Proceedings of the FIDJI International Workshop on the Scientific Engineering of distributed Java Applications. 2002, p. 33-46.

A Java Coordination Tool for Web-service Architectures: The Location-Based Service Context

P. Álvarez, J.A. Bañares, P.R. Muro-Medrano, J. Nogueras, and F.J. Zarazaga

Department Of Computer Science And Systems Engineering
University Of Zaragoza
María de Luna 3, 50015 Zaragoza (Spain)
{alvaper, banares, prmuro, jnog, javy}@posta.unizar.es
http://iaaa.cps.unizar.es

Abstract. The use of open technologies and standards have made easier the integration of Web services into end-applications. These interoperable services have been organized on distributed architectures over Internet in accordance with shared functional principles. But these Web-service architectures have not resolved the distributed computing difficulty in "gluing together" multiple and independent Web services. This paper presents an approach based on Java technology and Internet standard protocols and data formats for resolving coordination problems among Web services. Interaction models based on distributed events over HTTP are supported for providing the required coordination functionality. Cooperation problems and their solutions have been studied in the prototypical context of Location-Based Services.

Keywords: Web-service architectures, distributed service cooperation, Internet, Java and JavaSpaces Technologies

1 Introduction

Nowadays nobody doubts the Internet has become the most important global network infrastructure. Many companies are enclosing as software services their traditional computing tasks or introducing new tasks to connect them to Internet at a rapid pace searching new promising opportunities. However, this growth of services over the network has been faster than the formal efforts to agree on service-oriented architectures [6] and to identify the necessary support for enabling these distributed services to work together harmoniously [16].

First formal steps have progressed around the interoperability among systems (any Web service can interact with any other Web service [18]). This way, SOAP (http://www.w3.org/TR/SOAP/) has become a de facto standard for Web-service messaging and invocation, and for solving the problems of converting data between traditional distributed platforms such as CORBA, DCOM or EJB [19]. Additionally, many standardization initiatives have arisen in specific

Proceedings of the FIDJI International Workshop on the Scientific Engineering of distributed Java Applications. 2002, p. 33-46.

research areas for defining open, ubiquitous and interoperable service interfaces, such as the area of the Location-based Services (LBS).

LBS extend the spatial processing capabilities of the Geographic Information Services (GIS) integrating wireless communications, location data and Internet technology [25, 12]. In this context, two well-positioned organizations have emerged as the drivers of the LBS interoperability: LIF (Location-Interoperability Forum, `http://www.locationforum.org/`) and OGC (Open GIS Consortium, `http://www.opengis.org/`, and its Open Location Service Initiative (OpenLS), `http://www.openls.org/`). Both are promoting and defining standard interfaces for a collection of wireless, Location and GIS services for providing the required LBS functionality [2, 26, 29]. These public interfaces make easier the integration through Internet of these distributed services into end applications as individual computing entities, but in an isolated way, they have a very limited functional value. Therefore, LBS context may be considered as a prototypical technological-context where it may be evaluated the impact of the integration of industrial web-centric standards over the development of distributed applications.

Once services and their interfaces have been described, it arises the necessity of establishing an organization for supporting their use, interactions and automated discovery. This organization must define an architecture that provides a framework for making easier the collaborative work among the services and the access to them [14]. Past architecture experiences such as [24, 31] could help us to define these future ones: problem in the use of the object technology in large-scale applications when it must be combined and recombined [23], and in building of component-based frameworks [21]. Besides, in order to integrate services implemented with different computational models, services must cooperate and to be ensemble over this architectural vision in an orthogonal way to their computing tasks [8], allowing to exploit the true value of services beyond independent computing entities.

This work presents a coordination Web-service for distributed architectures over Internet which has been used inside the LBS context as a prototypical domain. This support service has been implemented in Java so it could be used independently of the hardware platform and the operating system on which it is being executed. Internet is a distributed environment where many hardware and software configurations can be found (`http://leb.net/hzo/ioscount/data/r.9904.txt`). Therefore, Java as programming language of Web services, is the key for achieving the required platform portability and independence. Besides, it provides a number of built-in networking capabilities that make it easy to develop Internet-based and Web-based applications [10]. A more detailed description of our technological evolution towards this approach may be found in [4].

