An architectural Pattern to Extend the Interaction Model between Web-services: The Location-based Service Context

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

Department Of Computer Science And Systems Engineering
University Of Zaragoza
María de Luna 1, 50018 Zaragoza (Spain)
{alvaper, banares, prmuro}@unizar.es
http://iaaa.cps.unizar.es

Abstract. Internet has succeeded as a global information system mainly because of its availability and openness, and the simplicity of its standards and protocols. However, the current use of Internet as universal middleware has clearly shown the lack of maturity of Web technology to support distributed applications, which involve communication, cooperation, and coordination. This paper proposes an architectural solution to solve these interaction restrictions. It is based on an extension of the service-oriented architectures, adding a new coordinator role that allows more flexible relationships between service providers and requestors than the provided by the client/server model. This role is inspired by the Blackboard architectural pattern and it is the conceptual basis of a Web-Coordination service able to coordinate distributed and heterogeneous applications through Internet. To prove the effectiveness of this proposal, the Web-Coordination service has been used in an highly dynamic and collaborative application context, the Location-Based Services.

Keywords: Web-service architectures, Web-service coordination, Location-based services

1 Introduction

The World Wide Web was born to provide a minimal technology for the delivery of simple multimedia hypertext documents across the Internet. For this reason, its architecture has been kept as simple as possible, with a elementary communication protocol (HTTP), a simple hypertext description language (HTML), and an addressing schema for document resources globally valid all over the Internet (URLs). Nowadays, Internet has become the de facto standard distributed platform for rapid application development by integrating network-accessible services. This new tendency has shown that the current state of Internet provides little to support the collaborative work of Web-distributed applications owing to the passive nature of the World Wide Web [7]. Therefore, it requires to

redesign the Web technology to provide enough support for building Web-based applications which involve communication, cooperation, and coordination.

These interaction restrictions are more obvious in highly dynamic and cooperative application-context, such as Location-Based Service (LBS) context. Our experiences in this context have become apparent that the integration of Web services and applications requires new interaction models more complex than the provided by the client/server model: 1) many-consumer models, 2) asynchronous models based on events, or 3) reactive models that allow to the server to have the initiative of the interaction [2]. Traditionally, these interaction requirements have been resolved using solutions based on wrapping techniques or/and middlewares. These ad-hoc solutions are hardly reusable and their results are a collection of highly coupled and cohesive services and applications. Therefore, the modern Web must be based on alternative architectural-solutions.

It is basic to understand the Web-based architectures, particularly the Service-Oriented Architectures (SOA) [10], because they encapsulate the decisions about the architectural elements and their relationships [15]. These Web architectures must be improved without changing their original capabilities to support these interaction models. The main aim of this paper is to describe an architectural solution to support more flexible relationships between Web service. It is based on an extension of the SOA model and on concepts of architectural patterns, coordination models and event-based technology.

This paper is structured like is described below. In the section 2 it is presented the architectural solution to support the required models (many-consumer, asynchronous, and reactive models). A new coordinator role has been added to the model. From a conceptual point of view, this coordinator role encapsulates a coordination model able to communicate and coordinate Web-services. This coordination model, called Linda, has been modified to be used in open and distributed environments. The coordination model and its extension are presented in the section 3. From this conceptual solution, it has been designed a Web-Coordination service to play the coordinator role in any Web-service framework built according to the SOA model. Details about its design are presented into the section 4. This coordination service has been used in the development of a LBS complete solution based on the integration of two Web-service frameworks. The suitability of the solution has been tested in an use case to improve a parcel service. Details are described in the section 6. Finally, conclusions and future work are described.

