

# SMART: Towards Spatial Internet Marketplaces

David J. Abel, Volker Gaede, Kerry L. Taylor, Xiaofang Zhou

CSIRO Mathematical and Information Sciences

GPO Box 664, Canberra, ACT 2601, Australia

{Dave.Abel,Volker.Gaede,Kerry.Taylor,Xiaofang.Zhou}@cmis.csiro.au

**Abstract.** Spatial Internet Marketplaces are attractive as a mechanism to enable spatial applications to be built drawing on remote services for supply of data and for data manipulation. They differ from the usual distributed systems by treating the services as published by providers on the Internet and bought by customers on an as-required basis. Design of a Spatial Internet Marketplace essentially seeks to overcome the problems of heterogeneity in large collections of autonomously-provided resources while avoiding complex requirements for publication. The architecture for a marketplace is considered in terms of the components required and the special issues in constructing an infrastructure for a Spatial Internet Marketplace. We initially describe the SMART (Spatial Marketplace) basic model, with its four service types of query, function, planning and execution services and different message types. A prototype application, the ACT-TAP system, is sketched to demonstrate an application of the SMART model. Some desirable extensions to the basic model to expand its functionality are considered. Finally we consider some important open research questions for marketplaces.

## 1 Introduction

Spatial Information System (SIS) applications have traditionally been built as local collections of databases, software, hardware and operational procedures. However, there remain some important and interesting potential applications of SIS technology which often cannot be established using purely local resources. One example is in Earth Observation (Sarrat et al. 1995). Tasks such as near-real-time monitoring of environmental disasters require rapid processing of satellite imagery which in turn requires high-performance computing facilities and specialist software. The cost of assembling these components as a local resource is a barrier for many prospective users. A second example is in Spatial Decision Support Systems (SDSS) for ill-defined problems. Here there is typically a high likelihood that the investigation will discover needs for more data and more forms of visualization, simulation and optimization as the problem becomes better understood (Cameron and Abel 1996). The delays in acquiring the data and software and in integrating it within the SDSS can severely limit the applicability of SDSS to problems which must be resolved quickly.

Advances in federated databases and distributed processing allow an application to be built from resources accessible through a network. With off-the-shelf technology, for example, it is now readily possible to establish a multi-site corporate information system with core databases and many applications systems. Tools such as CORBA (Orfali, Harkey, and Edwards 1996) simplify the use of computational procedures at remote sites. A relatively new concept with great promise

is the Internet marketplace (Bhargava et al. 1996; Bhargava et al. 1997), which envisages data and computational services (e.g., analysis and optimization software modules) being available on the Internet and a customer *buying* these services when required. In such a marketplace, a user might perform a complex operation by obtaining data from a data provider and passing it on for processing through a sequence of computational service providers. An Internet marketplace would be established using existing distributed systems technology. A key design challenge is to accommodate a potentially large and heterogeneous collection of autonomous services while providing levels of simplicity and uniformity for providers and customers to encourage participation in the marketplace. Common elements of proposals to date are to assume a certain community of interest and stateless interaction between customers and providers. Both restrict the scope of the design problem.

In this paper, we consider the concept of a Spatial Internet Marketplace in which a customer can buy data, computational and other services when required. Like the proponents of marketplaces aimed at other communities of interest, our basic motivation is to make spatial data and computational facilities for its manipulation more widely accessible. For example, a marketplace aimed at the Earth Observation community of interest might include data services to supply satellite imagery and ancillary data sets, computational services to perform image rectification and enhancement, and modeling services to estimate yields from the observed current state of crops. An analyst wishing to predict the production of wheat in the current season might then buy both the data and the processing, receiving at his desktop system only the final products of the analysis. The analyst would essentially buy the derived data product, would have a high degree of flexibility in choosing the products, and would pay on a *by use* basis rather than investing heavily on hardware and software. Similarly, an environmental manager might investigate a problem not by integrating data and modeling software locally in an SDSS, but by using services offered on the Internet by government or private-sector providers. In most cases, of course, a user will blend use of local resources and of marketplace services, making *purchase or rent* decisions.

We are particularly concerned to establish the key enabling components of a spatial marketplace and the research issues. While there are many issues of charging for data and services, ownership, security and so on, these are outside our scope. We are especially interested in determining what special issues are present in a spatial marketplace and to assess how the design problems might differ from those of other marketplaces.

We begin by reviewing reported work on marketplaces to note the major functional components and noting how a marketplace differs from other forms of distributed systems. In Section 3, we consider whether a spatial marketplace does indeed present distinctive problems. After introducing a running example in Section 4, we describe the infrastructure for the Spatial Marketplace. A pilot system, the Australian Capital Territory Tourism Advisory Prototype (ACT-TAP), is reported in Section 5. We conclude by assessing the scope of SMART and by discussing some important open questions.

## 2 What Would an Internet Marketplace Be?

Before we discuss in what aspects a spatial marketplace differs from other previously proposed marketplaces, we introduce some definitions that are essential for the remainder of the paper. We describe a marketplace in terms of the types of services available through it and the ways in which providers and customers participate.

At first glance an electronic marketplace consists of a set of customers and providers who interact with each other through an electronic medium such as the Internet. In this sense there is little difference between a traditional and an electronic marketplace. Providers (vendors) offer various kinds of services and customers make use of these services. However the “goods” and services offered through an electronic marketplace are considerably different to traditional ones. In electronic markets consumers are interested in using computational services and data, made accessible by providers. As the name suggests, data services make data available by allowing its extraction from structured databases or native file systems. Computational services offer a range of software modules for analysis, simulation, optimization, transformation and so on. Depending on the execution paradigm, these modules are either executed at the provider’s site or the customer’s site. In order to assist customers in finding particular services, there have to be distinguished directory services, called brokers, whose function is similar to the yellow pages. By means of these brokers it is possible for a customer to identify services relevant to a task, to plan a sequence of invocations of services, and to invoke the services.

In summary, our refined model of an electronic marketplace requires providers to offer at least one of the following services: data, computational or broker services in order to participate. By advertising these services, providers enter the marketplace and make their service public. The advertised information is accessible for customers, who can buy services by passing requests to the chosen providers.

In outlining our refined model of a marketplace, we already described some of the actions necessary in a marketplace. Following Günther et al. (1996), we identify the following actions as necessary in a marketplace.

