Mobile Networks and Applications 8, 377–387, 2003

The AROUND Architecture for Dynamic Location-Based Services

RUI JOSÉ, ADRIANO MOREIRA and HELENA RODRIGUES
Information Systems Department, University of Minho, 4800-058 Guimarães, Portugal

NIGEL DAVIES
Computing Department, Lancaster University, Bailrigg, Lancaster, LA1 4YR, UK, and
Department of Computer Science, University of Arizona, Tucson, AZ 85712, USA

Abstract. This paper presents a generic concept of location-based service as an abstraction for supporting the association between computational resources and location. The objective is to extend the advantages of service-based architectures to the development of location-based systems, thus providing a more open and extensible alternative to the "vertical" approaches typically used in this type of system. The novel AROUND architecture is proposed as an approach for supporting location-based services in the Internet environment. AROUND provides a service location infrastructure that allows applications to select services that are specifically associated with their current location. The architecture includes a flexible scope model that defines the association between services and location, and a service location infrastructure organised by spatial criteria and optimised for location-based queries. Based on a prototype implementation of this architecture, we have developed two case studies that illustrate the use of this approach for developing location-based systems. The overall results provide a valuable insight into the applicability of the architecture, and suggest that this model of location-based services can provide a useful approach for the development of a wide range of location-based applications.

Keywords: location-based services, mobile computing, service discovery, info-mobility

1. Introduction

Recent years have seen considerable research and market interest in applications that exploit information about the physical position of users to provide them with contents that are tailored for their particular location. This type of application enables scenarios in which people are moving on their daily lives while electronically interacting with their surrounding environment or satisfying complex information needs that are directly related with that environment. However, developing this type of system is still a very challenging task, the main reason being the lack of generic mechanisms for supporting the association between network resources and physical location. As a result, each new location-dependent system must develop its own location model and its own implementation of functionality that is overall common to many other systems of that nature. Furthermore, these "vertical" approaches are valid only for narrow application domains or specific technologies, and typically result in systems with costly and inflexible designs that rank low in terms of openness and extensibility. As this type of application matures, the role of third-party entities in the provision of location-based information is likely to experience a rapid growth, leading to the key research challenge of how to address the issues of openness and heterogeneity in the design of location-dependent information spaces.

In this paper, we propose to address those issues with the use of a particular concept of location-based service as an abstraction for modelling the association between network resources and physical location. In the context of our work, a *location-based service* is a system providing a facility to the network whose usage is semantically associated with physical space. This semantic association with physical space is a fundamental characteristic of these services, and typically results from one of the following situations: the service provides some interaction with the environment, e.g., controlling the temperature of a room; the service is acting as an electronic counterpart to some real-world entity, e.g., a restaurant; or the service provides information associated with a geographical area, e.g., maps or traffic information.

Our main objective with this service model is to extend the advantages of service-based architectures, more specifically their ability to deal with openness and heterogeneity, to the development of location-dependent applications by allowing applications to dynamically select services that are specifically associated with their particular location. This particular form of service location allows applications to be made location-dependent simply by using whatever services are available at each visited location. With this objective in mind, we propose a globally available service selection infrastructure based on two key requirements:

(i) **Physical locality**. The system must provide a service selection mechanism that is effectively based on physical locality, thus allowing two physically co-located devices to discover the same services independently of their networks or administrative domains. Supporting this form of locality implies a discovery process capable of working in the wide area, over multiple network technologies, and across multiple administrative domains.

(ii) **Spatially distributed operation**. The service location space should allow queries to be scoped in spatial terms

and be answered without having to search the entire offer space. Even though the system aims to be globally available, in the sense that it should be possible to discover local services anywhere, it does not aim to support global discovery, in the sense of searching for services available anywhere. This particular characteristic of location-based services discovery should be explored in the design of a scalable global infrastructure for location-based services by creating a distribution model that reflects the spatial organisation of the service offer space.

Based on these requirements, we propose the novel AROUND architecture, which provides a model for defining the association between services and location, and a service location infrastructure optimised for location-based queries.