2   An Extension of the Service-oriented Architecture

The architectural solution to support the new interaction models is based on an extension of the service-oriented architecture. As shown in figure 1, a new architectural role was added to the model to communicate and synchronize requester and provider services. This new role, called *Service Coordinator*, is inspired by the Blackboard architectural pattern [5]. It allows a collection of independent services to work cooperatively on a common data structure or blackboard using

a shared vocabulary. These services communicate by putting data messages into the common structure, which can be retrieved later on by other services asking for a certain template message. The Blackboard architectural pattern determines the next process that will change the common data structure according to an opportunistic strategy.
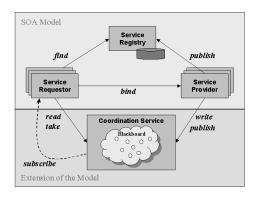

Fig. 1. Extended SOA model

This proposal provides a collection of promising features to be used in distributed and open environments:

– It allows services distributed in space to cooperate distributed in time. The cooperation is uncoupled because the writing services do not have any prior knowledge about the readers and vice versa. The interaction style is adequate enough to be used in environments where it is very important to reduce the shared knowledge between different remote entities to the minimum, such as Internet.
– Writing and reading services can cooperate without adapting or announcing themselves. It allows flexible modelling of interactions among services in highly-dynamic environments.

The key to this approach is the definition of a pure coordination model and inspired by the Blackboard pattern that it is the conceptual basis of a Web-Coordination service that plays the coordinator role. This new service must be able to coordinate distributed and heterogeneous applications through Internet supporting many data consumers and asynchronous and reactive interactions. In this way, new communication and synchronization mechanisms complement available ones. We did not consider the possibility of creating a new coordination model from scratch because there are some proposed solutions that can be used as starting points (the creation of a new model is a different goal and involves other research areas). The LINDA model was selected [9]. It is based on Generative Communication that defines a model for inter-process communication inspired by the Blackboard architectural pattern.

3   A Web-coordination Service Based on the Linda Model

In Linda, the common data structure and messages are called tuple-space and tuples, respectively. A tuple is something as ["Gelernter", 1989], a list of untyped atomic values. The tuple space is a collection of tuples that acts as a shared memory, to which certain operations, that able processes to read and take tuples from and write them to it, can be applied in a decentralized manner. For instance, the operation in(x?) tries to match the tuple x? with a tuple in the shared space. If there is a match, a tuple is extracted from the tuple space; otherwise, it blocks until a convenient tuple appears. The parameter for in() can be a query tuple with a wildcard, like ["Gelenter", ???] (these query tuples are called templates). The match is then free for the wildcard and literal for the constants values. To use this coordination model in an open and hostile environment, it must be extended. Our proposal is based on this obvious observation: if this simple matching strategy is replaced with a complex matching, then very general kinds of interoperability can be achieved.

3.1   Linda for Open Environments

Improving data representation capability Due to the fact that distributed processes over Internet communicate exchanging XML-encoded data, the idea of tuple has been extended to be able to represent data according to this standard format. Tuples are extended to be described by means of attribute/value pairs, like: [(author, "Gelernter"), (year, 1989)]. Although this is still an untyped setting, this bit of structure, allows recovering information from a distributed context. For the presented approach, it is important to remark that the "top-level" structure of any XML document admits the expression

```
[(att1,<val 1>),...,(attN,<val N>)],
```
but where each <val i> is structured (in particular, it may be XML-based).

Supporting reactive operations The reading operations are blocked if no desired tuple is available in the space, which can involve long waits. An event-based approach suggests the possibility of improving the collection of operations of the tuple-space interface, adding a more reactive coordination style. Processes can subscribe their interest in receiving event notifications when other writing entities insert specific tuples into the shared space, instead of being blocked until tuples appear.

Now Linda can be used in a XML context supporting a structured matching, and providing a collection of operations that promise an interesting way to coordinate, communicate and collaborate on Internet. Besides, this structured matching can be applied to support semantic interoperability (in [1], a semantic matching for distributed heterogeneous environments has been described). Thus, if an operation in() is invoked on a different entity, where the term "author" is not used, but "creator" is used in its place, and if there is a convenient mapping between ontologies, then the request [(creator, ???), (year, 1989)] can be

successfully satisfied. This kind of semantic matching is implemented through Internet, using XML as transfer format.