The paper is structured as follows. Section 2 presents a description of services provided by standards that constitute the LBS framework. It is shown in a succinct way the underlying conceptual model, the hierarchical levels of functionality, and the found problems in order to orchestrate these services. Section 3 justifies the adopted coordination-approach based on JavaSpaces technology

Proceedings of the FIDJI International Workshop on the Scientific Engineering of distributed Java Applications. 2002, p. 33-46.

and standards to develop web-centric solutions from XML and HTTP. Section 4 shows design and implementation details. Section 5 reviews the benefits of the proposal. Finally, future work and conclusions are presented.

2 A Web-Service Architecture for Providing LBS Functionality

2.1 Conceptual Model of Architecture

In general, a Web-service architecture is composed by a collection of services that are organized according to any functional aspects. An example of this kind of distributed architectures are the LBS frameworks, which integrate GIS and Location Services [26, 28, 32]. According to the functional aspects related to this LBS context, a conceptual model of architecture has been defined.

The proposed model is hierarchical and organized in relation to the level of intensity of data processing involved. Three functional levels of services have been identified:

- Data Management, which is responsible for data storage and recovery.
- Data Processing, which generates new data from raw data.
- Data Analysis, which provides high-level functionality from generated data by the lower levels.

It is important to underline that requirements of levels are not independent in this model. Processing level requires raw data storage, and analysis level is built on the lower levels.

2.2 Building a LBS Framework on the Basis of Web Services

The presented architectural model has been the conceptual base for the development of a LBS framework whose functionality may be integrated into end-applications through Internet (see fig. 1), such as ERP or CRM systems [4]. Required services are organized according to the proposed functional levels and built according to the Web-service philosophy: their operations are provided through a standard, published interface to ensure interoperability, and are accessible via ubiquitous Internet protocols and data formats, such as HTTP and XML.

The Data Management level is the base of the proposed architecture. Its services must be able of providing the necessary support for the storage and recovery of geodata. LBS frameworks require a wide variety of geodata: georeferenced maps; location descriptors such as street addresses, roads, place names or telephone numbers; sensor data such as immediate locations of mobile devices; or more specific data of the LBS context, such as traffic conditions or road repairs. A collection of services has been implemented for fulfilling these requirements. For example, basic GIS services: Web Map Server (WMS), for

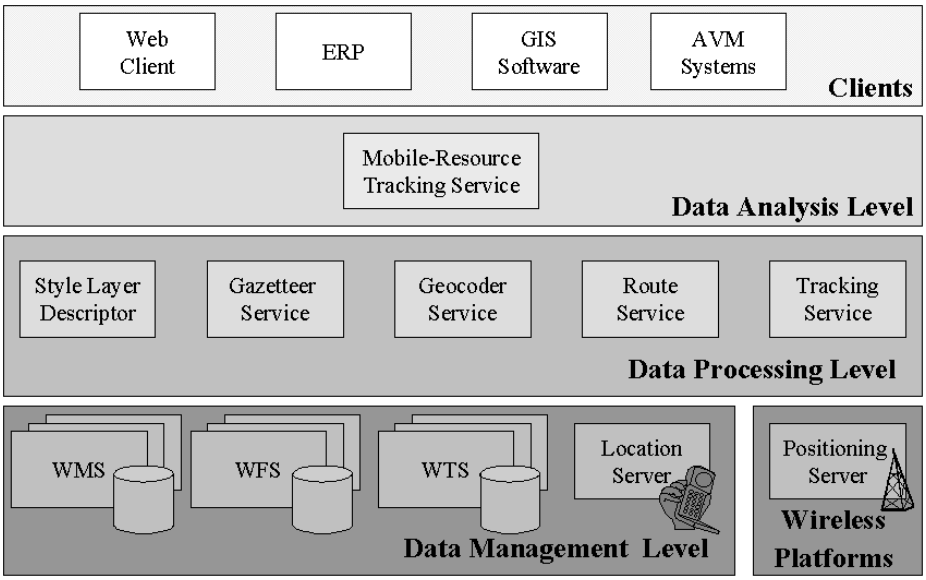

Fig. 1. LBS Web-service architecture

visualizing digital maps on the Internet as rendered raster data [3, 13]; Web Feature Server (WFS), for storing, spatial and non spatial querying and discovering geographical features, such as the previously presented location descriptors [1]; and Web Traffic Server (WTS) [26], for providing traffic conditions of a specific region of interest. Interfaces of these services have been developed following the specifications proposed by the Open GIS Consortium (OGC).