- *Resource advertisement (resource description)*. A provider registers a service in a catalogue.
- *Resource discovery and selection*. A customer searches catalogues for query, computational or broker services matching a description of a required service.
- *Data translation*. A provider transforms a data set to a form suitable for submission to a service provider. The transformation can involve mapping from one underlying data and process model to another, transformation of attribute values (e.g., conversion of units of measurement), or translating the data into a required transmission format.
- *Task planning*. A sequence of invocations of operations by one or more providers (a plan) is constructed. Construction might involve consideration of alternative plans and selection of the optimal plan (in terms of, for example, minimum elapsed time to complete the task).
- *Command translation*. The expression of an operation is re-written in the native command language for a target service.

- *Task execution.* Upon execution, the customer or an agent supervises execution, taking appropriate actions on error conditions.

For a marketplace to be attractive it is necessary to have an extensive set of providers offering a diversity of services but conforming to a set of standards which provide some degree of uniformity to the customer. Clearly, some degree of standardization is needed to soften the effects of heterogeneity in the marketplace. Importantly, this standardization relates to the customer interface (the external interface) for a service, and does not restrict the internal implementation.

We can now express our view of an Internet marketplace more succinctly by comparing it with previous proposals.

### 3 What is Special about a Spatial Internet Marketplace

Proposals for Internet marketplaces integrating data and computational services originated in different fields. For example, in the area of Decision Support Systems the primary motivation for setting up such a marketplace has been the prospect of greater diffusion of decision support technologies, either by access to specific tools (Bhargava et al. 1996) or by building Decision Support Systems from services accessible through a network (Jeusfeld and Bui 1997). Within this field, the marketplace is presented in terms of providers offering to solve problems presented by customers, using software available at the respective sites. In order to use this service, customers have to ship their data to the provider's site. This is in contrast to the function of *software* or *model repositories*, which offer convenient access to libraries from which a customer can extract modules for use at his own site (Dongarra and Grosse 1987). The software comes to the customer.

Examples of systems implementing marketplaces for mathematical services are DecisionNet (Bhargava et al. 1995) and MMM (Günther et al. 1996). These systems only provide support for a subset of the actions we identified in the previous section. There is a focus on computational processes and an absence of data services in the experimental systems. They also lack support for automatic planning and sophisticated executing of tasks requiring use of a sequence of services.

Given the design of these Internet marketplaces, the question is whether a spatial marketplace is any different to earlier proposed models (Abel 1997). A first preliminary analysis of the likely applications which might use a Spatial Internet Marketplace indicate that 'spatial' imposes different requirements on marketplace design and implementation. We see the following differences to other marketplaces.

First, in a spatial marketplace many customers will wish to *buy both data and computational services*. For example, we expect that digital spatial data is often acquired from a supplier rather than captured by a user. This reflects the increasing costs of acquiring spatial data and the central role of government agencies in assembling and maintaining spatial data sets. Increasingly, organizations are likely to source at least part of their data from external providers.

Second, geographic space acts as a *common domain* for spatially-referenced objects. That is, two objects from autonomously-developed databases which both refer to the same geographic space can be associated by their spatial relationships. Suppose, for example, there is a database of the

spatial distribution of soil types in a region, and we have a record of production of wheat at a certain location, we can then determine the type of soil on which the wheat crop was grown by associating these two databases. Within the constraints imposed by the spatial descriptions (e.g., imprecision), it is possible for an application to draw on databases from different providers, with schema integration performed within the application. We are not aware of any other discipline where there is a similar common domain which allows objects from different databases to be readily associated. This means that a Spatial Internet Marketplace has a greater potential than other types of marketplaces for supporting the automatic selection of data resources and for planning a sequence of tasks without schema integration.

Third, spatial information is relevant for *many communities of interests* such as land administration management, urban planning, environmental management, transportation planning, minerals exploration, and so on. At present, each of these communities applies its own data and process models and has its own specialized forms of data manipulation. This makes it likely that the various so far separated communities will contribute and use resources of a Spatial Internet Marketplace. However, sharing of resources among different communities of interest is considerably harder than in marketplaces where a tight community of interest allows common data and process models, and their semantics, to be assumed. A Spatial Internet Marketplace needs to make explicit provision for describing type systems and semantics.

Clearly, the design and implementation of a Spatial Internet Marketplace encounters problems similar to those in other fields dealing with the integration of heterogeneous services and draws on the same technology base for distributed systems. A comparison of the marketplace with other forms of systems such as multidatabase or workflow management systems is useful to clarify further what a marketplace is and to identify which techniques can be borrowed from these fields as a foundation for a marketplace.

Multidatabase systems deal with uniform access to a collection of autonomous, heterogeneous and distributed databases. The usual reference model (Sheth and Larson 1990) addresses heterogeneity by requiring the services of the site databases to be represented in an export schema conforming to a chosen data model and by including wrappers for site databases to offer a uniform external interface. In order to relate data across different site databases, it is necessary to perform schema integration, the result of which is stored in a *global data dictionary*. Schema integration is necessary, because the underlying application domains are often quite different and it is not immediately clear how they relate to each other and how to combine them.

This is in contrast to workflow management systems which have traditionally been used for fairly restricted application domains such as business processing (Lomet 1993). In traditional workflow management systems the integration and the access of multiple systems in heterogeneous, distributed environments is greatly facilitated by common and simple exchange formats (e.g. document, electronic mail) and the use of simple computational services. Due to these restrictions, Workflow management systems have attracted little attention outside of business processing and not until recently researchers have applied workflow ideas to other areas (Alonso and Hagen 1997). Compared to multidatabase systems, workflow management systems offer limited support for trans-

action management and no support for automatic planning. The sequence of actions (i.e., plan or process) is specified in a procedural way by a user, who determines in detail the execution by setting explicitly synchronization points and specifying possible alternatives. By contrast, in a multidatabase system, a client issues commands to the system in terms of the objects in the global data dictionary and the query execution planner performs without human assistance a mapping of the user's request to operations on the site databases. Therefore, workflow management systems are much more human centered than multidatabases.

Workflow management systems offer a range of interesting and desirable features not found in multidatabase systems but desirable in a marketplace. For example, in addition to provision for the access of database systems, they support the integration of computational services as typically found in a marketplace. We note, however, that the 'external operations' concept (Schek and Wolf 1993) offers a means for multidatabase systems to incorporate sites providing computational services. Another very useful concept of workflow systems is the idea of roles, that is, certain services (persons in a workflow) fulfill certain tasks within the process. If a service cannot fulfil its role, it is possible to nominate a substitute that can take over the request and dispatch it. In a marketplace this corresponds to service substitution, which is of crucial importance in a dynamic environment. Thus, workflow management systems seem to be much more adaptable to a dynamic environment than multidatabase systems. Compared to a marketplace infrastructure, a workflow management system does not cater for resource discovery.

