Evaluation of OGC Web Services for Local Spatial Data Infrastructures and for the Development of Clients for Geographic Information Systems

Leonardo Lacerda Alves, Clodoveu A. Davis Jr.

Instituto de Informática
Pontifícia Universidade Católica de Minas Gerais

1 Introduction

Interoperability is one of the most important challenges related to GIS. Through the last years, research on interoperability has evolved from the simple off-line exchange of standardized-format files, through the establishment of spatial data clearinghouses, and to the first initiatives in the treatment of semantic aspects of data. Practical interoperability, however, is still hampered by the need to agree on standards, and to develop appropriate tools and methods.

The Open Geospatial Consortium (OGC) has proposed a number of standards to that respect, with the intention of promoting interoperability through the use of services [25]. However, OGC's definition of Web-based services for spatial information predates the World-Wide Web Consortium's definition of the Web service architecture [34]. The necessary adjustments between OGC's and W3C's proposals are still under way.

Meanwhile, we observe that a number of difficulties arise when someone tries to effectively implement the interoperability-through-services approach. Issues regarding fault tolerance, server-independent implementation, delayed-time transactions, privacy, and others reflect the need for further study and discussion. In fact, studies about the conformance of OGC standards to the distributed systems development are still scarce.

This paper extends the work of Alves and Davis Jr [2] and it discusses the current status of service-oriented architectures as applied to interoperable GIS, or, more specifically, to the implementation of local spatial data infrastructures (LSDI). Most existing spatial data infrastructures refer to regional or country-wide data, while LSDI deals with a more complex and rich set of geographic data [24, 28]. Thus, LSDI involve a wide variety of services, while also dealing with users of a wide range of devices, such as PDAs, cell phones and personal computers.

For this discussion, we defined a real-world use case based on an urban context, and developed a services prototype following OGC's abstract model. We used this prototype to assess the engineering guidelines for the server and client development, according to the viewpoints established by the OGC Reference Model [25]. To solve some of the limitations and issues that we have identified, we proposed and developed special infrastructure services which illustrate some deficiencies of OGC specifications. Nevertheless, we do not imply here that such services should become part of the standard. Using the proposed services, it is possible to enable asynchronous communication between clients and services, to access data provided by clients, and to improve on critical points of the services-oriented architecture, such as recovery from failures, dynamic service chains creation, and others, while staying within the OGC Reference Model.

This paper is organized as follows. Section 2 presents concepts about Spatial Data Infrastructures (SDI), in general, and Local Spatial Data Infrastructures (LSDI), in particular, including the ideas behind configuring SDI and LSDI as services. Section 3 introduces our discussion as to the required functionality of a local SDI, and the way to achieve that using the aforementioned infrastructure services. Finally, Section 4 presents our conclusions and indicates some research directions from the concerns presented here.

2 Related Work

2.1 Spatial Data Infrastructures

Spatial Data Infrastructures (SDI) constitute a set of policies, technologies and standards that interconnect a community of spatial information users and related support activities for production and management of geographic information [26]. The idea behind SDI involves avoiding redundant effort and reducing production costs for new and existent datasets through the sharing of resources. In order to achieve this, it is very impor-

tant that the various partners have convergent interests, agree on common rules and are allowed to make use of data or information produced by others.

Geographic information from one or several partners can be consolidated and thus form important resources for high-level decision-makers. In this case, SDI can be seen as a set of building blocks, as defined by Rajabifard et al [28], in which SDI hierarchies are built through the exchange and consolidation of information from corporate and local levels, to regional and global levels. In this hierarchy, lower levels provide detailed information that helps in the consolidation of the upper, more general, levels [27, 17, 21].

Guiding the technology standardization and, consequently, defining the key elements for spatial data infrastructures, a number of standards were proposed by the OGC, through a framework called OGC Reference Model [25]. This framework has been implemented successfully in state, national and regional scales [12, 13, 30], but reference cases of local SDI are still scarce. Furthermore, most of them are not OGC-compliant [18, 22].

Local SDI has the potential of bringing a numerous group of users together, each of which with distinct needs [9]. This characteristic is in part responsible for an increased level of complexity in local SDI development and deployment [5, 23]. Indeed, LSDI is valuable to all of the other SDI levels as a detailed information source, and the implementation of actual data sources in the local level should evolve simultaneously with, and guided by, the new demands for Geographic Information (GI) from a new range of users, such as business travelers, citizens, small companies and others [23].