Besides these GIS services, Location Services are required for communicating with remote sensors, an example could be services for acquiring location data from mobile devices throw wireless media to send and receive location requests and responses. These services define their interface according to the LIF (Location Inter-operability Forum) specification, which describes a Mobile Location Protocol (MLP) that can be used by an Internet application to request location information from a Location Server. As a part of this Location Services, it is possible to integrate the Mobile Positioning Servers provided by a telecommunication operator through its Location Service Platform.

Geodata provided by these Data Management services are not usually used in an isolated way, instead they are used by the Data Processing services for generating more complex and elaborate data. It is interesting to have services for combining different kinds of geodata, such as maps and location descriptors; for linking many location descriptors; or for calculating structured geodata, such as ideal routes from a set of mobile-device locations.

To achieve this functionality, geospatial services have been implemented for geodata presentation (such as, Style Layer Descriptor Server), utility (Gazetteers and Geocoders), and determination (Route Server and Tracking Server). De-

tails about their specifications can be found in [26]. A Style Layer Descriptor Server (SLD) visualizes the result of a WFS query over a digital map returned by the WMS, applying a visualization style specified by the service client to the displayed geoinformation. On the other hand, the implementation of the WFS has been utilized as the baseline for developing another geoservices [11, 4]: Gazetteers, used to link text-based location descriptions to geographic locations; Geocoders, used to transform a textual term or code, such as an address, place name, or telephone number, into a geographic location; Route Server, used to calculate ideal routes along a set of locations; and Tracking Server, used to store, query, retrieve the latest known geographic location of mobile device. The Tracking Server is not proposed in the OpenGIS specification. However we propose it as a natural way to provide mobile data.

Finally, at the higher level of the architecture, the Data Analysis level is composed by specific application services, such as the Mobile-Resource Tracking Service (MRTS), built on the lower-level services for integrating their functionality into end-applications through Internet. This service allows to make tracking tasks of mobile resources with an installed location-device (such as, vehicles or employees with a mobile phone with an integrated GPS-receptor), to visualize their real-time positions, to plan a route and tracking it, or to generate operational reports. To provide these operations through its interface, it is necessary that data and geoprocessing services of lower levels collaborate among them in an adequate way as an only global system.

2.3 Problems in the real implementation

The interoperability among built services is guaranteed by the use of Internet protocol and data format for accessing to services' operations, and by the definition of these operations according to widely accepted standards (OpenGIS and LIF specifications). Services may interoperate, making easier the integration of services provided by different suppliers. The need of cooperation has been already identified when the conceptual model was presented: "the requirements of levels are not independent".

But the problem is more complex. The remarked interoperability guarantees that services communicate and understand among them. However, from this standard-based interoperability, it is also necessary to have tools for orchestrating services: defining chains of services, synchronizing services and services with applications, or building more complex communication models than HTTP. This need becomes more apparent when the specific implementation of the proposed model is built. For example, the Mobile-Resource Tracking Service requires that services of lower levels collaborate among them for providing its application functionality.

A collection of restrictions has been found in the development of the presented LBS framework that could be extrapolated to other contexts. They are mainly related with the functional characteristics of Location Services and the difficulties for communicating them with other distributed services, such as data and geoprocessing GIS services:

1. Location Services have not persistence for storing received locations from mobile devices. If a service requests the geographic location of a device for processing it, a Location Service must communicate with the remote mobile-device for acquiring and providing it to the requester service.
2. Many location-data consumers are not simultaneously supported. Owing to the fact that a Location Service has not persistence and its interface's operations are invoked using the HTTP Internet protocol, the invoker service can only receive the requested location. If many services require the location-data of a same device, then each one must make an independent request.
3. Asynchronous responses for operations of the interface are more adequate. A service can request an immediate location of a device or a periodic collection of them, but a Location Service does not know in advance how long it is required for acquiring and receiving the requested location. This fact is owing to the introduced delay by communication networks. Therefore an asynchronous model for receiving location responses is more adequate. However, HTTP provides a synchronous request/response model.
4. A Location Service provides operations whose initiative comes from remote mobile-devices and not from the service client. For example, generation of emergency alarms or location events when a device comes into/out from a specific geographical region. Service clients must be able to subscribe to be notified when those alarms or events happen, instead of being continuously requesting to the Location Service to check their occurrence through its HTTP interface.