To evaluate the approach, we have created a prototype implementation of the key elements of this architecture and a number of case studies in which the architecture is used as the basis for location-dependent systems. The overall results provide a valuable insight into the effectiveness and applicability of the approach, and suggest that location-based services can provide a useful model for the development of a wide range of location-based applications.

The remainder of this paper is organised as follows: in the next section we describe the AROUND architecture, our proposed mechanism for enabling location-based service selection over the Internet. This is followed, in section 3, by the description of two case studies which exemplify the use of location-based services as the basis for location-dependent systems. The results of these case studies are discussed in section 4, where we also discuss other aspects of the evaluation of the architecture. We then analyse, in section 5, some existing systems and relate them to our own work. Finally, in section 6, we discuss future work and present our conclusions.

2. The AROUND architecture

This section describes the architecture of the AROUND system, which is our proposed mechanism for enabling the discovery of location-based services over the Internet.

2.1. Proximity models for service discovery

A key issue when modelling the association between services and physical space is which geographical criteria to use when determining the "nearby" services that a client should select. In the AROUND architecture, we have considered the two distinct models of location-based selection represented in figure 1: distance-based and scope-based.

In the *distance-based* model the client selects the servers located within some distance from its own position. Given that proximity is a rather abstract concept, and that what we perceive as being proximate may have dramatic variations according to our current activities, we also consider that the client is able to dynamically change the proximity range to be used in the selection process. The main limitation of this model, however, is that the correlation between context and

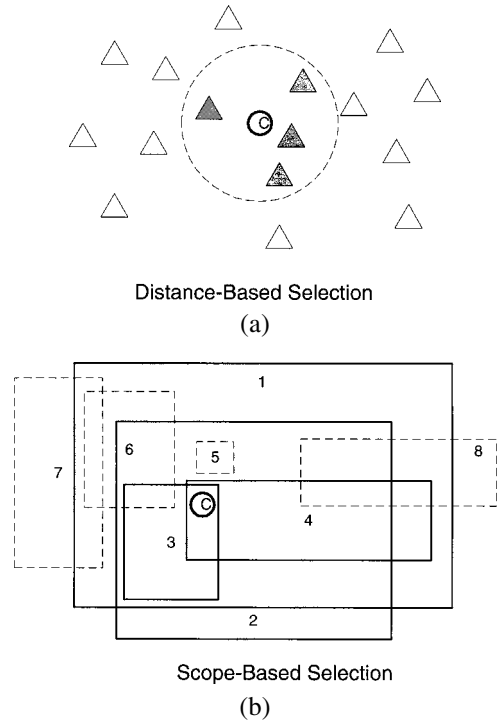

Figure 1. Proximity models for location-based selection.

proximity tends to decrease as we enlarge our notion of proximity, i.e., more things that we do not care about will be seen as being in our "proximity". In the *scope-based* model, each service is associated with a service scope that explicitly represents the usage context of that service as a region in physical space. The client selects those services whose scope includes its own location, i.e., a client is able to discover a service if it is located within that service's scope. To allow clients to introduce the proximity scale that most suits them, we also consider that they are able to limit discovery to services with a scope that is larger or smaller than some physical scale. The main characteristic of this model is that the correlation between context and proximity is assured. The services discovered, no matter how distant, are guaranteed to be relevant to the client's location.

The distance-based model places the focus on the location of the server providing the service, whereas the scope-based model places the focus on the geographical area defined for the service usage. These differences make each of the models the best approach for particular types of service. The distance-based model is the most adequate for services with a strong association with a specific point in space, typically services acting as electronic counterparts to real-world entities, e.g., a restaurant or a bus-stop. In these cases, finding the physically nearest server is usually the most natural type of query. On the other hand, the scope-based model is the most adequate for services with a usage that is geographically bounded but is not linked to any particular point in space, such as maps, weather forecasts or restaurant guides. We thus believe that it is relevant to support both these models, as each of them is the adequate approach for certain types of service. However, we have chosen to use the scope-based model as

the primary mechanism for associating services with location, and thus location-based services are always assumed to have an associated scope. This design decision has been motivated essentially by the possibility of expanding the range of proximity from a small scale to a large scale without necessarily increasing the number of selected services.