3.2   Using Linda to Coordinate Web-services

Once LINDA has been extended, a Web-Coordination Service (WCS) has been designed to play the coordinator role in any Web-service framework built according to the SOA model. The WCS encapsulates the shared tuple-space, and offers the reading, writing and reactive operations of the extended Linda model as a part of its interface. These operations are accessible via ubiquitous and standard Internet protocols, such as HTTP or SOAP, and exchanged tuples are stated using standard data formats, like XML. Web-services implemented in accordance with different computation models (written in several programming languages, or running over heterogeneous hardware or software platforms), can cooperate if they share a common XML-based vocabulary. The Web Coordination service ensembles them in an orthogonal way to their heterogeneous features.

Recently, a similar solution has been presented by *IBM*, *Microsoft Corporation*, and *BEA Systems* [6]. They propose a framework to create *coordination contexts*. A coordination context provides functionality to registry: 1) Web-services and applications that require to coordinate; and 2) coordination protocols to make their coordination requirements possible. According to this proposal, Web services and applications are responsible to define the used protocols. This solution is more flexible than ours, because it is possible to create many coordination contexts using different protocols (we propose an only free-context protocol based on Linda to coordinate any Web resource). However, it ignores the complexity of defining new coordination protocols and promote the spread of ad hoc protocols.

4   Design and Implementation of the Web-coordination Service

In a more detailed description, the designed WCS is composed by three software components (see the figure 2):

The *XML-based Space* component encapsulates the tuple-space. Its interface provides a collection of operations for writing XML tuples into and reading them from the tuple-space, and being notified of the writing of a new XML tuple into the encapsulated space, according to the presented extension of Linda. This component has been built from *JavaSpaces* technology [8], a Java implementation of Linda, and its detailed description is the focus of this section.

On the other hand, the *Java Coordination* component is the core of the service. It has two different interfaces: the Basic Coordination Interface (BCI), which provides the collection of writing and reading operations proposed by Linda and encourages a cooperative style based on blocking readings; and the Reactive Coordination Interface (RCI), whose operations allow a process advertising its interest to generate a specific type of XML tuples, publishing the advertised XML tuples and subscribing its interest to receive XML tuples of a
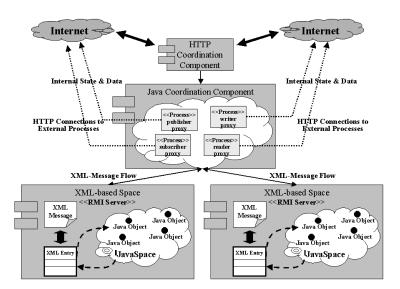
Fig. 2. Software Components of the coordination service

specific XML schema, encouraging a reactive style of cooperation among processes.

This component is a repository of agents (a computational entity which acts on behalf of other entities in a semi-autonomous fashion, and performs its tasks cooperating with other agents; the mobility and learning attributes have been excluded in this context). Every time an external Web service or application invokes a coordination operation, it is created an internal agent. These agents are able to coordinate with another agents exchanging XML tuples through XML-based spaces. Therefore, the required cooperation by external entities is executed by their respective internal agents. The result of this cooperation is communicated from agents to external entities using an event-based mechanism scalable to Internet (more details can be consulted in [2])

Finally, the *HTTP Coordination* component plays as a Web-accessible interface of the Java Coordination component, providing through its HTTP interface the same collection of operations.

4.1   Building XML-based Spaces

According to the described design, the interactions between Web service and applications happen into the *XML-based Space* component. This component has been built using *JavaSpaces*, which provides a shared repository of Java objects and operations to read and write objets from/into it. This saves rewriting of routine code to manage tuple-spaces. Nevertheless, it was necessary to enrich *JavaSpaces* to work with XML tuples.

A generic Java object has been designed to encode any XML tuple. It is composed by structured fields to store the nodes of the XML tuple. In more

detail, the generic object has two structured fields to store in ordering the tag names and values of the XML tuple. The tag name of the first node of the XML tuple is stored in the first component of the tag-name field and its value in the first component of the tag-value field, and so on. But the matching rules of *JavaSpaces* are not adequate for working with objects composed by structured fields. Objects are matched by complete fields, not within the contents of the filed. This is owing to objects are serialized for storing them into the space, and the matching between objects is made applying the equal operator on the corresponding field value. Therefore, it is required to extend the matching rules of *JavaSpaces* to support this XML-based interoperability.