SDI can be implemented by chaining services of different sources [1] and integrating software components [15] that can be found in geoportals [20]. The emphasis on services has increased since the emergence of Web services and of the Service-Oriented Architecture (SOA) in the field of distributed systems development. This led Bernard and Craglia [4] to propose a new translation for the the SDI acronym, to mean *Service-Driven Infrastructures*.

2.2 Web Services and OGC Services

Web services are usually seen as Web-based enterprise-wide or inter-organizational applications that use open standards (mostly based on XML) and transport protocols to exchange data with clients, thus forming a loosely-coupled information systems architecture [11, 14]. Web services, as specified by the World Wide Web Consortium (W3C) [34], use the Hy-

pertext Transfer Protocol (HTTP) in the application layer. This protocol only allows the client to perform synchronous calls. This characteristic represents a problem when dealing with delayed-time transactions and call resume.

Some initiatives try to emulate asynchronous calls in Web services through the use of *listeners* that receive responses and forward them to the requester [29], usually over non-HTTP protocols [7]. Other strategies involve the use of the electronic-mail protocol, SMTP [8]. However, as far as we know, no pattern enables asynchronous communication using only Web service standards.

Regarding geographic information specifically, the OGC Web Services specification (OWS) [33] describe a set of functionalities and components which support standardized information exchange. Among the OGC services, there are asynchronous communications only for sensor services [6] such as the Web Notification Service, but the communication protocol is not specified in the standard.

In this work, we are particularly interested in LSDI, i.e., a network of clients and services that deal with an urban context and typical urban applications. The urban-specific OGC proposed platform is called Open Location Services (OpenLS) [19], in which services such as directory (yellow pages service), routing services, geocoding services, and others are grouped. A proposal for urban services of interest as LSDI elements can be found in [10]. The variety of possibilities for using such urban services to build compact and useful geographic information-based applications is staggering. The next section presents one of them.

2.3 Use Case: a Consumer Travel Assistance Application

The OGC Reference Model [25] defines a scenario entitled ``consumer travel assistance'' [p. 83] through which a GI consumer uses a mobile client to (1) get its actual position or location; (2) get the destination address, given a telephone number; (3) get a location, given the destination address; (4) get a route between the origin and the destination; (5) determine the traffic, weather and road conditions along the way; and (6) obtain real-time travel advice. Of course, such a task potentially requires a large amount of geographic data, possibly from various sources. This is something that is well above the capabilities of a typical mobile client, thus requiring a new computational model. In this paper, we extend this scenario and use it to implement OpenLS platform clients and an OWS framework, in order to evaluate the OGC Reference Model for LSDI client development.

Our modified scenario involves two basemap services, one routing service, one geocoding service, one directory service for telephone numbers, one directory service for companies, one public transportation service, one public-services service, two emergency services, one gateway for personal location, and two kinds of clients (thin and thick).

While designing LSDI clients to deal with such a variety of services, we have identified some limitations of OGC specifications, more specifically in important engineering requisites such as service availability assurance, recovery in case of failures, privacy control, communication cost reduction and transparent support for the peculiarities of different kinds of client devices.

Even though we have not yet fully developed this scenario as to its actual functionality, we have worked on determining and implementing the ideal communications protocols for each situation, so that some of the limitations inherent to OGC's architecture can be adequately solved beforehand. A prototype was implemented, based mostly on Web service aspects, and leaving the many related GIS issues for the near future. The goal of this prototype is to make sure the required transactions are adequately designed, and all communications issues are solved. The next section presents and discusses the alternatives we have conceived and implemented to that effect.

3 Innovative Services for LSDI

Given the anticipated needs of urban GI applications and the main limitations of GI Web services, some novel infrastructure services are useful to improve the LSDI clients development and the LSDIs themselves.

In the next three subsections, we present the infrastructure services we conceived and developed. The first, called *Data Exchange Service*, is used for persistence maintenance between clients and servers. The second, called *Client Access Service*, is used for information exchange and access control to clients by servers and other clients. The third, called *Transaction Control Service*, is used by clients and servers to improve engineering capabilities such as failure recovery, dynamic service chains creation, workflow definition, and others. The fourth subsection presents details about the prototype implementation and its analysis.