2.2. Scope model

The key mechanism used by the AROUND architecture for modelling the association between services and location is to associate each service with a service scope that explicitly expresses the usage of that service as an area in physical space. The expression of service scopes is achieved through a shared set of symbolic locations, known as location contexts.

Definition 1. A *location context* is a symbolic representation of an area in physical space that can be explicitly referenced by a global name and can be used across multiple networks and domains as a context for service discovery.

A service scope is thus the set of location contexts in which a service is registered. Servers, when making a service registration, will always include a reference to the particular location contexts in which they are registering the service. Similarly, clients, when searching for location-based services, will use one or more location contexts to indicate the spatial scope of their queries. Each location context is identified by a globally unique name with a format defined by the following URN namespace:

"urn:x-around:"<FQDN>":"<LID>

A Fully-Qualified Domain Name (FQDN) identifies an entity responsible for creating names, for guaranteeing their uniqueness and for providing or delegating their resolution. The Location Identifier (LID) unambiguously identifies a location context within the administrative domain specified by the FQDN, and can be arranged in a hierarchy of atomic names, e.g., "/library/room6". Location contexts are also associated with a type that indicates the nature of the represented location, e.g., hospital, or airport. This explicit knowledge about the characteristics of the represented locations allows applications to make important deductions about their current environment, such as the most appropriate service selection strategy or the types of service that the user may need most.

2.2.1. Relationships between location contexts

In our architecture, location contexts can be linked through unidirectional relationships whose goal is to enhance the process of service discovery by transforming location contexts from isolated service aggregations into components of a shared distributed service location space organised according to spatial criteria. Relationships between contexts establish a mechanism for the propagation of queries from a source context to a target context, resulting in a graph structure similar to that of federated trading architectures [12]. However, unlike links between federated traders, links between location contexts have a spatial semantics that determines the way queries are propagated in the graph and allows the search domain to be specified in spatial terms.

Our architecture employs two types of context relationship: containment and adjacency. The *containment* relationship reflects the spatial inclusion of the area of a contained context within the area of a container context, and defines a partial order over any arbitrary set of location contexts. Given the spatial semantics associated with containment, the relationship is transitive and the resulting graph is assumed to be acyclic. Each location context can have more than one containment relationship to other contexts, resulting in a lattice structure. We assume the maximum number of location contexts in a single chain to be restrained by its effect on users. A very deep hierarchy could become too complex to use and would necessarily involve location contexts with very little meaning to the user. The number of contexts in a chain should thus be fairly small, and typically correspond to only a few levels, such as room, floor, building, street, area, town and region. Another restriction results from a universal set of principles that guides our notion of space and constrains the containment relationships that may occur in the real world, e.g., it should not be possible to create a containment relationship from a context of type "Building" to a context of type "room". To guarantee this consistency, the spatial semantics of the trading graph is enriched with a set of restrictions on the establishment of containment relationships between certain context types.

Containment is employed as the central relationship of the architecture because of its key role in enabling the scope-based model of proximity. A containment relationship implicitly makes the services registered at the container context available for selection in the contained context. This is the opposite of the approach typically found in location systems, where "*if a located-object is a member of a particular domain, it must also be a member of all parents of this domain*" [9]. As an example, figure 2 represents a set of location contexts and their associated containment relationships. The horizontal arrows represent the registration of a set of services, A, B and C, in their respective contexts. The lower half of each context indicates the services that can potentially

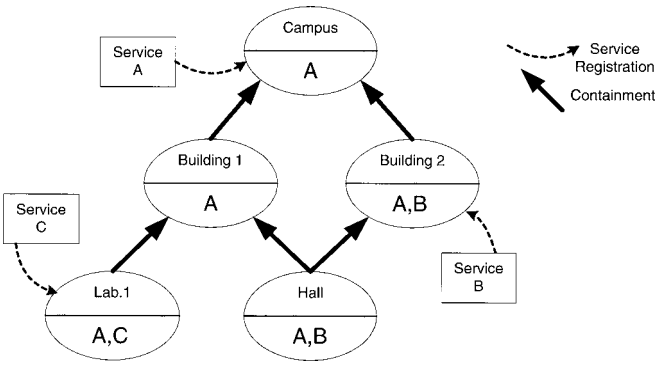

Figure 2. Service location through containment relationships.

