-up mechanisms at each node. Namely, the initial distributed look-up service request is propagated to other nodes in a similar way in which typical FQP "sub-queries" are decomposed and sent to the other involved

---

**Figure 5.7:** Example evaluation of DIMS distributed look-up service request.
nodes. Similarly, the results of the propagated requests must be merged and returned to the requesting end user or application. Also, the access rights defined on each proxy by its owner enterprise are always validated and reinforced by the FQP mechanism via the ESM components of each node. In this way, the requirement of selective and secure access to distributed enterprise services described in Section 5.3.1 is properly addressed by the DIMS. Even more, through the extended FQP component presented in this section, the DIMS is able to support the federation of "services" as well as the federation of static data.

**DIMS Internal Database Manager**

As mentioned in Chapter 4, the DIMS system actually acts as a client of an internal database server. This back-end server is represented by the Oracle DBMS. In the case of reference infrastructure described in Section 5.2, the DBMS server also needs to be extended to store and manage the local part of the tourism Federation Repository information (see Figure 5.8), and to provide the associated data access mechanisms, including OCI, ODBC, JDBC, PL/SQL, necessary database stored procedures, etc. Please notice that some of the functionalities developed at the server tier, such as database stored procedures and triggers, can be conveniently reused by different applications servers in the DIMS Server Agent tier. For example, a given database stored procedure to retrieve information about the service definitions catalogue, may be reused in the implementation of the SERV-CAT and DBP application servers.

According to certain access scenarios for the Repository information, it is convenient to allow that other infrastructure components e.g. PROMAN, can access the database via standards such as JDBC if necessary. In this way, other architectural components can count on a flexible, reliable and generic mechanism for data manipulation. The basic database access rights and security issues for each components can still be supported by the JDBC connection, preventing for instance, non-authorized modules from modifying or deleting service catalogue information, or changing the database schema. By using JDBC, other modules can also develop their own functionalities without necessarily waiting or depending on specific DIMS developments. However, if requested and if deemed convenient, some of these functionalities can be encapsulated and moved to the DIMS middle tier in order to provide a higher level

![Figure 5.8: Part of the information stored in the DIMS internal DBMS.](image)
of abstraction support for the DIMS client components.

In this section, the general architecture of the extended DIMS was presented. In the next section, the design and implementation of the “Service Interface Definitions Catalogue” component is presented in more details.

### 5.4.2 Design of Service Interface Definitions Catalogue

The design of the Service Interface Definitions Catalogue component follows the general three-tier model considering the DIMS architecture described in Section 5.4.1. Namely, the SERV-CAT repository database, application server, and client tools have been developed following this model and are described in the next subsections.

Please notice that the SERV-CAT addresses one of the “specific” tourism VE requirements described in Section 5.3.1. Namely, the SERV-CAT design and implementation represents an important extension to the DIMS design presented in Chapter 4. For these reasons, this section introduces several new data structures and objects that were not described in Chapters 3 and 4.

#### SERV-CAT Repository Database

In order to develop the SERV-CAT tools, a detailed definition of the data structures (e.g. a formal data model) representing the service interfaces and its corresponding data types, must be achieved. The proper data structures that are necessary for the Service Interface Definitions Catalogue were analyzed and defined by the Entity Relationship (ER) diagram showed in Figure 5.9. A detailed description of this diagram is outside the aim of this chapter; 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 FETISH project, which has provided an environment to test and validate some of the DIMS extensions described in this section. The entity TFET_FET_TYPES is a generalization of the concepts of Java classes (TFET.CLASSES) and interfaces (TFET.INTERFACES). Both classes and interfaces can be defined recursively in terms of other types, and that they are encapsulated in packages (TFET.PACKAGES). Furthermore, classes and interfaces are associated with class members (TFET_MEMBER), which in turn can be either methods (TFET_METHOD) or fields (TFET_FIELD). 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 (these parameters provide the link between the Service Interface Definitions Catalogue and the Object Interface Definitions Catalogue as will be explained later in this chapter).

Based on the described ER diagram, the corresponding database definitions can be generated to create an Oracle database instance within DIMS. However, these definitions cannot be directly used by browser or management applications that need
Figure 5.9: Entity-Relationship diagram for service interface catalogue definitions.