In summary, many of the ideas of workflow management and multidatabase systems are present in a marketplace implementation but are heavily modified. In outline, a multidatabase solution is aimed at integrating a known set of databases within an enterprise while the marketplace solution is aimed at the publication of essentially an arbitrary set of services on a commercial or quasi-commercial basis for general use. As there is a large number of services in the marketplace, schema integration to provide a unified view of the services would be difficult, and we envisage that this would be usually left to a designer on an application-by-application basis. It is to be expected that schema integration in a multidatabase is replaced to some extent by resource discovery. By this we mean that in a Spatial Internet Marketplace, customers (or agents) are more likely to search for readily usable services instead of trying to integrate services manually. In order to identify data or functions suitable for a task at run-time, it is necessary to search existing registries which store information about the services advertised. Integration in a Spatial Internet Marketplace is then automatically performed on a task-by-task basis, either for the purposes of an application (i.e. the plan is devised and stored, and repeatedly executed) or is done dynamically. Heterogeneity is accommodated either by requiring some degree of uniformity in the external interfaces of services or by providing transformation services. Table 1 juxtaposes the different approaches.

For completeness, we observe that the work of the Open GIS Consortium is applicable to the development of Spatial Internet Marketplaces. Essentially, the Open Geodata Interoperability Specifications (Open GIS Consortium 1997) offers a classification of services and a spatial data model which provides a basis for foundation elements for standards to facilitate the transfer of data within a Spatial Internet Marketplace.

Feature	MDB	Workflow	Marketplace
Automatic planning	yes	no	possible
Transaction support	yes	little	little
Human centred	no	yes	possible
Integrated schema	yes	no	no
Coupling	tight	loose	very loose
Execution specification	declarative	procedural	declarative
Requirements imposed on participant	high	low	low
Role concept	no	yes	yes
Adaptability	low	high	very high
Resource discovery	no	no	yes

Table 1. Feature Comparison.

## 4 The SMART Architecture

### 4.1 SMART Design Intent

The SMART (Spatial Marketplace) project is exploring the design, implementation, and use of Spatial Internet Marketplaces. The project is generally aimed at establishing an architectural model to guide the specification of key standards and protocols, at building key components such as planners, and testing and demonstrating marketplaces through a series of pilot applications. Our design of the SMART architecture is guided by a few objectives, including

First, *simple access* for customers and service providers is clearly the key for the success of a spatial marketplace. Obviously, the more complex and costly it is to publish a service, the fewer will be the services. On the other hand, if potential customers have to follow a steep learning curve before they can take advantage of the offered services, is not attractive to use them. This approach is motivated by the success of the World Wide Web, where the simplicity of establishing Web pages has encouraged a very large number of providers to offer a great range of material. Consequently, in order to encourage possible providers to contribute to a spatial marketplace, we aim at a minimal set of mandatory requirements to be provided by services.

Second, SMART should *use existing standards*, protocols and tools wherever possible. Apart from limiting the effort in building a system by using familiar well-known tools as a starting point, the adoption of standards also facilitates the integration, interconnection, and communication with other technology providers. By doing so, SMART permits other services outside of the spatial marketplace to benefit from its services. On the other hand, it enables SMART services to exchange computational resources with other communities, which greatly contributes to its flexibility.

Third, *extensibility* is a deliberate design objective. We see extensibility as the key to supporting multiple communities of interest in such areas as definitions of data types, the computational services offered, the descriptions of services in registers, and so on. Extensibility is also vital for the

incorporation of new services and new technologies.

Fourth, *scalability* is also of crucial importance for the success of a spatial marketplace. If the system does not adapt gracefully to the growth of the marketplace and it cannot provide efficient access to its services, participation is clearly discouraged. We therefore adopt a distributed approach with clusters formed by communities of interest.

Fifth, *manageability* and *reusability* are very important for the long term success of the SMART system, since we expect that the marketplace is going to evolve over time. We therefore apply object-oriented design principles, which allow a high-level specification while encapsulating the concrete implementation. Moreover, we consider mechanisms such as late binding and polymorphism as being essential in a dynamically changing environment such an Internet marketplace.

## 4.2 Roadmap

A SMART marketplace consists of a set of customers (clients) and a set of service providers. Providers advertise to a registry the services available at their sites and customers can locate services by searching registries for relevant services. Since we expect that most services are not native SMART services, they typically consist of a SMART compliant wrapper which encapsulates the concrete underlying system or service (e.g., database, mathematical library). The design of the service interface follows an object-oriented design with superclass `Service`. This class specifies the set of methods and attributes common to all SMART services and the interface of all other services are derived from it by means of inheritance. Thus the class `Service` establishes the minimal requirements imposed on each SMART compliant service and each provider has to offer at least these attributes and methods in order to participate in a spatial marketplace. It is up the respective services to provide additional attributes or methods, required by the specific service. Polymorphism allows methods to be overwritten and late binding guarantees that the correct method is executed at run time.

Basically, we distinguish four types of service:

- *Query services* provide retrieval of data maintained by the provider in a local data store;
- *Function services* provide a range of computational services including data transformation, analysis, predictive simulation, optimization, inferencing and so on;
- *Planning services* provide resource discovery, selection and task planning;
- *Execution services* execute plans prepared by a planning service or by a customer.

Figure 1 depicts the SMART architecture.

Services interact by passing information using some agreed protocols (e.g., HTTP, MIME). The interaction is coarse-grained (stateless) which means in the simplest case that a customer makes a request to a service and receives the results. At present we distinguish three message types:

1. a *service request* is passed to a chosen service to invoke an action by using the Request Specification Language which is basically a first-order predicate query language with arithmetic constraints;