These restrictions show the need of a more complex communication mechanism among Web services than the one provided by the HTTP protocol:1) able to store exchanged data and to support many consumers and 2) an asynchronous and reactive communication model. Besides, these communication requirements involve matters of service synchronization too: services can be waiting for receiving new location data or being notified by alarms or events for doing their task, such as to update the vehicle location or to show a new alarm on a digital map, to track a predefined route or to recalculate a tracking report. Therefore, the particular target is to provide a high-level tool for coordinating (communicating and synchronizing) Location Services with GIS services. However, this target will be deal with a broader perspective, trying to provide a flexible tool to coordinate any Web service over Internet.

3 Coordinating Web Services

To make possible the coordination among services in an Internet-architecture, the proposed solution has been designed and implemented as a new support-service to provide coordination functionality. Using this new service, distributed services over Internet could communicate and synchronize among them. This coordination service has been built in accordance with the Web-service philosophy for making easier its integration into open architectures: coordination functionality must be accessible via ubiquitous Internet protocols and data format, such as

HTTP and XML; its open interface must hide the implementation details of the service, such as the hardware or software platform on which it is implemented or the programming language in which is written; and it must encourage a flexible and loosely-coupled interaction among Web services.

3.1 A Coordination Model for Distributed Environments

Before building the service, a coordination model for representing supported interactions must be defined. However, it has not been considered the possibility of creating a new model starting from scratch because there are some proposed solutions that can be used as starting points (the creation of a new one should be a different objective and involve other research areas different from our focus). A distributed shared memory (DSM) model for inter-process communication has been selected. This model provides the illusion of a shared memory allowing communicating processes to be uncoupled logically, temporally, and spatially. A well-known DSM model is the Generative Communication [17, 7]. It is based on a so-called blackboard that is used as a shared data space. Entities communicate by putting messages into the blackboard, which can be retrieved later by other entities asking for a certain template message. In the model, senders do not have any prior knowledge about receivers and vice versa. This uncoupling is necessary in an open environment, such as Internet, because it is very important to reduce the shared knowledge between different entities to the minimum. Moreover, this model presents another basic advantage, entities can be replaced or added without adapting or announcing other entities.

The Generative Communication model is based on writings into and readings from a shared space. But it has a failure if writing and reading processes work in a hostile and not reliable environment, such as Internet. The reading operations are blocked if no desired message is available into the space yet. Besides, they may involve long waits.

An event-based approach suggests the possibility of improving the collection of operations proposed by the Communication Generative model, adding a more reactive coordination style. Processes subscribe their interest in receiving event notifications when other writing process insert specific messages into the shared space, instead of being blocked until messages are received.

This communication style, which is very prevalent for distributed systems [9], makes easier and loosely coupling communications [27]. Furthermore, this event-based communication style allows to model asynchronous data communications, identifying a read operation from the space as a subscription and a non-blocked waiting for the event notification.

3.2 Building the Coordination Service

A Java implementation of the Generative Communication model, called JavaSpaces Technology[15], has been used for building the coordination service. In JavaSpaces a collection of processes may cooperate via the flow of Java objects into and out of one network-accessible shared space. Besides, the Jini Distributed

Proceedings of the FIDJI International Workshop on the Scientific Engineering of distributed Java Applications. 2002, p. 33-46.

Event model is incorporated into JavaSpace for firing events when entries that match templates are written into a space. It allows to react to the arrival of entries as they are placed in a space.

The built coordination Web-service encapsulates one or more spaces implemented by JavaSpaces, and provides through its interface the proposed operations by the extended Generative Communication model. In order to coordinate Web services through Internet, these operations are accessible through HTTP protocol.