3.1 Data Exchange Service

The Data Exchange Service (DXS) supports interactions between clients and servers (client to server communication), different servers (server to server communication), and different clients (client to client communication) as a workspace where data may be freely stored and retrieved. The objective of this service is to reduce the volume of interactions between clients and servers, and to minimize the connection costs in service invocations and data retrieval, even when failures occur.

Figure 1 shows an example of the DXS in action. A service A, replicated in A' and A" for redundant availability, as it is invoked by a client. Initially, the request parameters are sent to a DXS (1). Next, the ordering of the procedures is defined by the client, in the form of a workflow (2). An invocation process (3) reaches service A, or resorts to services A' or A" in case of failure. In any case, the parameters are recovered from the DXS (4), and the results are stored in that server for future retrieval by another service, or by the client (5).

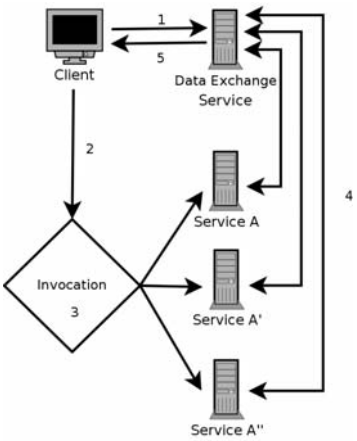

Fig. 1. Data Exchange Service in the invocation of a replicated service

The Data Exchange Service works in three different ways. The first way effectively establishes *asynchronous communication* between the client and the target service. The DXS mediates the communication, so that the client does not have to wait online for a response from the target service. When the processing is through, the client receives a message from the DXS, and is then allowed to retrieve the results.

The DXS also works as a *temporary repository* for intermediate responses in a service chain. Intermediate results are kept by the DXS for the

benefit of services along the chain, but the client is only allowed access to the final results, as they become ready.

Furthermore, the DXS supports *failure recovery*, since it can keep information on the status of a service chain, along with intermediate results. With this, recovery and continuation from the client side is possible by re-invoking any service that fails, using the stored parameters, thus avoiding reinitialization of the entire chain.

To understand and to specify how the Data Exchange Service works, we defined the client and service invocation patterns using an UML (Unified Modeling Language) sequence diagram. Figure 2 presents the diagram.

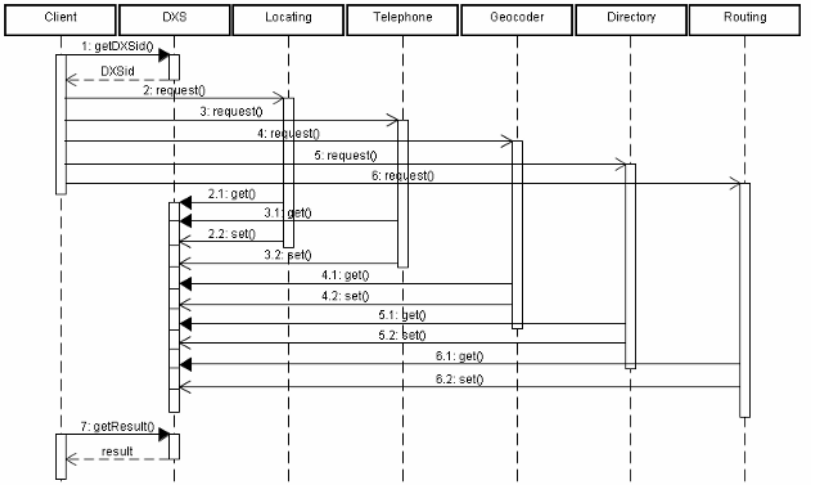

Fig. 2. UML Sequence Diagram for a System using DXS

Messages 1 and 7, between the client and the DXS, are the same generic processes 1 and 5 shown in figure 1, which initiate the processing and collect the final results, respectively.

Messages 2, 3, 4, 5, and 6 invocate all needed services at the beginning of the process, when the client stops and the services keep running. Next, as the *Locating* and *Telephone* services can be executed in parallel, messages 2.1, 2.2, 3.1, and 3.2 can occur in any order.