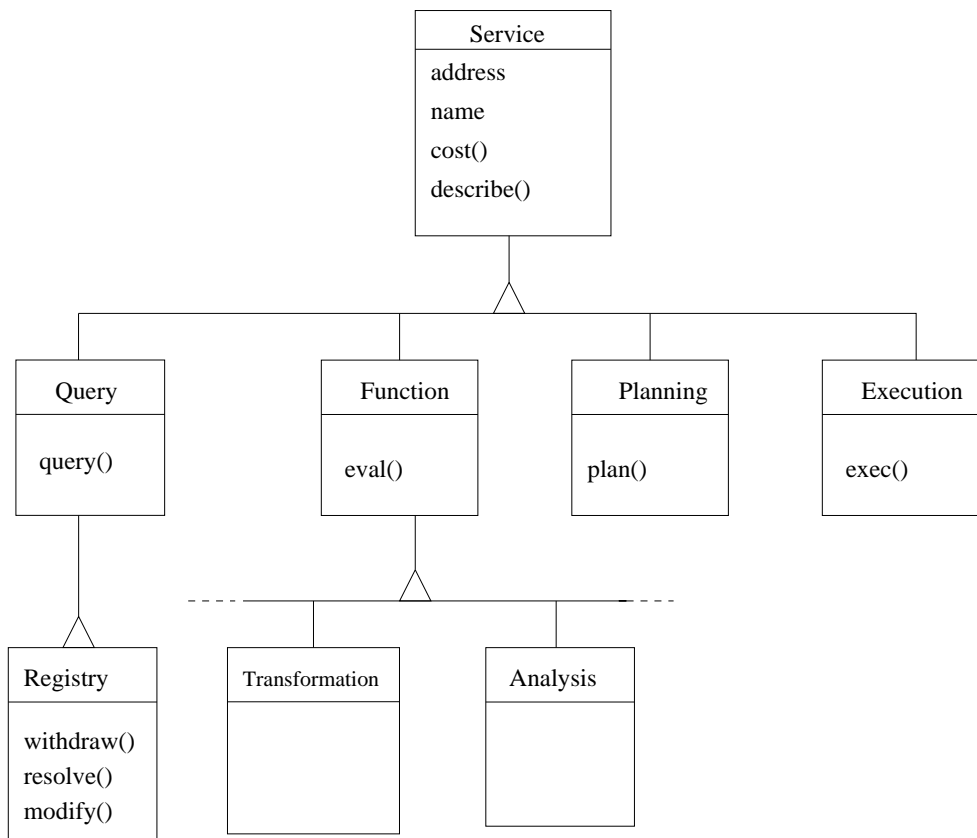


Fig. 1. Architecture of SMART.

2. a *data stream* transfers data values of requested objects and attributes from one service to another or back to the customer;
3. a *plan* specifies the service classes (that is a set of equivalent services with respect to the request) to be invoked in order to meet the customer's requirements, but does not necessarily fix the their invocation sequence. Plans can be developed either by a customer using SMART tools or by a planning service. In general, they are self-contained programs that can be directly executed by the customers or passed on to an execution service.

In the remainder of this section, we present the basic SMART model, which reflects the current state of implementation. Since this model is fairly restricted, we discuss in Section 6 possible extensions.

### 4.3 SMART Model

**Request Specification Language** By means of the *Request Specification Language* (RSL) it is possible to express simple requests in a declarative way. This language has to meet the following criteria: (1) it can be easily translated into other languages, (2) is declarative in nature, (3) supports the most common database requirements. Any service request consists of two parts: a *constraint*

*specification* which specifies the set of objects of interest or performs the parameter binding necessary for the function invocation and a *target list* describing the required output resulting from the operation. The two parts are separated by a *#*. In its current form the RSL is equivalent to a non-recursive first-order constraint language with arithmetic constraints.

**Wrapper** In general, most services participating in a Spatial Marketplace are not native SMART services. Hence, we presuppose the existence of a SMART compliant wrapper which encapsulates the concrete underlying system such as databases or mathematical libraries. Besides being responsible for communication with other services, wrappers also have to translate SMART requests into operations that are run against the underlying system and translate the results back into a SMART compliant format, which may be transferred to the calling service or some other service.

SMART definitions do not deal with implementation details, but rather are concerned with external interfaces such as the external schema and the supported operations. That is, SMART does not prescribe how the services are implemented, neither does it impose restrictions on the services offered. A provider might even draw on other SMART services to perform an operation.

We continue with a more detailed description of the different services.

**Service** In order to be SMART compliant, a provider must offer a mandatory set of methods and attributes as specified by the **Service** class. The set consists of

– **name** and **address**:

Any service is identified uniquely by its name and address. The combination of the two attributes is used to communicate with and invoke services at this site. The address by itself is not sufficient, since at one site there might be multiple different services.

– **desc**

This attribute provides a meta description of the service in a high-level language, which is intended to be read by other services as well as human users. This description is replicated in a publicly readable registry.

– **describe()**

This method returns by default all the information available at the requested site such as collection names, offered functions, etc. By means of the request specification language it is possible to perform a well-aimed search for relevant information.

– **cost()**

This routine takes as input a valid description of the task (i.e., a request) and returns a cost estimation of how much it would cost to execute the give task. In the simplest implementation form, this routine may only return a constant value, but the approach also caters for much more sophisticated (discount) policies.

These methods essentially require that a service is self-describing. Invocation of the **desc** attribute and **describe** method by a registry (see below) or by a customer provides the information to populate the registry's catalogue or for a customer to specify a request to the service. The **cost**

method provides the information needed for customers or their proxies to choose between services offering equivalent data or computational services. Placing this method with the individual services allows the service providers to estimate costs dynamically, by reference to (for example) the current demand on them.

**Query Services** A query service fetches data from a (conceptual) data store maintained by a provider. Thus, they provide single-site database services in a Spatial Marketplace. The SMART model does not limit the form of the databases. Although in most cases query services will be implemented using structured (relational, object-relational or object-oriented) database management systems, but they might also include text database systems or native file systems.

The Request Specification Language allows us to define the set of objects of interest as well as possible local query predicates and the target list defines the attributes whose values are to be returned and the representations of the result stream. The target list may consist not only of the attributes that should be returned as a result of the query evaluation, but also of type constructors and conversion routines. The list of supported type constructors (e.g., list, bag, set) conforms to the ODMG-93 specification (Cattell 1994). The conversion routines are intended to simplify data exchange by performing certain transformations at the data site, before encoding the data into a data format used for data shipping (the default is MIME).

As an example of an invocation of a query service, let us consider the following scenario. Suppose a provider has made the following schema definition of restaurants available:

```
Restaurant(Address(Suburb: String, Street: String) : Tuple,  
           Cuisine: String, Name: String)
```

and we are now interested in finding all Chinese restaurants in a suburb called Kingston. The corresponding RSL request is

```
Restaurant.Address.Suburb = "Kingston", Restaurant.Cuisine="Chinese" #  
Restaurant.Name
```

**Registry** An important subtype of the class **Query** is **Registry**. For simplicity, we assume that there is a single instance of registry and its creation establishes a SMART marketplace. In order to make their services known to the community, the service providers have to register with the registry and provide information about their services. Since the information stored in the registry is read by human users as well as automatic planners, it has to be in a format suitable for both groups. We therefore divide the registry into the following three categories.

1. *Catalogue information* identifies a collection of services available at the marketplace along with their description;
2. A *thesaurus* and *glossary* contains a list of the descriptors used;

3. The *Type and Relationship register* stores information about types and possibly existing relationships between types, functions and data collections. The availability of such information is indispensable for automatic planning.