Distributed services cooperate among them inserting and retrieving messages into/from the encapsulated space using the HTTP operations provided by the coordination service interface. Exchanged messages are encoded in XML format, and producing and consuming services must understand their content. Standards which define how to express the exchanged data (such as the MLP proposed by LIF that defines XML-Schemas for location, alarm or location-event data) are used for achieving this syntactic interoperability.

4 Design and Implementation of the Coordination Web-service

According with the ideas proposed before, a coordination web-service has been implemented and tested in the LBS context. Its kernel consists of three software components (see fig. 2):

XML-Based Space Component This component has been implemented as a Remote Method Invocation (RMI) server over the technological base of JavaSpaces. Its interface provides a collection of operations for writing XML-messages into and reading them from an interaction space and being notified of the writing of a new XML-message into the encapsulated space, according to the previously model presented.

The encapsulated interaction space is a Java space provided by the JavaSpaces implementation (see fig. 3). It allows storing and retrieving Java objects. Therefore, the XML-messages must be internally stored as Java objects. A generic object, called XMLEntry, has been defined for representing an XML-message. This object is able to parser the XML-message and stores the information of its nodes and vice versa, restoring the original XML-message.

The main problem to solve by the use of JavaSpaces is how to specify XML-templates for retrieving XML-messages from the space. The matching rules of JavaSpaces say that a template object and an inserted object can potentially match only if (1) they are from the same class and (2) for each field that is not wildcards in the template object, it must have the same value as its corresponding field in the inserted object. These rules have been extended for working with XML-messages: the XML-Schema is the class of an XML-message, and each node of the message is a field. At the present, simple Schemas are only considered. But in the future, a subset of the XQL language specification will be incorporated

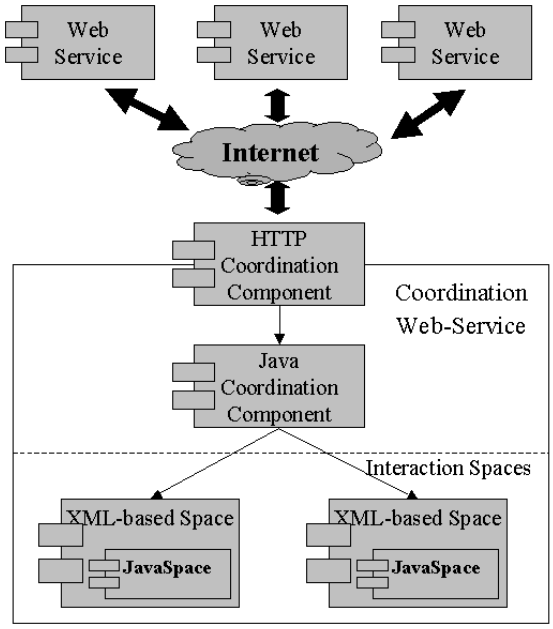

Fig. 2. Software Components of the coordination kernel

to the matching rules for supporting more complex XML queries. XQL is a path expression based query language proposed to the W3C query workshop (`http://www.w3.org/TandS/QL/QL98/pp/xql.html`).

Java Coordination Component This Java component is the core of the coordination service. It has two different interfaces: Basic Coordination Interface (BCI), which provides the collection of writing and reading operations proposed by the Generative Communication model and encourages a cooperative style based on blocking readings; and Reactive Coordination Interface (RCI), whose operations allow a process advertising its interest to generate a specific type of XML-messages, publishing the advertised XML-messages and subscribing its interest to receive XML-messages of a specific type, encouraging a reactive style of cooperation among processes.

When an external process invokes an operation of the interface, a proxy of the invoker process is created inside the component for representing it (see fig. 3). External processes delegate their coordination tasks to their respective internal proxies, which cooperate among them exchanging XML-messages through one or more XML-based Spaces. Therefore, the coordination among external processes happens among their internal proxies, which communicate them the cooperation result. According to the invoked operation, a proxy expert on communication, synchronization or reactive behaviour is created. Proxies has been implemented as Java process able to act as clients of XML-based spaces for exchanging mes-

sages with another proxies and as remote listeners that can be called by spaces when a matching occurs.