To resolve these rule restrictions, the matching is made in two steps. In the first step, the matching rules of *JavaSpaces* are used. The original template is saved for the second step and a copy of it is created for being used in the first step. The tag-value field of this copy is set to null value. When it executes a read operation (provided by the *JavaSpaces* interface) using this template object, a returned object represents an XML tuple with the same XML-Schema as the template because the tag-name field is only consider for the matching. In a second step, it is invoked a particular matching method of the retrieved object, using the original template as a real parameter. This method checks that each not-null component of the templates value-field have the same value in the corresponding component of the retrieved objects value-field. If it returns true value, the retrieved object matches the template according the XML-tuple matching rules. Otherwise, the retrieved object has the same XML-Schema as the template but it does not match the template according the second matching rule for XML tuples. Following, the first step is made again until an object matches according the XML-tuple matching rules. In [4], a matching strategy in two steps is also described to exchange encrypted data among distributed processes.

It is important to realize that this matching proposal does not guarantee the semantic of Linda model because objects are retrieved from the space according to a non-deterministic strategy. For example, the first step could retrieve again and again from the space a set of objects that return false value in the second step. Nevertheless, another different object stored into *JavaSpaces* could match the template.

4.2   A Pattern to Guarantee the Semantic of the Linda Model

To resolve these semantic problems, a communication pattern has been designed which partitions the tuple-space in accordance with the XML-Schema of the stored tuples. All tuples with the same XML-Schema are stored in the same partition. A new Java object, called *Channel*, has been designed to manage a partition. Its responsibility is to guarantee the access to all the tuples stored into the partition. To achieve this aim, a channel is composed by a structured field that stores a collection of references to the tuples stored into the partition (tuples are stored as Java objects). The figure 3 shows an example of tuple-space partitioned by two different channels.
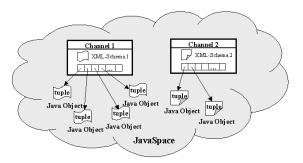
8   P. Álvarez et al.

Fig. 3. Partition based on *channels* of *JavaSpaces*

When a writing operation is invoked, it is checked if there is any channel into the tuple-space that stores XML tuples with the same XML-Schema. If it exists, then the XML tuple is stored into the channel (a reference to the inserted tuple-object is saved into the channel object); or else, a new channel is created and the XML tuple is inserted into it.

On the other hand, when a reading operation is invoked, it is retrieved a copy of the channel that stores all the tuples with the same XML-Schema than the reading template. This copy is used to: 1) access all the XML-tuple objects stored into the partition; and 2) mark those XML-tuple objects that have been retrieved from the space, but they do not match the reading template. This internal strategy based on marking the checked tuples allows to retrieve an only time an XML-tuple object in the first step of the matching. If it exists a tuple into the space that matches the reading template, it will be found; or else, the reading operation will be blocked. Therefore, it is guaranteed the semantic of the reading operations of the Linda model.

The time costs involved in performing the reading operations have been evaluated for analyzing the efficiency of the proposed pattern. It is important to define a cost time categorization which represents the execution time of a Linda primitive from the user process's point of view. In this work, the categorization proposed by Bonita, a Linda implementation for distributed environment, has been used [17]. For our analysis, the most important time-cost is $T_{ReadingProcess}$, which is the time taken for the execution of a reading operation on the XML-based space finding the suitable tuple (a tuple that matches the reading template is always available into the space). To evaluate this time cost have been created partitions composed by a changeable number of tuples with the same XML-Schema, but with different content. The figure 4 shows the $T_{ReadingProcess}$ for the proposed pattern. This time cost shows a linear increase according to the number of tuples stored into the partition. As this number increases, a greater number of tuples must be checked by the matching until founding a tuple that matches the reading template. Therefore, this pattern based on channels is inefficient when the created partitions store a high number of tuples (for example, for partitions composed by more than one thousand tuples).
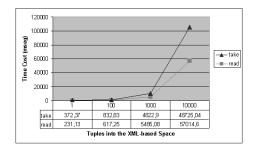
Fig. 4. Efficiency of the reading primitives