When the registry is created, the catalogue has an empty set of services and a foundation set of descriptors and data types chosen to meet the needs of the marketplace's intended community of interest. We are assuming that SMART services are used within a community of interest, so that there is a prespecified but extensible vocabulary with a commonly understood meaning which helps in interpreting this description. Each of the words used is explained in a publicly available glossary that can be consulted in case of doubt.

A provider adds a service to the marketplace by requesting its registration by the registry. In addition to the description of the service, the provider must also supply any descriptors and data types referenced in the description which are not yet present in the catalogue. This involves the description of the service together with the definition of any descriptors and data types.

The SMART registry extends the idea of a conventional trader service in the following ways. First, it contains a glossary which helps in interpreting the semantics of types and operations using a prespecified vocabulary. Second, it offers provisions for type management. For the trader we adopt and slightly extend the idea of the OMG Trader (OMG 1996) which enables to offer and discover instances of services in a federated environment. Furthermore, it offers provisions for applying different policies, proxy services, administering, and so on. It also stores information about the relationship of functions and data collections.

The registry also acts as a query service to allow customers or planning services to fetch descriptions of services, descriptors and data types and to search for services. Figure 2 gives a more detailed example of the appearance of a registry.

A service is also defined by a set of preconditions, which describe the domain in which the service operates reliably.

**Function Services** This class models a very broad range of computational services, including transformation between representations (e.g., conversion between coordinate systems, conversion from vector to raster representations), analysis, prediction, optimization and inferencing. The implementation of a function service might include a local data store or models referenced during an evaluation. As examples, a coordinate transformation service might store geodetic constants, and a hydrological model-based prediction service might store a representation of certain rivers using the constructs of a certain hydrological model.

For a function service, the constraint specification describes the objects to be derived or transformed. The elements of a constraint specification for a function service are the function name and a binding of its input parameters. Suppose for example a service has registered the following function for computing the shortest path between two given points (**to**, **from**) and returns the associated cost **cost** as well as the corresponding route **route**.

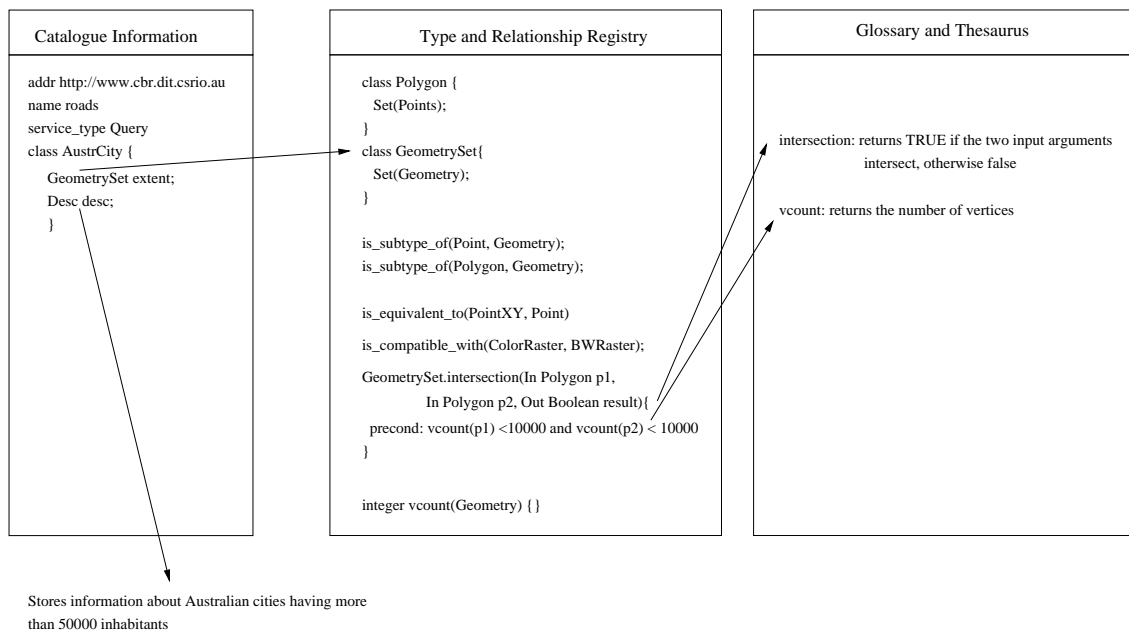


Fig. 2. Snapshot of a Possible Registry

```
shortestPath(In from : Point, In to : Point, Out cost: integer, Out route: Polyline)
```

Here **In** and **Out** denote the input or output parameters, respectively. If a parameter can be both input and output parameter, neither **In** nor **Out** are specified. All input parameters have to be bound to a value at invocation time, while any subset of the output parameters may be used in the request. Given this function definition, the following two equivalent RSL requests would return the cost associated with the shortest path between the two given input points.

```
shortestPath.(from(100.5, 100.5), to(200.1, 300.7)) # shortestPath.cost
```

```
shortestPath.from.x = 100.5, shortestPath.from.y = 100.5,
shortestPath.to.x = 200.1, shortestPath.to.y = 300.7 # shortestPath.cost
```

A function service typically has a certain scope of reliable performance of the service. This is included in the service's description as a set of preconditions for accepting a request from a customer. The scope of reliable performance can be determined by a number of factors:

- limitations of the process models underlying the function service. For example, a service to predict river flows might be based on a process model which considers only smooth flows. The service would not reliably predict flows where the flows are not smooth.
- limitations of data held or used privately by the service and used by it in derivations. For example, a service's computation may rely on data that is not updated regularly or changes

daily such as the water level of a river. If the customer specifies a river not included in that set, the service is unable to perform the prediction;

- the scope of the implementation of the algorithms. As examples, a service to perform a map overlay operation on two sets of polygons might only be able to deal with polygons of 10,000 vertices or fewer.

**Planning Service** Planning services generate plans whose execution will deliver to a customer a required set of values for a given set of input parameters. A planning service differs from a Trader by returning to the customer a program (the plan) which can be executed immediately by the customer to buy services. A Trader, on the other hand, simply returns the name and description of a single service which satisfies certain requirements. Hence, planning services in a spatial market place take over the role of the query optimizer in multidatabase systems.

The constraint list for a planning service includes:

- the attributes and their output format to be delivered as a result of plan execution;
- preferred services to be used;
- criteria of merit to choose between alternative plans. For example, a customer might wish to minimize the total charges levied by the providers of the services used or to minimize the elapsed time for execution of the plan, or to minimize the elapsed time while ensuring that the data generated achieves a desired accuracy.