Proxies must be able to inform about its internal state and to push data to respective external processes. This connection between both must be established when the proxy is created and remained until it is destroyed. The technique used to connect each other depends on the executing environment where is being used the Java Coordination Component. For example, if a collection of Java processes are cooperating through the developed coordination component and both are running inside a same Java Virtual Machine (JVM), processes and their respective proxies can be connected through message passing. However, in a distributed environment, such as Internet, the HTTP streaming technique can be used to connect them. It consists of remaining open an HTTP connection to push fresh data from the proxy to the remote process (see fig. 3). These processes may even be simple HTML-pages able to receive JavaScript events (for more details, `http://www.pushlets.com/`). In this case, exchanged data are the XML-messages that are the result of its coordination task.

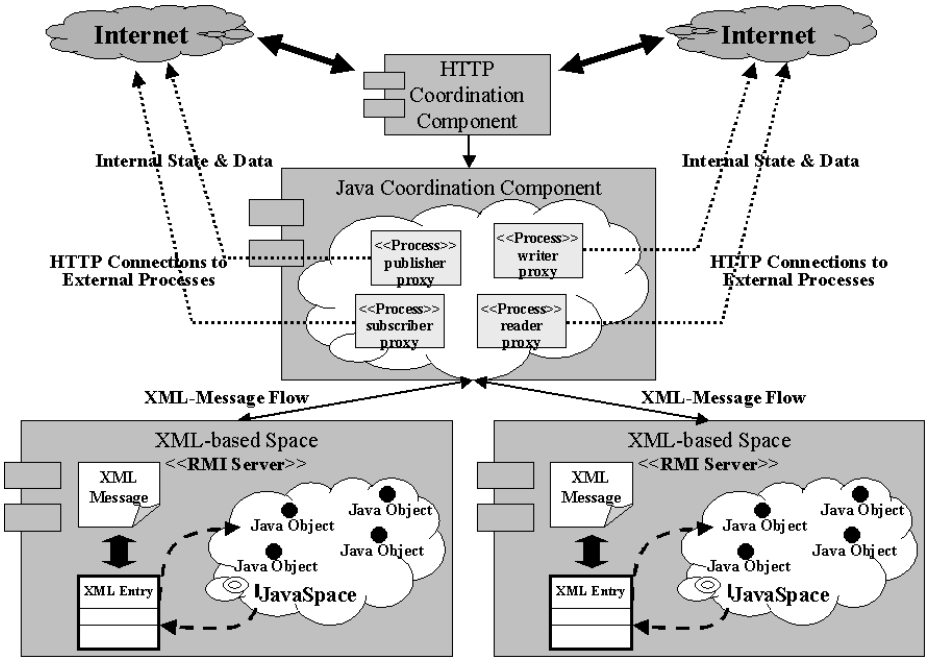

Fig. 3. Coordination Kernel Details

HTTP Coordination Component This component plays as a web-accessible interface of the Java Coordination Component previously presented, providing

through its HTTP interface the same collection of operations. This interface allows web-applications to cooperate independently of the hardware and software platform where they are running and independently of the programming language in which they are written.

The core of this component has been implemented as a Java Servlet, a Java program that resides and executes on a Web-Server, in this case on Apache Server (`http://www.apache.org/`) using Tomcat as a servlet container (`http://jakarta.apache.org/tomcat/`).

5 Benefits of the Proposed Approach

This section presents how the proposed coordination Web-service is able to solve the identified communication problems among Location Services and other distributed GIS services:

1. Spaces encapsulated into the coordination service are persistent and messages may be indefinitely stored into it. Therefore, Location Services can use it as a persistent repository of location data writing received locations from the mobile devices into it.
2. Many distributed services are simultaneously able to access to the coordination service for reading a stored message, such as a location data. So, a location that has been requested and stored into the coordination service may be shared by many consuming services.
3. The publishing and subscribing operations provided by the coordination service's interface support an asynchronous interaction model. It allows time-uncoupled interactions between producing and consuming processes.
4. Distributed services can be subscribed for retrieving messages instead of being making constant readings over the coordination service, being the server who has the notification-based initiative.