Finally, messages between 4.1 and 6.2 occur sequentially. Notice that all intermediate data exchange would pass through the client if the DXS was absent. However, using this service, such intermediate response traffic is avoided. Most of the service chain can be invoked in parallel to reduce client work, and the responsibility for most invocation callbacks is transferred to the DXS.

In the highest conceptual level, the Data Exchange Service has thus the function of intercepting service responses and forwarding them to other services (including other DXS) or to the client. In this sense, the communication costs for thick and rich clients remains the same, while costs for thin clients fall drastically. The length of the lifeline remains the same, regardless of the involvement or not of the DXS. Therefore, the DXS offers advantages only when the client has energy limitations or it is slower than the DXS server.

It would be possible to implement DXS-like persistence through simple Web services, but a standardized interface that functions as an infrastructure service is important to ensure the independence between clients and OGC-services providers.

As presented, the DXS may suffer from a number of security issues, such as unauthorized access to data from third parties. However, protocol enhancements can ensure authorized access, so that only the participating services can retrieve intermediate data, and not even the client is allowed access to privileged or confidential information [3].

3.2 Client Access Service

As mentioned in the previous section, the Data Exchange Service enables asynchronous communication between the client and the target service, notifying the client when the results are ready to use. However, asynchronous calls are not supported in W3C Web services using HTTP [7, 29]. Among OGC services, only the Web Notification Service implements asynchronous answers, even though it also does not use a HTTP interface [6].

```
transactionID = service.do(params);
```

Code. 1. A call that return a transaction ID

In an urban context, asynchronous services are often necessary, working as delayed-time transactions for long service chains. They are also necessary in particular applications, such as using servers as sensors to send information to clients without counting on connection-oriented communication.

```
while (!service.isReady(transactionID))
    self.sleep(1000);
result = service.get(transactionID);
```

Code. 2. Getting the service result

Even though normally a client does not have a valid IP (Internet Protocol) address, it is able to access HTTP resources. However, without an IP address it is inaccessible from other clients or services, receiving data only during its request connections. We propose an alternative to reach clients without an IP address. A client can invoke a service using a method that returns a transaction Id, but not the final result. This is illustrated in code 1.

```
transactionID =
    service.do(responseMethod, responseURI,
                                        params)
```

Code. 3. The client define a method for receive a notification when the result is ready

In a second moment, the client invokes the service again through a method that returns the final results, giving the transaction Id, as illustrated in code 2.

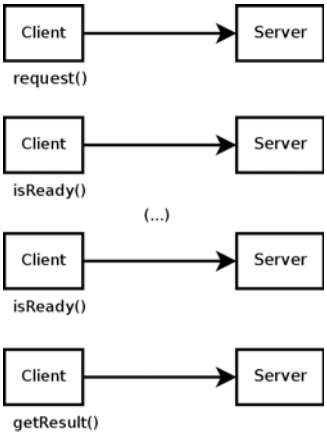

Fig. 3. Client without valid IP address using communication based on HTTP

However, getting results is only possible when the processing has been concluded, and the client uses up network and processing resources to poll the service for a response. Figure 3 presents this communication pattern.

If the client has a valid IP address and implements the functionalities of HTTP servers, we have a different situation. The client waits for responses through a DXS-like service, precisely until it receives a response from the service, informing that the results are ready. Code 3 illustrates a call that specifies a response method (HTTP, SMTP, SMS, etc) as part of its pa-

rameters, along with an URI (a address through which the client expects the notification when the process had ends).

```
while (self.waiting(transactionID)) {
    // do nothing
}
result = service.get(transactionID);
```

Code. 4. Waiting for Notification from the Gateway or Other Service

Figure 4 shows the described communication protocol, and the code 4 illustrates the last call from the client to the service when de service processing becomes ready.

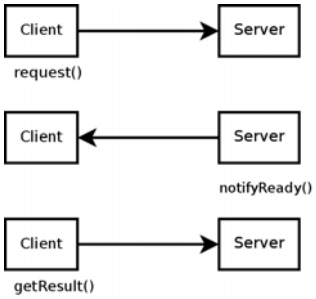

Fig. 4. Client with valid IP address using communication based on HTTP

The last discussed pattern is dependent on a local gateway service such as a DXS, and it is adopted when the client does not have a valid IP address, or when the client is protected behind a firewall, or when the client does not want to be identified. In this case, the client can access the target service directly or indirectly, but the notification process always occurs between the chosen gateway and the client, as illustrated in figure 5. The client implements the same code presented in code 3 and in code 4. We call this approach the *Client Access Service* (CAS).