4.3   Partitioning Interaction Spaces to Improve the Efficiency

To improve the previously pattern, making more efficient the XML-based space, it is necessary to minimize the amount of search required for the matching. This solution must be transparent for processes that cooperate through the XML-based space, and orthogonal to the Linda model. The idea is to make an hierarchical partition of the tuple-space based on *meta-channels* and *channels* (see figure 5). A meta-channel stores all the XML tuples with the same XML-Schema. This partition level of the tuple space is created at run-time. To make more efficient the reading operations, a meta-channel is divided into one or more channels. A channel stores all the XML-tuples with a specific value into an XML-node, called *reference node*. This node must be specified at compile time (a good choice of the reference node is a determining factor to improve the efficiency; it is important to choose a node whose value is usually not-null in the reading templates).

   In a more detail, a new object has been designed to manage partitions. The meta-channel object is composed by a structured field able to save a collection of references to channel objects. On the other hand, the channel object has been reused from the previously pattern, and therefore, it stores a set of references to Java objects that represent XML tuples. However, an improvement has been added to control the size of the channels. It is possible to set up a maximum size of channel, storing into it the most recent XML tuples.

   When a writing operation is invoked, it checks if there is any meta-channel into the tuple-space that stores XML tuples with the same XML-Schema. If it exists, the tuple is inserted into the adequate channel according to the value of its reference node (if the channel does not exist, then it is created in run-time); or else, a new meta-channel and a channel indexed by the value of the tuple's reference node are created, and then, the XML tuple is inserted.

   On the other hand, when a reading operations in invoked, it is retrieved a copy of the meta-channel that stores all the tuples with the same XML-Schema than the reading template. This copy is used to check if it exists into the meta-channel a channel indexed by the same value than the reading template's reference node. If it exist, a copy of the channel is retrieved and used to look up the required tuple according to the tuple-marking strategy of the pattern based on channels;
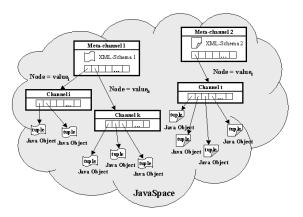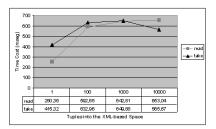
10   P. Álvarez et al.

Fig. 5. Partition based on *meta-channels* and *channels*

or else, the reading operation will be blocked. Finally, to remark that it is possible to specify a reading template with a wildcard in the reference node. In this case, the search is applied over the meta-channel (a meta-channel is the union of all its channels) instead of being applied over a specific channel.

To evaluate this new proposal, a meta-channel composed by a changeable number of channels has been created (the maximum size of a channel is one hundred XML tuples). In the figure 6, the graph at the top shows the $T_{ReadingProcess}$ for the proposal. The time cost does not show a linear increase according to the number of tuples stored into the meta-channel, because channels reduce the search space. Beside, it is important to remark that the great size of channels will be conditional on increased time cost. In the figure 6, the graph at the bottom shows how the size of a channel has an influence on the $T_{ReadingProcess}$.
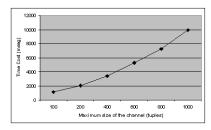

Fig. 6. Efficiency of the reading primitives making a hierarchical partition

5   The Location-Based Service Context

Geographic Information Services (GIS) and Location-based Services (LBS) are two prototypical technological contexts where many standardization initiatives

have arisen. LBS extend GIS spatial processing capabilities by integrating wireless communications, location data and Internet technology [11]. In this context, two well-positioned organizations have emerged as the leaders of the LBS interoperability: LIF (Location-Interoperability Forum[1]) and OGC (Open GIS Consortium[2], and its Open Location Service Initiative (OpenLS)). Both are promoting and defining standard interfaces for a collection of wireless, Location and GIS services to provide the required LBS functionality. In more detail, LIF specifies the Mobile Location Protocol (MLP) that defines a simple but complete interface between location-data acquisition services (called, Location services) and location-based applications. On the other hand, OGC provides specifications for GIS and geoprocessing services. These initiatives are not the only ones in the LBS context, but they are the most representative.