6 Conclusions and Future Work

In this work it has been presented an architectural model for organizing Web services and an implementation of it based on standards in the context of the LBS. Despite the lack of problems from a conceptual point of view, real restrictions arise when distributed services must work together harmoniously. For resolving them, it is proposed a coordination Web-service implemented using Java and Internet technologies. The coordination functionality provided by the service is orthogonal to the computing functionality offered by the coordinated entities. This fact keeps the independence between the computing and coordination models.

Open research issues are trying (1) to discover the real potential of the XML language to express synchronization restrictions and work flows among Web services, and (2) to add a new component that integrates thesaurus and ontologies

for supporting the semantic interoperability among Web services. The underlying idea is to extend the concept of matching rules in the way that different values and XML representations will match if correspond to the same concept [22, 30].

Finally, it is important to have a formal instrument for reasoning upon the behaviour of coordinated distributed-services and the coordination service. From a formal point of view, Petri nets are the most attractive formalism for modelling concurrent system that allows formal analysis, graphic representation and the execution/simulation of the system models. In this sense, advanced software development approaches for modelling, implementing and reasoning upon open parallel and distributed systems are based on principles presented in this paper, that is, concurrent object-orientation, generative communication, and Petri nets. In [20] it is presented a Petri net formalism to provide semantics for the Objective Linda language, and modelling the internal behaviour of concurrent objects; and in [5] it is presented transition merging as the main mechanism to represent the interaction between concurrent objects, providing a symmetric form of communication very close to generative communication that allows the cooperation of an arbitrary number of entities, and no direction of communication.

Acknowledgment

The basic technology of this work has been partially supported by the Spanish Ministry of Science and Technology through projects TIC2000-1568-C03-01, TIC2000-0048-P4-02 and FIT-0700002000B270827 from the National Plan for Scientific Research, Development and Technology Innovation, co-supported by FEDER resources.

References

1. OpenGIS Project Document 01-065, *Web feature server implementation specification (version 0.0.14)*, Tech. report, OpenGIS Consortium Inc, 2001.
2. OpenGIS Project Document 02-112, *The OpenGIS abstract specification. Topic12: OpenGIS service architecture (version 4.3)*, Tech. report, OpenGIS Consortium Inc, 2002.
3. OpenGIS Project Document 99-077r4, *OpenGIS Web map server interface specification (version 1.0)*, Tech. report, OpenGIS Consortium Inc, 2000.
4. P. Álvarez, J.A. Bañares, P.R. Muro-Medrano, and F.J. Zarazaga, *Integration of location based services for field support in CRM systems*, GeoInformatics **5** (2002), no. July/August, 36–39.
5. J.A. Bañares, P.R. Muro-Medrano, J.L. Villarroel, and F.J. Zarazaga, *Object-oriented programming and Petri nets*, Lecture Notes in Computer Science, no. 2001, ch. KRON: Knowledge Engineering Approach Based on the Integration of CPSs with Objects, pp. 355–374, Springer Verlag, Berlin Heidelberg 2001, 2001.
6. S. Burbeck, *The tao of e-business services. The evolution of Web applications into service-oriented components with Web-services*, Available in http://www-4.ibm.com/software/developer/library/ws-tao/index.html, October 2000.