The Client Access Service offers means to use clients as sensors for others, and means to enable the client to work as a Data Exchange Service provider (specially in the case of thick clients). In both cases, the client is able to provide some information to other clients and services and to support a number of concurrent processes, with adequate security and privacy constraints.

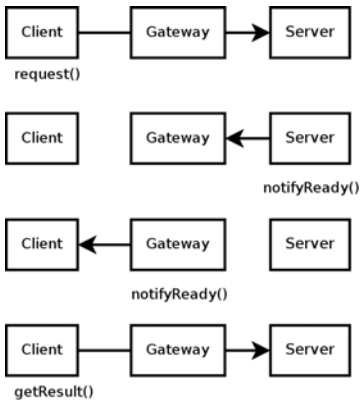

Fig. 5. Client without valid IP address using a Gateway as mediator

3.3 Transaction Control Service

Through the two services previously presented, a GI client can improve its processing capabilities and make the access to geographic information easier. However, it is hard to develop a generic client, suitable to different LSDIs. Furthermore, some capabilities, such as fault tolerance and engineering transparency, are actually the server's responsibility, but shifting this responsibility towards the client is sometimes convenient, specially for mobile clients in an urban context.

When traveling, the client should be independent of specific service chains. In this case, when a client finds itself in a distant city, it can load and execute a chain of different services to obtain local information.To do this, services can be readily available within chains, but it would be better if each service was available individually and listed in a public catalog service. The selection and chaining of services can occur dynamically whenever the client needs to access them.

The Transaction Control Service (TCS) performs transformations in generic chaining code in order to generate a fully functional service chain, suited to a particular context. The generic code defines a basic workflow that is to be followed by the client during the execution process. That same basic workflow can serve as a template for other service chains, counting on the Data Exchange Service for persistence and to store the processing state.

The codes 5 and 6 present the result of using the TCS for the conversion of an abstract and generic codes into client-specific service chains in the particular context of a routing application. The main objective of this ser-

vice is to ensure that neither client nor service development are dependent on the local context, i.e., on peculiarities of the services that are available at each different location. The code 5 is presented as a pseudocode, albeit the code is actually specified using XML.

```
// Client software in control
target = client.from();
myLocation = client.from();

// Concurrent commands without user interaction
myLoc = location(myLocation,Tdirectory) |
p=pointsOfInterest(location(target),Tdirectory);

// Client software in control
client.to(p);
p2 = client.from();

// Non-interactive commands
query(Troute,myLoc, p2, location(target));
```

Code. 5. Abstract code, independent of technology and services

The generic code makes changes on the previous code when the Data Exchange Service and the geographic services are introduced in the algorithm and the requesites of the current LSDI are taken into account.

```
// Concurrent commands without user interaction
myLocAd = cityHall.location(myLocation,
          Tdirectory, dxs);

cityHall.location(target, dxs.setItem(0));
teleCo.location(target, dxs.setItem(0));
coAssociation.location(target, dxs.setItem(0));
tourismGuide.location(target, dxs.setItem(0));

pAd = cityHall.pointsOfInterest(dxs.getId(0),
      Tdirectory, dxs);
```

Code. 6. Part of specific code adapted to a LSDI and thin client

Finally, the code 6 should implement an algorithm as efficient as possible to the client software, which may be designed according to the needs of specific clients and users for the current LSDI.

With these services, LSDI clients can be implemented based on informational and computational aspects, avoiding the introduction of technological aspects into client code. Thus, thick-, rich-, and thin-clients can be implemented transparently, and their particular constraints are addressed by a generic workflow that changes easily when the client moves to another place or has its capacity improved.

3.4 Prototype Implementation and Analysis

We have implemented Web services for GIS as specified by the OGC, following the Abstract Model specification. The distribution patterns are (a) 1 Client to 1 Server (or known provider) to *n* Servers, where there is a provider which mediates resources from others, and (b) 1 Client to *n* Servers, where the client performs all of the tasks related to dealing access to Web services and geographic information processing.

The first pattern (a) made it easier to implement clients and servers with important engineering requisites, but introduces much dependency on known providers, in situations where no information about their resources and about external resources is available.