OGC and LIF are aware of the necessity of integrating their standardization initiatives, but they have not reached an agreement on the integration strategy yet. Starting from the work of LIF and OGC specifications we have designed two Web-service frameworks to provide LBS functionality (see figure 7). Service interfaces have been specified according to previous standards to ensure the interoperability, and are accessible via standard Internet protocols and data formats, such as SOAP, HTTP or XML. As result of this Web standard-based approach, the resulting functionality has been easily integrated into different location-dependent applications, such as a Web-enabled fleet tracking system, and a Customer Relationship Management system [3].

As it is illustrated in figure 7, the OGC-based framework is composed by three functional levels of services . The *Data Management* level is the base of the designed framework. Its services must be able of providing the necessary support for the storage and recovery of a wide variety of geodata: maps, location descriptors (street addresses, roads, place names, telephone numbers, etc.), 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 are integrated to fulfill these requirements: Web Map Servers (WMS), able to visualize digital map on the Internet; Web Feature Servers (WFS) [12], to store geodata and execute spatial and non-spatial queries over them; and Web Traffic Servers (WTS), that provide a interface to gain access to a variety of traffic information sources (a standard specification of this service is not mature yet). The result of a query to a WFS or WTS is a collection of GML-encoded data (an XML-based standard specified by OGC to encode geographical information [13]). These GML data can be directly visualized on the user interface of a Web-based application.

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

---

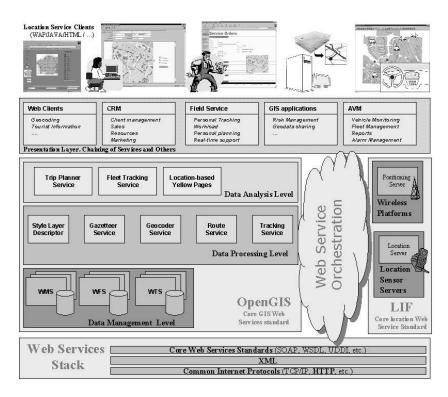[1] http://www.locationforum.org/
[2] http://www.opengis.org/

Fig. 7. Standard-based LBS framework based on LIF and OGC orchestration.

as ideal routes from a set of mobile-device locations. Services of these level include geodata presentation (such as, Styled Layer Descriptor Server), utility (Gazetteers and Geocoders), and determination (Route Server). Details about them can be found in [14]. A Styled 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 geodata. On the other hand, the implementation of the WFS has been utilized as the baseline for developing another geoservices [3]: 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; and Route server, used to calculate ideal routes along a set of locations.

Finally, the higher level of the OGC framework, called *Data Analysis* level, is composed by a collection of application services. These services are built on the lower level services and integrate their functionality into a wide variety of location-dependent Web-based applications.

Our strategy to integrate the OGC and LIF frameworks lies in the incorporation of mobile location data into the functional chain of the OGC framework. This strategy has been materialized with the design and implementation of a *Tracking service* like a WFS able to store location data of mobile entities. The

mobile resources may be seen as "dynamic" features whose geographical information changes throughout the time. This decision makes possible to perform spatial queries over location data of mobile entities exploiting the OGC standard geospatial query language.

This Tracking service requires to collaborate to update the stored locations with services able to acquire location data from remote mobile devices via wireless media. These acquisition services are called Location services and compose the LIF-based framework. This collaboration with the Location services is described by the MLP protocol proposed by LIF, which can be used by an Internet application to request location information from a Location Server. Although from a functional point of view the Tracking service (or Web-based application, in general) and Location services are all compatible and standards guarantee their interoperability, some orchestration problems have been identified when they cooperate through Internet such as it has been previously mentioned [2]. It is possible and easy to find solutions to them based on ad-hoc or wrapping techniques. However, the proposed WCS is a an adequate and more reusable solution to solve them as it will be illustrated in the next section.