The target list is empty, since a planning service can return only a plan.

Depending on the number of services involved in a plan, we distinguish between a *simple* and a *complex* plan. When executed, simple plans make only a single invocation of a service while complex plans contain a sequence of invocations of possibly many services. The more powerful use of a planning service is to determine how a number of services can be combined to provide data or to perform a manipulation which cannot be provided by any single service. For example, a customer might request a street map of the Australian Capital Territory (ACT) to be returned using the Australian Map Grid (AMG) coordinate system. The only available query service able to provide streets in the ACT can deliver data only in the latitude/longitude coordinate system. The planning service will then search for a function which can convert from latitude/longitude to AMG, and construct a plan with two invocations: the first to retrieve data from the query service and the second to pass the data to the coordinate conversion function service.

In SMART, a plan is expressed in a restricted subset of the Java language (Flanagan 1996).

Our motivation for adopting Java is simply that it is widely available and has been designed to provide such important features for Internet environments as trusted execution of a program supplied by an external agent.

The task of planning in a Spatial Internet Marketplace is considerably more complex than in multidatabases and it may not be immediately obvious that implementation is feasible. If we assume that there is a sufficiently large number of providers offering services, success or failure of planning depends on the information made available by the services and stored in the registry. This is very

important since the planner has to work hand in glove with the registry to acquire the information necessary for proper planning. Of course, a plan can be prepared manually by a customer using techniques similar to those in workflow systems or by conventional programming. To reduce the cost of repetitively generating plans for the same task, plans might be stored by the customer, possibly with some degree of parameterization.

**Execution Services** Execution services execute a plan as an agent of the customer. There are three motivations for including this service class. The first is simply convenience for the customer in transferring plan execution to an agent. In some cases, this will mean that the hardware and software requirements for a customer to operate an application will be minimized. Second, an execution service allows a plan execution to be sited remotely to a customer. For example, if a customer has a low-bandwidth connection to a set of required services, but an available execution service has a high-bandwidth connection to them, siting the plan execution at the execution service will minimize the elapsed time to complete the task. Third, an execution service allows a provider acting as a value-adding reseller to construct a service which draws on existing query and function services. This involves scripting the service as a plan invoking operations from the base services. Execution of the constructed service is then performed by passing the plan to an execution service.

An execution service has the plan as its constraint specification. The target list is empty.

## 5 Using SMART

The SMART infrastructure gives considerable flexibility to an application's developer in precisely how the services are selected and combined within an application. It is not essential that the full set of registry, planning and execution services be present for SMART-compliant services to be used in an application.

The simplest approach is to invoke the individual services directly. Here the applications developer would browse a registry to establish the services to be used and their schemas. The invocations would then be coded within the applications program.

An intermediate approach is that the applications developer would encapsulate the use of SMART services in one or more plans. These plans would be prepared by the application's developer and stored for repeated use. They would act as modules of the application, executed by being passed either to an execution service or to a Java interpreter under the control of the application.

The most sophisticated approach is where the application requests the automatic generation of a plan by a planning service at run-time. The plan is then immediately passed to an execution service for execution, and so avoids the problem of 'stale' plans.

### 5.1 An Example: ACT-TAP

In order to illustrate the viability of the SMART architecture and to show how to build integrated services using SMART-compliant services, we have implemented a prototype system called ACT-TAP (Australian Capital Territory Tourism Advisory Prototype). ACT-TAP is notionally intended

to assist tourists to devise an itinerary for visiting places (e.g., tourist attractions, restaurants) in the ACT. ACT-TAP has been implemented using a partial infrastructure. The registry service is absent: the specifications of services were taken as known to the applications developer. The planning and execution services are present in their simplest forms. In addition to these services the marketplace consists of the following services.

1. **KDB** is a query service that provides information on tourist attractions in the Australian Capital Territory (ACT) such as names, addresses, opening hours and expected durations of stay.
2. **ActMap** is a query service which stores spatial and non-spatial data of roads, land features and buildings in the ACT.
3. **RoadNet** offers a function service, which computes the shortest path between any two given locations.
4. **Scheduler** is a function service for the computation of a schedule that minimizes the total traveling distance between a given set of places. It takes into account time constraints, such as the arrival, departure time and minimum duration at each place. Unless the user specifies these constraints, the **Scheduler** access directly the **KDB** database via the SMART interface to extract the opening hours and the recommended duration of stay.

Figure 3 shows possible interactions among the different services and their role within the spatial marketplace.

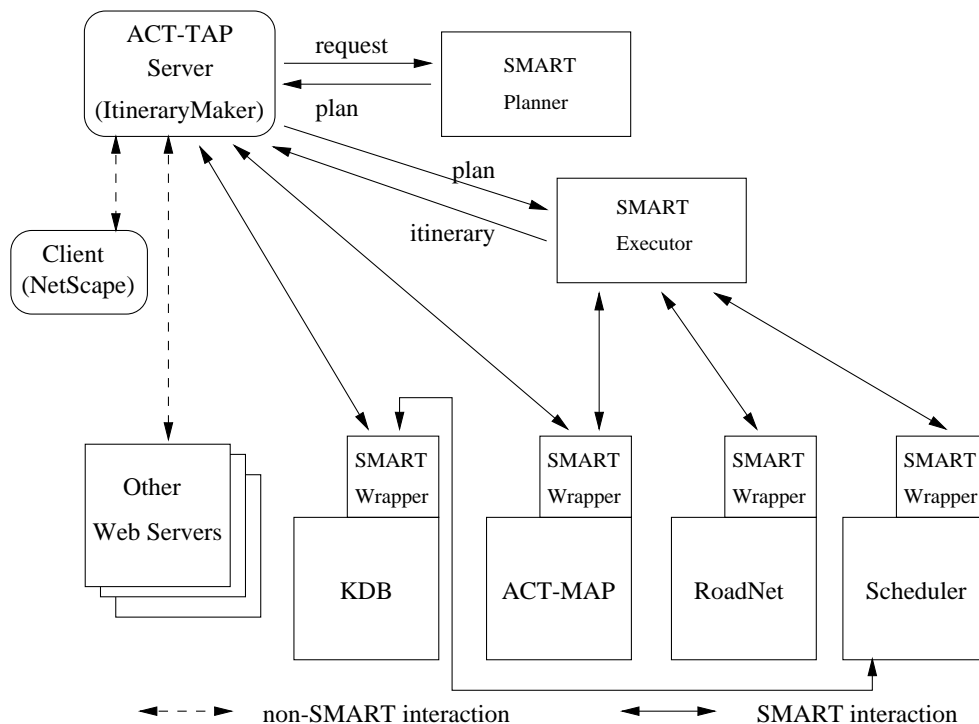


Fig. 3. SMART modules and service providers in ACT-TAP.