Figure 5.10: UML class diagram for Service Interface Definitions Catalogue information.
act with DIMS will use this package in order to exchange service interface catalogue information in a common format as Java objects.

The package is called *FetishTypes*, and a snapshot of the documentation generated for this Java package is presented in Figure 5.11, including some of the main classes that are defined in the package. As can be seen in this figure, the Java classes directly correspond to the UML model classes.

**Application Server – SERV-CAT AS**

Following the extended architecture of the DIMS, there are several application server components that need to be developed in order to provide high-level information management functionalities to the client tier applications, and hiding for instance, low-level JDBC and database access details. In the case of the Service Interface Definitions Catalogue, the application server (SERV-CAT AS) provides functions for manipulating catalogue data using the FETISH Types package described in the previous section. These functions provided by the application server include for instance: get packages, get type information (for a given package identifier), get method information (for a given class identifier), etc. A prototype of the SERV-CAT AS providing these functions has been developed, and a part of the HTML documentation generated for this server is included in Table 5.1.

**SERV-CAT Browser/Administrator Client Applications**

Regarding the DIMS client tier, this section briefly describes the developed prototypes for the browser and manager client applications of the Service Interface Definitions Catalogue, i.e. the SERV-BROWSER and SERV-ADMIN end-user applications introduced in Figure 5.6. The SERV-BROWSER client application can be used for instance

![Figure 5.11: Description of FetishTypes Java package.](image-url)
by service providers, and the manager application will be used by the catalogue administrator according to the general system structure described in Section 5.2.2. A snapshot of the browsing interface is shown in Figure 5.12. 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 5.13).

Please notice that for the snapshots of the interfaces in this document, only a few example data is shown for the sake of simplicity in illustrations of the usage of the interfaces. In reality, the data that is actually loaded into the database, will be generated with the help of the expert organizations in tourism.

Furthermore, through the administration tool (SERV-ADMIN), its end user i.e. the catalogue administrator, is able to insert, delete, and modify any data item in the catalogue information. These capabilities are not included in the normal browsing client applications that the service providers use.

The client application described in Figure 5.12 and Figure 5.13 was developed entirely based on the functionality offered by the application server described in Section 5.4.2. This means that the browser application only needs to access the high-level functionality of the application server without being concerned about the specific database structures and JDBC access mechanisms.

<table>
<thead>
<tr>
<th>Service Catalogue Application Server Functionality</th>
</tr>
</thead>
<tbody>
<tr>
<td><strong>void deleteClass(java.lang.String classld)</strong></td>
</tr>
<tr>
<td>Deletes a Class from the service catalogue.</td>
</tr>
<tr>
<td><strong>void deleteMethod(java.lang.String methodld)</strong></td>
</tr>
<tr>
<td>Deletes a Method from the service catalogue.</td>
</tr>
<tr>
<td><strong>void getClass(java.lang.String classld)</strong></td>
</tr>
<tr>
<td>Retrieves the Class with the specified ID from the service catalogue.</td>
</tr>
<tr>
<td><strong>ArrayList getClassMethods(java.lang.String classld)</strong></td>
</tr>
<tr>
<td>Gets all the methods for a given class ID.</td>
</tr>
<tr>
<td><strong>ArrayList getMethod(java.lang.String methodld)</strong></td>
</tr>
<tr>
<td>Gets the method with the specified method ID.</td>
</tr>
<tr>
<td><strong>void getPackage(java.lang.String packageld)</strong></td>
</tr>
<tr>
<td>Retrieves the Package with the specified ID from the service catalogue.</td>
</tr>
<tr>
<td><strong>DefaultListModel getPackageTypes(int packageld)</strong></td>
</tr>
<tr>
<td>Gets the types (classes, interfaces) for a given package ID.</td>
</tr>
<tr>
<td><strong>DefaultListModel getTypeMembers(int typeld)</strong></td>
</tr>
<tr>
<td>Gets the members (fields, methods) for a given type ID.</td>
</tr>
<tr>
<td><strong>void insertClass(TFet_Classes aClass)</strong></td>
</tr>
<tr>
<td>Inserts a new Class into the service catalogue.</td>
</tr>
<tr>
<td><strong>void insertMethod(TFet_Method_Member aMethod)</strong></td>
</tr>
<tr>
<td>Inserts a new Method into the service catalogue for the specified Class.</td>
</tr>
</tbody>
</table>

Table 5.1: Sample function definitions for SERV-CAT Application Server.
Integration of Service and Object Interface Definitions Catalogues

As mentioned in Section 5.3.1, the data structures for the Object Interface Definitions Catalogue take the form of XML Data Type Definitions (DTDs) for different entities and concepts in the tourism domain. An example DTD used in DIMS for different accommodation facilities such as hotels, is presented in Figure 5.14.