7. N. Carriero and D. Gelernter, *Linda in context*, Communications of the ACM **32** (1989), no. 4, 444–458.
8. ———, *A computational model of everything*, Communications of the ACM **44** (2001), no. 11, 77–81.
9. A. Carzaniga, E. Di Nitto, D.S. Rosenblum, and A. Wolf, *Issues in supporting event-based architectural styles*, 3rd International Software Architecture Workshop (Orlando FL, USA), November 1998, pp. 17–20.
10. H.M. Deitel, P.J. Deitel, and T.R. Nieto, *Internet and world wide Web. How to program*, Pentice Hall, 2000.
11. V. Dessard, *GML & Web feature server. The baseline for online geoservices*, GeoInformatics **5** (2002), no. March, 38–41.
12. ESRI, *What are location services? the GIS perspective*, Available in http://www.geojava.com, December 2000.
13. P. Fernández, R. Béjar, M.A. Latre, J. Valiño, J.A. Bañares, and P.R. Muro-Medrano, *Web mapping interoperability in practice, a Java approach guided by the OpenGis Web map server interface specification*, EC-GIS. 2000, 6th European Commission GI & GIS Workshop (Lyon, France), May 2000.
14. P. Fingar, *Component-based frameworks for e-commerce*, Communications of the ACM **43** (2000), no. 10, 61–66.
15. E. Freeman, S. Hupfer, and K. Arnold, *Javaspaces. Principles, patterns, and practice*, Addison Wesley, 1999.
16. F. Friday, N. Davies, and E. Catterall, *Supporting service discovery, querying and interaction in ubiquitous computing environments*, Second ACM International Workshop on Data engineering for wireless and mobile access, Santa Barbara, California (USA), ACM Press, 2001, pp. 7–13.
17. D. Gelernter, *Generative communication in Linda*, ACM Transactions on Programming Languages and Systems **7** (1985), no. 1, 80–112.
18. G. Glass, *The Web services (r)evolution. Applying Web services to applications*, Available in http://www-4.ibm.com/software/developer/library/ws-peer1.html, November 2000.
19. S. Graham, S. Simeonov, T. Boubez, D. Davis, G. Daniels, Y. Nakamura, and R. Neyama, *Building Web services with Java. Making sense of XML, SOAP, WSDL, and UDDI*, SAMS, 2002.
20. T. Holvoet and P. Verbaeten, *Object-oriented programming and Petri nets*, Lecture Notes in Computer Science, no. 2001, ch. Using Petri Nets for Specifyin Active Objects and Generative Communication, pp. 38–72, Springer Verlag, Berlin Heidelberg 2001, 2001.
21. G. Larsen, *Component-based enterprise frameworks*, Communications of the ACM **43** (2000), no. 10, 25–26.
22. E. Mata, J.A. Bañares, J. Gutiérrez, P.R. Muro-Medrano, and J. Rubio, *Semantic disambiguation of thesaurus as a mechanism to facilitate multilingual and thematic interoperability of geographical information catalogues*, Proceedings of the 5th AGILE Conference on Geographic Information Science (Palma de Mallorca, Spain), April 2002, pp. 61–66.
23. M. Mattsson, J. Bosch, and E. Fayad, *Framework integration. problems, causes, solutions*, Communications of the ACM **42** (1999), no. 10, 81–87.
24. P.R. Muro-Medrano, D. Infante, J. Guilló, F.J. Zarazaga, and J.A. Ba nares, *A CORBA infrastructure to provide distributed GPS data in real time to GIS applications*, Computers, Environment and Urban Systems **23** (1999), 271–285.
25. H. Niedzwiadek, *All businesses are in pursuit of Java location services*, Available in http://www.geojava.com/, January 2000.

26. OpenLS, *A request for technology. In support of an open location services (OpenLS) testbed*, Tech. report, OpenGIS Consortium Inc, 2000.
27. D.S. Rosenblum and A. Wolf, *A design framework for Internet-scale event observation and notification*, Proceedings of the sixth European Software Engineering Conference (Zurich, Switzerland) (M. Jazayeri and H. Schauer, eds.), Springer-Verlag, September 1997, pp. 344–360.
28. J.C. Thill, *Geographic information systems for transportation in perspective*, Transportation Research Part C: Emerging Technologies **8** (2000), no. Issues 1-6, February-December, 3–12.
29. J. VanderMeer, *Ubiquitous wireless location interoperability*, Available in http://www.directionsmag.com/, July 2002.
30. U. Visser and H. Stuckenschmidt, *Interoperability in GIS. Enabling technologies*, Proceedings of the 5th AGILE Conference on Geographic Information Science (Palma de Mallorca, Spain), April 2002, pp. 291–297.
31. F.J. Zarazaga, P. Álvarez, J.A. Bañares, J. Nogueras, J. Valiño, and P.R. Muro-Medrano, *Examples of vehicle location systems using CORBA-based distributed real-time GPS data and services*, Computers, Environment and Urban Systems **25** (2001), 293–305.
32. A.K. Ziliaskopoulos and S. Travis Waller, *An Internet-based geographic information system that integrates data,models and users for transportation application*, Transportation Research Part C: Emerging Technologies **8** (2000), no. Issues 1-6, February-December, 427–444.