In addition to accessing services through the SMART interface the ACT-TAP server can also access particular sites directly. ACT-TAP itself is implemented as a Web server, which can be visited using well-known browsers. The current mode of usage is that users first select interesting places in the ACT region by using the query services **KDB** and **ActMap**. The selected places are shown in a separate window on the screen and the user is free to specify time constraints for each selected place. Once the user has finished the selection, she can initiate the generation of a plan by pressing a button. This triggers the following steps. First, a request is sent to the planning service to generate a plan for computing an itinerary. ACT-TAP uses pre-stored plans, which are Java programs with embedded SMART service invocations. Next, the returned plan is forwarded to the execution service (a Java interpreter) which in turn invokes the **ActMap** service to translate place names into locations, then **RoadNet** service to find the cost of the shortest path between each pair of locations, and finally the **Scheduler** to get an itinerary. As mentioned, the **Scheduler** may need to visit the **KDB** service through the SMART wrapper to obtain the time constraints for certain places. Finally, the execution service returns the itinerary to the ACT-TAP server.

## 6 Extensions to SMART

The description of SMART to this point has dealt with a basic architectural model for a Spatial Internet Marketplace. We now discuss some extensions to that basic model which provide some important provisions for extensibility and for accommodating alternative standards.

We have earlier noted extensibility as an important goal for the design of marketplaces. A marketplace with only a single registry, however, has some limitations in scalability and in catering for multiple communities of interest. Two forms of segmented marketplaces, the *hierarchical* and *federated marketplaces* (Figure 4), overcome these limitations to a certain degree.

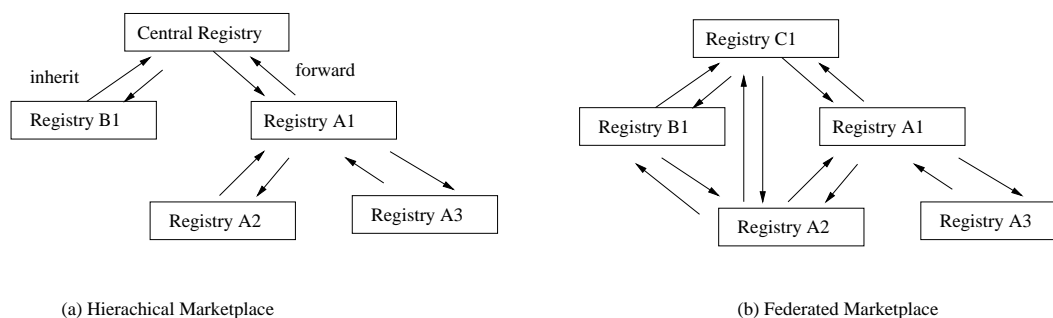


Fig. 4. Marketplaces.

An *hierarchical marketplace* is established simply by allowing a registry to register other registries, with some provisions for inheritance of the thesaurus and glossary components of catalogues. The first registry, created when the marketplace is established, holds a distinguished status as the central registry. Other registries (community registries) can be created, possibly to service a community with a common disciplinary focus or a community in a geographic region. Once established,

a registry can grant registration to other registers. A community register is established by requesting (and being granted) registration by the central register or another community register. There is then a parent-child relationship between a community registry and the registry authorizing its establishment, and the marketplace becomes a tree of registers with the central registry as the root node and the community registries as the other nodes. Attached to the nodes are the query, execution and planning services which have been registered with the corresponding registry.

On establishment, a child registry inherits the thesaurus and the type and relationship registry (i.e., the catalogue without the services registered) of its parent. The services registered with a child registry then inherit the semantics of data and function descriptors and the data types of the parent registry and extend them by their additional semantics and data types. We envisage that this behavior would allow a community registry to define a community of interest, with increasing specialization down the tree of marketplaces. If a provider registers a service with a community registry, its information is forwarded to the corresponding “parent” registry and finally to the central registry.

A *federated marketplace*, on the other hand, is established by two registries establishing an association. This could involve one registry copying to its catalogue some or all of the services registered with the other registries, together with data and function descriptors and data types. Topologically the federated marketplace is a network. An unresolved problem is reconciling autonomously-generated sets of descriptors and possibly data types. This appears to be very similar to the problem of schema integration for multidatabase systems. It appears that reconciliation of descriptors to recognize (for example) identity of concepts expressed differently in the two catalogues or the non-equivalence of descriptors appearing in both requires external knowledge and so requires manual intervention.

The design of the basic SMART architecture is biased towards simple implementation of services by providers, in keeping with our intent to encourage the availability to a customer of a large number of services. An important case in point is the expression of a request to a query service (i.e. its query language). Clearly, simplicity is often bought at the expense of expressiveness, conciseness or the set of operations supported, so that a simple query language will have limitations. Additionally, many customers will be familiar with and expect availability of languages such as SQL and many providers will be readily able to offer it. However, the current forms of SQL do not provide the rich type system supported by the basic SMART query language.

A solution is to allow providers to advertise other query languages, service-by-service, either as an alternative to the basic query language or as a replacement. The extensions to the catalogue entries appear minor, although extra complexity is required for planning services. The decision by a provider will be based on market forces. Providing both the basic and an optional query language increases the cost of implementing the query service (although the translation from the basic query language to, say, SQL3 is simple) but ensures that the service is available to a wider set of users.

As became clear from the discussion in Section 4, the planning service has to cooperate closely with the registry in order to devise a feasible plan. This, in turn, requires that the information stored in the registry is well understood by the planner and can be used without human support,

which is only possible if the description of the syntactic and semantic properties of services are formally defined. To support the formal specification it is desirable to have tools which support the simple integration of service descriptions in a simple way and hide the complexity of the underlying system from the user. In order to be extensible it is also necessary to integrate new information. In other words, SMART must enable the gradual evolution of an appropriate vocabulary within the domain of interest. When a service provider registers its service, tools that support navigation of the registries to encourage re-use of existing terms must also support the addition of new terms that are unique, meaningful and likely to be reused in subsequent service descriptions. Ideally such a tool could implement high standards without the need for manual intervention to determine the appropriateness of new terms suggested by service providers. Some early work in developing ontologies seems to be promising for this task (Gruber 1993).