The main relationship between the DTDs and the current Service Interface Catalogue definitions, is realized by analyzing the types of the parameters of the methods associated with specific classes or interfaces defined in the catalogue. Namely, the types of the objects passed as parameters of methods defined in classes and interfaces must match with DTDs defined in the Object Interface Definitions Catalogue.

Therefore, the complete database definitions for the Service Interface Definitions Catalogue reflect the link between the two catalogues. Furthermore, the Object In-
terface Definitions Catalogue entries are stored as a set of DTDs represented as XML schema documents, which suggests that the internal representation can be in simple text format. However, the definition of the Object Interface Definitions Catalogue structures is still being achieved.

5.4.3 Application of Internet Technology for Extended DIMS Implementation

Besides the analysis of information management requirements for the tourism sector described in Section 5.3, and after the identification of the main required technology and tools to be used, it is necessary to perform a more detailed analysis of the specific features of existing Internet technology and development tools and systems to be considered for the implementation of the extended architecture of the DIMS system. Basically, the scope of this analysis of existing technology and development tools and systems is delimited by the following points:

- Java programming language
- Extensible Markup Language - XML
- Jini
- Oracle8i DBMS

To achieve a complete and exhaustive description of each of these standards and tools is not the aim of this chapter. However, in the next sections each of these key points is briefly described and analyzed, given the relevance of their practical application in the information management platform implemented to support a tourism VE infrastructure.

```
<-- DTD of ACCOMODATION -->
<-- GENERAL -->
<ELEMENT general (id,name+,description*)>
<ELEMENT id (#PCDATA)>
<ELEMENT name (#PCDATA)>
<ATTLIST name language (ESIENIDEIITIELIPT(IFR) #REQUIRED>
<ELEMENT description (#PCDATA)>
<ATTLIST description language (ESIENIDEIITIELIPT(IFR) #REQUIRED>
<-- ADDRESS -->
<ELEMENT address (area, street+, streetnbr?, postal-code?, areacitycode?, phone*,fax?, email*, url+)>  
<ELEMENT area (#PCDATA)>
<ELEMENT street (#PCDATA)>
<ATTLIST street language (ESIENIDEIITIELIPT(IFR) #REQUIRED>
<ELEMENT streetnbr (#PCDATA)>
<ELEMENT postal-code (#PCDATA)>
<ELEMENT areacitycode (#PCDATA)>
<ELEMENT phone (#PCDATA)>
<ATTLIST phone language (ESIENIDEIITIELIPT(IFR) #REQUIRED>
<ELEMENT fax (#PCDATA)>
<ELEMENT email (#PCDATA)>
<ATTLIST email language (ESIENIDEIITIELIPT(IFR) #REQUIRED>
<ELEMENT url (#PCDATA)>
<ATTLIST url language (ESIENIDEIITIELIPT(IFR) #REQUIRED>
...
```

Figure 5.14: Example of partial XML definitions for accommodation data types.
Java programming language

The Java programming language has been the chosen development language for the extended DIMS components. Namely, Java is the best option in order to implement a federation of tourism services as a set of Web-based components and applications, for which client interfaces can be executed by tourism service users on a wide diversity of system platforms. The extended DIMS implementation has been therefore carried out using Java, and exploiting the specific features that this language presents in terms of information management support in a distributed environment (see for instance [36]). In this section, some of the particular features that were applied in different development stages of the DIMS software components, are described.

The main Java features and mechanisms that are relevant to the design and implementation of the extended DIMS are:

- Multi-thread support mechanisms. Java is a language that offers advanced functionalities to support the execution of multiple threads and lightweight processes in order to handle different jobs at the same time. For a federated information management system such as DIMS, the multithreading facilities are crucial for the implementation of different modules. For instance, the advanced multi-threaded functionalities of Java are used in the DIMS application servers and in the implementation of the DSLM module.