However, to enable client access to information about available resources, it is necessary to abandon the “known provider” role. This led us to implement the pattern (b), 1 Client to *n* Servers. This pattern increases the costs of processing and traffic of intermediate data, because the client requests data to several services, and once it receives their responses it usually performs some operations and submits a large volume of data to other services in the service chain. However, this pattern enabled us to perceive and to define some engineering requisites on the client side.

Next, we separated design issues from technologic ones, and we grouped them into three groups, which constitute the services presented in the previous subsections.

By executing the features grouped into the Data Exchange Service, the intermediate data traffic in non-interactive processing was avoided on the client. In addition, in case of failure no previously processed data was lost. Nevertheless, we noticed no significant effect in interactive processing when there are multiple inputs during the service chain execution.

Through the Client Access Service, the delay and communication costs were reduced by avoiding the use of the network by the client when it needs to poll the service about its readiness before it really becomes ready.

Finally, we created a subset of operations and defined service chains using XML. The Transaction Control Service was then used to convert the XML data, given the client's requisites and characteristics, to a service chain which uses the above infrastructure services, automatically.

4 Conclusion

4.1 Results

This work evaluated the engineering aspects of OWS specifications and the main OGC services in the context of urban applications. Through these applications, we identified some implementation constraints that are characteristic of GI systems, such as non-standardized fault tolerance mechanisms, clients that are strongly dependent of providers, and others.

To achieve this, we implemented a prototype based on the services abstract model for a real-world use case, and tested the usefulness of OGC standards to LSDI, specifically considering LSDI client development. As the objectives of standardization include to guide uniform application deployment and services interoperability, distribution transparencies, such as access, failure, and persistence transparencies, become essential.

Since the availability of client software contributes to LSDI diffusion and implementation, we have also developed new infrastructure services, which facilitate the development of LSDI clients and interoperable GIS with urban characteristics. Thus, the new services constitute a synthesis of missing features in the standard technologies, and this prototype allows the exploration of other issues related to OGC standards and urban GIS applications in future research efforts.

4.2 Main Contributions

Through the Data Exchange Service, clients – either thin, rich, or thick – can be transparently developed without previous distinction. There are additional advantages for thin and rich clients, which have greater limitations of energy, storage, communication, and processing capacity.

The DXS replaces the persistence function of service providers and local storage with a third-party neutral service, through which the services chain exchanges parameters and results of its inner processing.

The establishment of the services chain is improved by two solutions. The first one is focused on the client, while the second one benefits from the communication capabilities among servers.

In the first alternative, the services chain orchestrated by the client defines the providers before and during the invocation of services. Then, the client becomes responsible for fault tolerance and it can choose alternative services according to established parameters. Additionally, when the client travels or in case of emergencies, the whole service set may be changed.

Nevertheless, the services chain may be orchestrated by a W3C- or OGC-like service, that calls others. Normally the client does not participate in service selection for the composition of a chain. It can only refuse a provider by selecting another. However, even this situation benefits from a shift of responsibility from the server to the client. The insertion of the Data Exchange Service into server workflow reduces the reissuing of requests in case of failures (specially if the client is thin), and increases the parallelism level of processing.

Therefore, we proposed a set of new services, which facilitate the development of GI clients for LSDI-based systems, and contribute to improve the diffusion and implementation of LSDIs.

4.3 Future Work

We identify three main directions for future work. First, we discuss the potential use of clients as information providers. Next, we propose some engineering enhancements to the development of services. Finally, we envision the possibility of developing new LSDI services, enhancing the prototype presented in this paper.

The GI client, viewed as a server of parameters for the service chain, can potentially become the server of various kinds of information. Information perceived directly by the user, such as traffic status [31], along with previously collected data, perceived quality of information, ontologies about its interests, geographic position, and others, can be passed along to other users through services. Possible applications to this are better traffic management using vehicles as real time sensors, more precise route planning by information exchange with other GI users, quality-of-service control, among others. For traffic status monitoring, information may be exchanged between these “sensors” by active client-server connections (in the case of wireless sensor networks with limited energy capacity), or peer-to-peer connections (in the case of wireless sensors networks without energy restrictions, i.e. ad-hoc networks constituted by in-transit vehicles), with new application possibilities, such as the dissemination of warnings or instructions. Peer-to-peer connections, when energy is not an issue, should improve the response time of applications [32], because previously collected data of other users is available. Geocoding, routing and

locating services can then be adapted for mobile geographic objects, such as vehicles (public transportation, emergency, particular vehicles), or for personal locating (for instance, using cell phones). However, additional studies about these applications are required, in order to deal with concerns such as security and privacy. For effective privacy control, privacy parameters could be dynamically negotiated between clients and servers.