The figure 7 shows examples of application services that uses the underlaying infrastructure: Fleet Tracking service, Trip Planner, location-based Yellow Pages, etc. A Fleet Tracking service has been developed to support the required functionality of the LBS use case presented in this paper. This service allows to make tracking tasks of mobile resources with an installed location-sensor 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.

6 Use case: Integration of LBS Functionality to Improve a Parcels Service

The work presented in this paper has been applied in the integration of LBS functionality into a parcel-service application to track mobile resources. Field personal, which delivers parcels carrying a radio terminal with a GPS receptor, could be on different radio coverage zones and move freely from one to another at any time. Let's suppose that due to the orographic characteristics, there are some different zones of trunking radio coverage without any link among them. It is necessary a "real time" tracking, but there is no possibility of getting the delivery men's locations continuously via radio.

The characteristics of the wireless communication media, with limited bandwidth and frequent disconnection, require a decoupled an opportunistic style of computation [16]. In our use case, we need to use the last known location, without the need to know the server that delivers this location. The proposed WCS provides the required uncoupling and provides the data in a opportunistic way.

14   P. Álvarez et al.

The best solution found has been to distribute the location-data acquisition process using the deployment illustrated in figure 8. To achieve this purpose, the system has a LIF Location service to acquire location data on each coverage zone. LIF Location servers are configured in a first step (see figure 8). Remote devices send its location to the receptor that is available at the moment, depending on the coverage zone where they are. Location data received by any Location service are immediately published into the Web Coordination service (WCS). In the figure, the second step shows an example of HTTP request invoked by a Location service to write an XML-encoded location into the WCS.
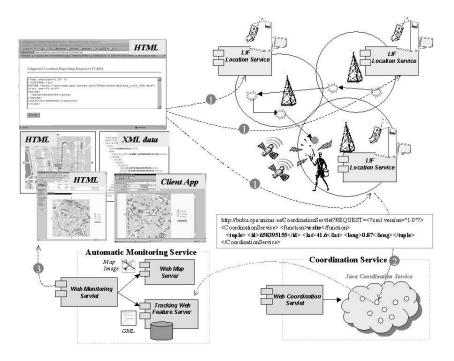

Fig. 8. Deployment architecture for tracking delivery parcels.

These location data stored into the WCS are integrated into the functional chain of the proposed framework by a Tracking service. This service is subscribed in the WCS to be notified when a new location is written into it. These notifications, third step in figure 8, trigger actions inside the Tracking service, being the most regular the updating of the field personal locations stored into its database (in this way, the last acquired and notified location is always accessible). The subscribed service receives location data without having any knowledge about the providers, and are unaware of the restrictions of the data-acquisition process.

The interface of the Tracking service provides operations to know these latest locations to make spatial or non-spatial queries. Remember that to make easier the integration of this service into the OGC-based framework, its interface has

been defined according to the Web Feature Server specification. Additionally, more elaborated and filtered events can be published into the WCS taking advantage of the spatial functionality of the services. For example, the Tracking service could publish location-based events to notify that a delivery man has come into/out an irregular geographical region (department/area of a city, a province, etc.), or that it is a specific distance away from a geographical point.

Finally, with this technology it is possible to establish Web-based applications that dynamically access to network-accessible interoperable services, from simple Web browser to complex client-systems. Additionally, other services can be developed to recover the inserted locations from the WCS and to store them into local database to analyze their routes and generate operational reports. It is clear that the combination of open architectures, domain standards and the ability to access a wide variety of data sources are critical for the success of this Web-based approach.

7   Conclusions and Future Work

This paper proposes an architectural solution to support new interaction models between Web services and applications. This proposal is based on the creation of a new role in the SOA model, interpreted by a Web coordination service, whose functionality is orthogonal to the computing functionality offered by the coordinated services. This coordination Web-service resolves the distributed computing difficulty in "gluing together" multiple and independent Web-services.

The key is the coordination model that defines how distributed entities collaborate. Linda has been extended to be used in open and hostile environments. This extension is based on replacing its simple matching rules with a complex matching able to support new kinds of interoperability (XML-based, semantic, etc). From this model, a Web coordination service has been designed focusing the main attention in the component that implements the interaction spaces, called XML-based space component, and how to get an efficient implementation of it using *JavaSpaces* technology and communication patterns. The final results has been applied in the LBS context.