- Java Database Connectivity. The Java Database Connectivity (JDBC) API and drivers provide application developers with a standard access interface for a wide range of databases irrespectively of the system platform and specificities of the underlying database management system. For instance, a database application based on JDBC is able to connect and use any other data source that provides a JDBC driver, such as Oracle, Access, Sybase, etc. Generalizing this feature, it is also possible by using JDBC, to develop applications that are capable of simultaneously interoperate in the same way with different databases, provided that these databases offer a JDBC-compliant driver [36]. JDBC can also play an important role in the task of integration of information coming from external existing legacy systems. JDBC is therefore a powerful tool to achieve multi-platform database connectivity in a distributed environment. Thus, the extended DIMS Application Server prototypes are currently based on the JDBC approach to interact with the underlying DBMS. The DIMS clients interact with the application servers via RMI or Jini services.

- Graphical User Interface (GUI). The DIMS implementation should also support a user interface through which, for instance, particular kinds of distributed queries on VE information can be issued, and the results are presented back to the end user. Here, the graphical interface components of the system should be able to run on multiple platforms and to present the data coming from different distributed databases. The GUI that is developed for the prototype of the Service Interface Definitions Catalogue of DIMS is based on the Java Foundation Classes (JFC), which define a complete set of graphic interface components for
Java applications. The Swing classes included in the JFC support GUI elements such as windows and frames, panels, dialog boxes, menus and toolbars, buttons, sliders, combo boxes, text components, tables, lists, and trees among other interface elements. The DIMS system implementation uses these facilities for designing and implementing the corresponding end-user interfaces.

- XML processing facilities. Another recognized advantage of Java regarding the implementation of the DIMS, resides in the fact that it provides certain features that facilitate the sharing and processing of data and documents using XML [113]. By combining the neutral data definitions of XML with portable applications written in Java, it is therefore possible to develop applications that can be downloaded over the Internet on any virtual machine, and that can also represent and exchange data in an open and language-neutral manner (see [113]). In other words, the utilization of Java together with XML, potentially increases the degree of interoperability between applications both inside and outside enterprises. Furthermore, the Java platform has been very valuable for implementing XML interfaces such as SAX (Simple API for XML). Other points that illustrate the benefits of using Java and XML can be found in [113].

The exploitation of these Java facilities has been done during the implementation of different modules of the extended DIMS.

Extensible Markup Language – XML

XML was developed by an XML Working Group formed under the World Wide Web Consortium (W3C) in 1996 [161]. In short, XML is a markup language used to describe structured documents (see also [162, 163] for a brief description of XML). The relevance of XML to support tourism VE infrastructures has been well recognized in projects such as FETISH. In the case of the DIMS, different XML data representations and tools can be applied to support different functionalities, including for instance, the Object Interface Definitions Catalogue Manager, as described in Section 5.3.

Also, the on-going initiatives regarding implementation of an XML data binding for Java promise to be fruitful in the context of future DIMS developments and extensions. The main objective of the XML data binding for Java, is to automatically translate XML schema definitions into Java classes [23]. These classes will actually carry out the work of parsing the XML code, building the internal data structures, and validating data contents.

Data binding facilities promise to provide a better support for XML data validation than other approaches to process XML documents, such as DOM [160] or SAX [66]. The idea is that the classes that are automatically generated based on XML schemas, already include the code that performs both structure and content validation [137]. Therefore, the classes can be regenerated at any time that the schema changes, relieving awkward maintenance tasks. Furthermore, applications using these classes can have a similar performance as SAX applications, while having an in-memory data structure as with DOM.
As an example, the project "Adelard" (currently known as JABX) was initiated by Sun in order to maximize the efficiency of XML-processing applications, based on the presented concepts of XML Data Binding [110]. Although this technology is still evolving, its application in the DIMS must be considered and evaluated in the future.

**Jini**

As described previously in Section 5.2.2, the facilities and advanced features of the Java/Jini application development environments can be used and exploited for tourism VE support infrastructures. In particular, in the case of the DIMS component, Jini is applied in the design and implementation phases whenever appropriate and/or necessary. For example, the implementation of the FQP/DSDL module of DIMS relies on the Jini approach to implement the distributed look-up service facilities, as illustrated in Section 5.4.1.