Clients, functioning as information providers, can also be the source of a measure of quality of service. Client feedback can be used to establish perceived quality, and to identify the quality parameters most valued by a user group. Ontologies on perceived quality parameters can then be developed from these observations.

As to engineering, fault tolerance parameters can be defined and negotiated according to availability contracts that should be automatically verified through a search into the service chain and evaluation of common failure points between main and substitute services. This is specially important in emergency response, critical services applications, and others.

Another engineering point regards workflow descriptions. In some cases, it is helpful that services be localized and dynamically injected into the original workflow, as in the case of the routing service, when routes cross regions not covered by the loaded data sources. Therefore, additional improvements should be made to workflow description mechanisms.

Davis Jr and Alves (2005) [10] propose several services for LSDI, considering the specific demands for urban geographic applications. We think that further services related to this level of infrastructure can be proposed and investigated, using the prototype presented here as a basic framework.

Finally, the creation of a more complex LSDI prototype development environment is important to enable observations and experimentation on current and upcoming OGC standards. Such a framework should be based on the OGC's implementation specifications rather than on an abstract specification model, as we did. Thus, it can be used to develop further formal studies about several concerns (privacy, performance, security, and so on) with real-world problems. It would also be important to build such a prototype development environment using free or open-source software, so that the community involved in spatial data infrastructures can benefit, while being able also to contribute to it.

Additionally, we think the standardization process, as it is today, can produce social problems if applied to LSDI. Therefore, according to [16], efforts to adapt to older to new standards can cause fragmentation, complexity and heterogeneity rather than solve this problems. An important research question should be which factors contribute to success and failure of standardization efforts on LSDI.

References

[1] Alameh, N. (2003). Chaining geographic information web services. IEEE Internet Computing, 7(5):22–29.

[2] Alves, L. L. and Davis Junior, C. A. (2006). Interoperability through Web Services: Evaluating OGC Standards in Client Development for Spatial Data Infrastructures. In VIII Brazilian Symposium on Geoinformatics Proceedings, pages 193–208. INPE.

[3] Belussi, A., Bertino, E., and Catania, B. (2004). An authorization model for geographical maps. In GIS'04 Proceedings, pages 82–91. ACM.

[4] Bernard, L. and Craglia, M. (2005). SDI – From Spatial Data Infrastructure to Service Driven Infrastructure. In Research Workshop on Crosslearning between Spatial Data Infrastructures and Information Infrastructures.

[5] Bishop, I. D., Escobar, F. J., Karuppannan, S., Suwarnarat, K., Williamson, I. P., Yates, P. M., and Yaqub, H. W. (2000). Spatial data infrastructures for cities in developing countries: Lessons from the Bangkok experience. Cities, 17(2):85–96.

[6] Botts, M., Robin, A., Davidson, J., and Simonis, I. (2006). Sensor Web Enablement Architecture, OGC document 06-021r1. OGC, Wayland.

[7] Brambilla, M., Ceri, S., Passamani, M., and Riccio, A. (2004). Managing asynchronous web services interactions. In Web Services, 2004. Proceedings. IEEE International Conference on, pages 80–87. IEEE.

[8] Chung, S., Pan, J. R., and Davalos, S. (2006). A special web service mechanism: Asynchronous .NET web services. In Telecommunications, 2006. AICT-ICIW '06. International Conference on Internet and Web Applications and Services/ Advanced International Conference on, pages 212–212. IEEE.

[9] Davis Junior, C. A. (2005). Considerations from the Development of a Local Spatial Data Infrastructure in Brazil. In Research Workshop on Cross-learning between Spatial Data Infrastructures and Information Infrastructures.

[10] Davis Jr, C. A. and Alves, L. L. (2005). Local spatial data infrastructures based on a service-oriented architecture. In VII Brazilian Symposium on Geoinformatics Proceedings, pages 84–89. INPE.