Finally, open research issues are attempting to: (1) discover the real potential of the XML language to express synchronization restrictions and workflows among services; and (2) tackle with the integration of XML data and pure coordination models such as Linda and Petri Nets, solving where and how to represent coordination restrictions and XML data in a systematic way, and how to model the internal behavior of the Web-services that cooperate externally through the coordination service.

Acknowledgment

References

1. P. Álvarez, J.A. Bañares, E. Mata, P.R. Muro-Medrano, and J. Rubio, *Generative communication with semantic matching in distributed heterogeneous environments*, Proceedings of the 9th International Workshop on Computer Aided Systems Theory. Extended Abstracts. (R. Moreno-Diaz jr., A. Quesada-Arencibia, and J.C. Rodriguez, eds.), Universidad de las Palmas de Gran Canaria, February 2003, pp. 237–239.
2. P. Álvarez, J.A. Bañares, P.R. Muro-Medrano, J. Nogueras, and F.J. Zarazaga, *Scientific engineering for distributed Java applications*, Lecture Notes in Computer Science, no. 2604, ch. A Java Coordination Tool for Web-Sercice Architectures: The Location-Based Service Context, pp. 1–14, Springer Verlag, 2003.
3. 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.
4. L. Bettini and R. De Nicola, *Scientific engineering for distributed Java applications*, Lecture Notes in Computer Science, no. 2604, ch. A Java Middleware gor Guaranteeing Privacy of Distributed Tuple Spaces, pp. 175–184, Springer Verlag, 2003.
5. F. Buschmann, R. Meunier, H. Rohnert, P. Sommerlad, and M. Stal, *A system of patterns*, Wiley, 1996.
6. F. Cabrera, G. Coopeland, T. Freund, J. Klein, D. Langworthy, D. Orchand, J. Schewchuk, and T. Storey, *Web service coordination (ws-coordination)*, Tech. report, IBM & Microsoft Corporation & BEA System, September 2002.
7. P. Ciancarini, R. Tolksdorf, and F. Vitali, *Towards an interactive Web*, Submitted for publication, IEEE Internet Computing. Available in http://flp.cs.tu-berlin.de/pagespc/ieeeip/ciancarini.html, February 2003.
8. E. Freeman, S. Hupfer, and K. Arnold, *Javaspaces. principles, patterns, and practice*, Addison Wesley, 1999.
9. D. Gelernter, *Generative communication in Linda*, ACM Transactions on Programming Languages and Systems **7** (1985), no. 1, 80–112.
10. 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.
11. H. Niedzwiadek, *All businesses are in pursuit of Java location services*, Available in http://www.geojava.com/, January 2000.
12. OpenGIS Project Document 01-065, *Web feature server implementation specification (version 0.0.14)*, Tech. report, OpenGIS Consortium Inc, 2001.
13. OpenGIS Project Document 02-023r4, *Opengis geography markup language (GML) implementation specification (version 3.0)*, Tech. report, OpenGIS Consortium Inc, 2003.
14. OpenLS, *Call for participation in the open location services testbed. phase 1 (openls-1)*, Tech. report, OpenGIS Consortium Inc, 2000.
15. D. E. Perry and A. L. Wolf, *Foundations for the study of software architectures*, ACM SIGSOFT Software Engineering Notes **17** (1992), no. 4, 40–52.
16. G.P. Picco, A.L. Murphy, and G.C. Roman, *LIME: Linda meets mobility*, Proceedings of the 21st International Conference on Software Engineering (ICSE'99) (D. Garlan and J. Kramer, eds.), ACM Press, May 1999, pp. 368–377.
17. A. I. T. Rowstron and A. M. Wood, *Bonita: A set of tuple space primitives for distributed coordination*, Proceedings of the 30th Annual Hawaii International Conference on System Sciences, vol. 1, IEEE Computer Society Press, 1997, pp. 379–388.