**Oracle8i**

The chosen DBMS to implement the extended DIMS architecture is Oracle8i Enterprise Edition (however, most of the described DIMS functionality described in this chapter still needs to be migrated from Oracle version 7.3). In general, this DBMS offers a set of functionalities in terms of efficiency, reliability, security, availability, etc. that will support the basic data management facilities of the DIMS. As was mentioned in Chapter 4, Oracle 7 has been already successfully used in the implementation of the DIMS in PRODNET [77]. Furthermore, the adopted 8i release of Oracle focuses on Internet computing and provides additional support for Java, XML, Web browser applications, and object-relational data management.

Finally, it must be mentioned that the use of Oracle products within the DIMS implementation has been limited to the database server together with certain generic interfaces and programming tools (see for instance Figure 4.4), in order to keep the level of dependency on a specific DBMS as low as possible. For this reason, advanced Oracle products such as automatic form generators, business-to-business development suits, and Internet directory components, were not used for the DIMS implementation.

**Extended DIMS Implementation Environment and Tools**

The implementation of several extensions to the DIMS component, as presented in Section 5.4, has been developed within the 5FP FETISH project using the Java 2 platform version 1.1.2. Furthermore, it can be mentioned that Forte, a Java Integrated Development Environment provided by Sun on free basis, has been used. As mentioned previously, the chosen internal database is Oracle8i (although version 7.3 is still being used). Also, most data access requests between DIMS client applications (or applets) and DIMS application servers are handled through Remote Method Invocation (RMI) mechanisms, and the application servers communicate with the internal DBMS mainly through JDBC. A set of public XML document processing and Java binding tools are being evaluated to be used for further implementation of other DIMS features.
5.5 Conclusions and Future Work

One of the main conclusions of this chapter is that the DIMS federated information management architecture presented in Chapters 3 and 4 of this thesis can be successfully extended and used to support VE infrastructures in substantially different VE application domains. Namely, in previous chapters of this thesis it was illustrated how the federated DIMS architecture approach can be applied to support VEs in the production-oriented industrial manufacturing sector, while in this chapter the DIMS architecture was applied to support VEs in the service-oriented sector (more specifically, in the tourism service industry). In order to further clarify this point, Table 5.2 summarizes and compares the application of the main federated architecture issues and components in both VE domains, considering the general federated information management requirements, federated integrated schema, federated query processing, and the federated export schema definitions. A more detailed analysis of the integration of these two VE application cases is outside the scope of this chapter and is being addressed as part of on-going research projects such as FETISH.

In particular, the application of the federated information management architecture to properly manage distributed service proxies (i.e. active functions) instead of applying it to manage information that traditionally encompasses only “static data items”, represents one of the distinguishing features of the extended DIMS federated approach proposed in this chapter. Please also notice that the the way in which the global federation of services is represented as a distributed hierarchy or graph of service directory nodes from where the service providers and end users must have protected access to specific service proxies, also has an impact in the way in which different federated architecture components need to be designed and implemented (see Table 5.2).

Furthermore, this chapter also presented the way in which several advanced Internet standards and development tools can be applied in the design and implementation of the extended features of the DIMS components, such as the Service Interface Definitions Catalogue. For example, the extended DIMS design and implementation considered the application of: Java as programming language and development platform; XML for data representation and exchange; XML data binding facilities with Java; Jini technology to support management of distributed services; and Oracle8i as the back end database server of the federated system.

Please also notice that the detailed design and development phases of the extended DIMS are still in progress. Thus, there are many other possible directions and extensions for the work presented in this chapter. For example, the following points can be addressed as part of future work:

- Versioning control mechanisms can be developed to support the evolution of both service and object interface definitions. Proper control of different versions of service and data type definitions is important in order to reinforce future compatibility and interoperability, specially for the continued use of value-added services among tourism enterprises.

- “Data/meta-data export” facilities can be added to the service/object interface
definition catalogue applications. Through this mechanism, definitions of catalogue entries can be exported as Java files, that can be used by service providers in order to start the implementation of complying Java/Jini proxies.

- Advanced query/search facilities on catalogue information can be incorporated to facilitate the browsing and querying tasks on information related to the service interfaces and proxies.