## 7 Conclusions

This paper has considered Spatial Internet Marketplaces as a means of enabling wider use of Spatial Information Systems technologies by making those technologies and data resources more easily and more widely accessible. To a great extent, we have drawn on proposals for marketplaces from other disciplines, and the contribution of this paper is primarily investigation of the infrastructure needed for a marketplace with a large and diverse collection of data and data manipulation services used by customers from several communities of interest.

We suggest that the infrastructure cannot be considered in isolation; rather it is one component of a system, and it is the effectiveness of the total system which is to be optimized.

Three clusters of topics deserve special attention. First, it remains to be determined how spatial applications of realistic complexity can be built using the marketplace to source data and computational services. As the idea of a marketplace is a relatively novel approach to building applications, it is possible that it will enable innovative spatial applications. We see research on the infrastructure and on prototype applications as synergistic. Second, marketplace implementations of some well-known GIS operations might require the formulation of new algorithms. For example, in a single-site system, the cost of establishing a data set (by building a spatial index, for example) can be amortized over many operation executions. In a marketplace, where data might be delivered, processed and discarded, the cost of preprocessing is totally included in the costs of a single execution. This issue is similar to that present in other forms of distributed systems (Zhou, Abel, and Truffett 1997), and arguably a marketplace simply increases the value of this research theme. Finally, a marketplace designer should cater for inherent limitations of source and derived data due to error, imprecision, classification schemes, and so on. In some ways, the marketplace puts these issues in a sharper focus and offers a framework, through its registration mechanisms, for the recording of metadata to support interpretation of results. Because a marketplace allows the easier use of data from external providers and to conduct complex analyses, we regard it as especially important to assist customers to understand the limitations of their results.

The SMART model and the ACT-TAP prototype provide some confidence that a basic form of

marketplace infrastructure can be established. Clearly, progress to more comprehensive models and implementations for market infrastructures will require the solution of several research problems. The apparent similarity of some core aspects of the marketplace forms of these problems to those of multidatabase and workflow systems suggests that existing solutions from these fields can be borrowed, possibly with some adaptation, and so accelerate the availability of Spatial Internet Marketplaces.

**Acknowledgments** This work has benefited from discussion with many colleagues. We are particularly indebted to Stuart Hungerford, Ross Ackland, Dean Jackson, Dean Kuo and Robert Power for their expert contributions to the ACT-TAP prototype. We also acknowledge discussions with colleagues in other CSIRO groups and in the Queensland Department of Natural Resources. The ACTMAP databases were provided by the Land Information Centre, ACT Department of Urban Services.

## References

- Abel, D. (1997). Spatial Internet Marketplaces: A grand challenge? In *Proc. 5th Int. Symposium on Spatial Databases (SSD'97)*, LNCS. Springer-Verlag.
- Alonso, G. and C. Hagen (1997). Geo-opera: Workflow concepts for spatial processes. In A. Voisard and M. Scholl (Eds.), *Proc. 5th Int. Symposium on Large Spatial Databases (SSD'97)*.
- Bhargava, H. K., A. S. King, and D. S. McQuay (1995). Decisionnet: an architecture for modelling and decision support over the World Wide Web. In T. X. Bui (Ed.), *Proc. 3rd Int. Society for Decision Support Systems Conf.*, Volume 2, Hong Kong, pp. 541–550. Int. Society for DSS.
- Bhargava, H. L., R. Krishnan, and R. Müller (1996). Electronic commerce in decision technologies: a business cycle analysis. Technical report, Humboldt-Universität zu Berlin. SFB Discussion Paper, Sonderforschungsbereich 373.
- Bhargava, H. L., R. Krishnan, and R. Müller (1997). Decision support on demand: Emerging electronic markets for decision technologies. *Int. Journal Decision Support Systems* 1997(19), 193 – 214.
- Cameron, M. and D. Abel (1996). A problem model for decision support systems. In M. Kraak and M. Molenaar (Eds.), *Proc 7th International Symposium on Spatial Data Handling*, Delft, The Netherlands, pp. 3A25–3A36.
- Cattell, R. G. G. (Ed.) (1994). *The Object Database Standard: ODMG -93*. Morgan Kaufman.
- Dongarra, J. and E. Grosse (1987). Distribution of mathematical software by electronic mail. *Communications of the ACM* 30, 403–407. URL: <http://achille.att.com/netlib/index.html>.
- Flanagan, D. (1996). *Java in a Nutshell*. O'Reilly & Associates, Inc.
- Gruber, T. R. (1993). Toward principles for the design of ontologies used for knowledge sharing. In *Workshop on Formal Ontology*, Padua.

- Günther, O., R. Müller, P. Schmidt, H. Bhargava, and R. Krishnan (1996). MMM: a WWW-based method management system for using software modules remotely. Technical Report 32-1996, Humboldt-Universität zu Berlin. SFB Discussion Paper, Sonderforschungsbereich 373.
- Jeusfeld, M. A. and T. X. Bui (1997). Distributed decision support and organizational connectivity: A case study. *Int. Journal Decision Support Systems* 1997(19), 151 – 170.
- Lomet, D. (Ed.) (1993). *Special Issue on Workflow and Extended Transaction Systems*, Volume 16. IEEE Data Engineering Bulletin.
- OMG (1996). OMG RFP5 submission: Trading object service. Technical Report orbos/96-05-06, Object Management Group. Version 1.0. Available under URL: <http://www.omg.org/orbos/96-05-06.ps>.
- Open GIS Consortium (1997). The OpenGIS guide. URL: <http://www.ogis.org/guide>.
- Orfali, R., D. Harkey, and J. Edwards (1996). *The Essential Distributed Objects Survival Guide*. New York, U.S.A.: John Wiley and Sons Inc.
- Sarrat, D., G. Pierre, J. Harms, G. Richard, R. Oizel, and S. Vizzari (1995). Modelling of the distributed EO data handling facilities- plans for year 2000. Technical Report DUS-MGT-006-TR, European Space Agency Report. 93 pages.
- Schek, H. J. and A. Wolf (1993). From extensible database systems to interoperable between multiple databases and GIS applications. In D. J. Abel and B. C. Ooi (Eds.), *Proc. 3rd Int. Symposium on Spatial Databases (SSD'93)*, Number 692 in LNCS, Singapore, pp. 207–238. Springer-Verlag.
- Sheth, A. P. and J. A. Larson (1990). Federated database systems for managing distributed heterogeneous, and autonomous databases. *Computing Surveys* 22(3), 182–236.
- Zhou, X., D. Abel, and D. Truffett (1997). Data partitioning for parallel spatial join processing. In A. Voisard and M. Scholl (Eds.), *Proc. 5th Int. Symposium on Large Spatial Databases (SSD'97)*.