[11] Davis Jr, C. A. and Alves, L. L. (2007). Geospatial Web Services. In Encyclopedia of Geographic Information Systems. Springer-Verlag, Berlin.

[12] Deegree (2006). Deegree Project (WMS, WFS, WCS, iGeoPortal, deeJUMP). Lat/Lon GmbH http://www.deegree.org.

[13] Demis (2006). Demis Web Map Server. Demis http://www.demis.nl/.

[14] Ferris, C. and Farrell, J. (2003). What are web services? Communications of the ACM, 46(6):31.

[15] Granell, C., Gould, M., and Ramos, F. (2005). Service composition for SDIs: Integrated components creation. In: Proceedings of II Internation on Geographic Information Management.

[16] Hanseth, O., Jacucci, E., Grisot, M., Aanestad, M. (2006). Reflexive Standardization: Side effects and complexity in standard making. MIS Quarterly. Vol. 30. Special Issue, pages 563–581.

[17] Jacoby, S., Smith, J., Ting, L., and Williamson, I. (2002). Developing a common spatial data infrastructure between state and local government: An Australian case study. IJGIS, 16(4):305–322.
[18] Kumar, P., Singh, V., and Reddy, D. (2005). Advanced traveler information system for Hyderabad City. Intelligent Transportation Systems, IEEE Transactions on, 6(1):26–37.
[19] Mabrouk, M. (2004). OpenGIS Location Services (OpenLS): Core Services, OGC document 03-006r3. OGC, Wayland.
[20] Maguire, D. J. and Longley, P. A. (2005). The emergence of geoportals and their role in spatial data infrastructures. Computers, Environment and Urban Systems, 29(1):3–14.
[21] Man, W. H. E. D. (2006). Understanding SDI; complexity and institutionalization. IJGIS, 20(3):329–343.
[22] Mansourian, A., Rajabifard, A., Zoej, M. J. V., and Williamson, I. (2006). Using SDI and web-based system to facilitate disaster management. Computers & Geosciences, 32(3):303–315.
[23] Masser, I. (2005). Some priorities for SDI related research. In Proceedings of the FIG Working Week and GSDI 8: From Pharaohs to Geinformatics, Cairo, Egypt, page 11. FIG.
[24] Nedovic-Budic, Z., Feeney, M.-E. F., Rajabifard, A., and Williamson, I. (2004). Are SDIs serving the needs of local planning? case study of Victoria, Australia and Illinois, USA. Computers, Environment and Urban Systems, 28(4):329–351.
[25] Percivall, G. (2003). OGC Reference Model OGC 03-040. OGC, Inc.
[26] Phillips, A., Williamson, I., and Ezigbalike, C. (1999). Spatial data infrastructure concepts. The Australian Surveyor, 44(1):20–28.
[27] Rajabifard, A. and Williamson, I. P. (2001). Spatial data infrastructures: Concept, hierarchy, and future directions. In Geomatics'80, page 11. Tehran, Iran.
[28] Rajabifard, A., Williamson, I. P., Holland, P., and Johnstone, G. (2000). From local to global SDI initiatives: a pyramid of building blocks. In IV Global Spatial Data Infrastructure Conference Proceedings.
[29] Ruth, M., Lin, F., and Tu, S. (2005). A client-side framework enabling callbacks from web services. In Web Services, 2005. ECOWS 2005. Third IEEE European Conference on, page 12. IEEE.
[30] Skylab (2006). J2ME OGC WMS Client. Skylab http :// www . skylabmobilesystems . com / en / products / j2me wms client . html.
[31] Tao, X., Jiang, C., and Han, Y. (2005). Applying SOA to intelligent transportation system. In Proceedings of the 2005 IEEE International Conference on Services Computing, pages 101–104. IEEE.
[32] Wang, H., Zimmermann, R., and Ku,W.-S. (2005). ASPEN: An adaptive spatial peer-to-peer network. In GIS'05 Proceedings, pages 230–239. ACM.
[33] Whiteside, A. (2005). OpenGIS Web Services Common Specification, OGC document 05-008. OGC, Wayland.
[34] World Wide Web Consortium (2004). Web Services Architecture W3C Working Group Note (Feb. 11 2004). W3C. www.w3.org/TR/2004/NOTE-ws-arch-20040211/.