- The implementation and design of some of the described DIMS functionalities need to be further extended and detailed in the future, including the end-user editor for setting the access rights through the FARM module, and the support for distributed business processes for VASs and for PROMAN.

As final remark, it is foreseen that the same general information management requirements that identified for the design and implementation of a federated information management architecture in two substantially different VE application domains (i.e. tourism services and industrial manufacturing), will also be identified in other kinds of VE domains and collaborative scenarios, such as: concurrent engineering, agribusiness, agile distributed scheduling, elderly health care, etc. For example, the federated information management architecture has been suitable to support distributed manufacturing scheduling for VEs in the context of the INCO-KIT MASSYVE project [128, 127].
<table>
<thead>
<tr>
<th>Federated DBMS</th>
<th>Service-oriented VEs</th>
<th>Production-oriented VEs</th>
</tr>
</thead>
<tbody>
<tr>
<td><strong>Requirement Analysis Issues</strong></td>
<td></td>
<td></td>
</tr>
<tr>
<td>General need for distributed information sharing among autonomous nodes.</td>
<td>Independent tourism service providers need to share internal resources with service requesters and end-users to achieve the VE goals (mostly focusing on VASs).</td>
<td>Autonomous SMEs need to share part of their local information with other VE partners to achieve the final VE goals (mostly focusing on value-added products).</td>
</tr>
<tr>
<td>Need for definition of access rights through export schemas on local data/services.</td>
<td>Access to tourism enterprise services and resources must be protected and selective, depending on agreements, VE contracts, etc.</td>
<td>VE members need support for fine-grained access rights on shared data in order to reinforce the trust and collaboration among enterprises.</td>
</tr>
<tr>
<td>Federated support for distributed coordination and monitoring of VE tasks.</td>
<td>VASs are represented as DBPs, for which information can be managed by the federated system in order to support VE monitoring and coordination tasks.</td>
<td>VE tasks can be modeled as DBPs representing the distributed production chain among VE partners; federated information management can properly support coordination/monitoring tasks.</td>
</tr>
<tr>
<td><strong>Architectural Design Components</strong></td>
<td></td>
<td></td>
</tr>
<tr>
<td>Concept of federated architecture node.</td>
<td>A federated node is represented by a node in a tree or graph of distributed service directory nodes; each node stores and manages a set of service proxies that are selectively offered to the federation by different service provider enterprises.</td>
<td>A federated node is represented by a specific VE member enterprise node; the federated system stores and manages information that is accessible only to the particular enterprises associated with the VE node.</td>
</tr>
<tr>
<td>Federated Integrated Schema and Associated Data.</td>
<td>Part of the integrated schema is represented by the service and object interface definitions that are stored in distributed service directories. The service proxies represent the “pointers to the data” (i.e., service wrappers) complying with the integrated schema, which always reside at the service provider site.</td>
<td>The common integrated schema (encompassing local, import, export and integrated VE information) is represented and accessed at each VE enterprise node. The data associated with the integrated schema is temporarily pushed/pulled from internal systems into the cooperation layer.</td>
</tr>
<tr>
<td>Federated Query Processing (FQP).</td>
<td>“service-oriented” FQP is carried out by the distributed look-up mechanism implemented on each node: a service requester can issue queries on the service catalogue asking for proxies that can be physically located at other nodes. The physical location of the proxies is irrelevant and at each node access rights of the query issuer are validated.</td>
<td>FQP is carried out to handle the local and federated queries issued on the integrated schema, while respecting the access rights defined at each node through the export schemas. Federated queries will be decomposed into subqueries, that will be in turn sent to other enterprises for execution and the retrieved results will be merged to generate the FQP answer.</td>
</tr>
<tr>
<td>Export Schema Management (ESM).</td>
<td>ESM supports the Federated Access Right Manager (FARM) module at each directory node, through which service providers can autonomously define the proper access rights for other individual (or groups of) enterprises or end users.</td>
<td>ESM is done through the ESMT, which allows each VE member to create export schema hierarchies that implement the proper access rights and visibility levels on the local information that needs to be shared with other VE members.</td>
</tr>
</tbody>
</table>

Table 5.2: Application of federated architecture approaches in different VE domains.