be discovered at that context.[1] Service A, registered at the "Campus" context, is available everywhere, because all other contexts are directly or indirectly contained in the "Campus" context. Service A is thus associated with a large scope through a single registration at a high-level location context. On the other hand, service C, registered at the "Lab.1" context, is only available within that context, as "Lab.1" contains no other contexts. Thus, a client searching for services in the "Campus" context can only discover service A, whereas a client searching in "Lab.1" can discover services A and C.

Adjacency is the other type of relationship employed by the architecture, and expresses an immediate physical proximity between the areas of two location contexts. Adjacency allows a query to go from its base context to contexts that represent neighbour areas, thus allowing physically near services to be selected, even if out-of-scope. Assuming that their registration includes a location attribute, services can be selected from multiple adjacent contexts based solely on their distance to the client, thus enabling a distance-based model of proximity with flexible selection range. Another role of the adjacency relationship is to support rapid context changes by allowing clients to adopt a "pre-fetching" attitude in which they perform service selection in adjacent contexts before actually entering them. Typically, an adjacency relationship is established between location contexts of the same nature in well-structured places, e.g., rooms in a building, and the set of location contexts involved must form an anti-chain, i.e., none of its members should be comparable from the perspective of the containment relationship.

2.3. Functional model

The functional structure of the AROUND architecture comprises the *AROUND service*, the *contextualisation* process, and the *name service*. The main component of the system, the *AROUND service*, is a distributed service location infrastructure that maintains a repository of service information organised by location contexts, and is described in the next section. Contextualisation is the process that determines the location context that corresponds to the current physical position of a mobile device, called *base context*. This mapping of positions in physical space into contexts in a space of location contexts can be based on information obtained from multiple types of location sources, and thus the definition of specific contextualisation mechanisms is regarded as being outside of the scope of this work. Ideally, a combination of contextualisation functions should complement each other and provide a seamless positioning coverage, but the existence of multiple sources may also lead to conflicting indications on the current base context. When there are multiple base context indications, fusion is attempted, particularly by exploring potential containment relationships between those location contexts, but, if that fails, they are all passed to the application, which may choose to use them all as independent contexts for service discovery. This situation is a normal consequence of the one-to-many relationship between physical space and location contexts assumed by the contextualisation process of the AROUND architecture. Even though each location context is associated with a specific area in physical space, there is no assumption that a position in physical space is uniquely associated with a single location context, i.e., the AROUND architecture does not assume the existence of a single, absolute hierarchy of location contexts, and allows multiple spaces of location contexts to overlap on the same physical space. Assuming otherwise would imply, either being able to clearly associate administrative authorities with each location, or having some global authority capable of managing a global location model and providing a universal mapping. Since none of these assumptions seems very reasonable from the perspective of the system scalability, the AROUND architecture simply assumes that anyone is free to create a location context for any area in physical space and that the relevance of competing spaces of location contexts would be determined by the number and widespread use of their associated contextualisation mechanisms. As a result, mapping a position in physical space into a location context is a process that necessarily implies some administrative option regarding the particular set of location contexts to which positions are going to be mapped.

The *name service* resolves global location context names into references to specific AROUND servers at which they can be accessed, allowing a single name to correspond to more than one instance of the same location context. The requirements of a name service for the AROUND architecture are basically high availability, scalability and performance. We believe that these generic requirements are fully addressed by many name services described in the literature and also that the particular characteristics of this architecture in terms of locality of reference can make the best use of most of the replication and caching techniques that are normally used for achieving those goals. Therefore, we have not addressed the creation of any specific name service for the AROUND architecture.

Contextualisation and naming, together, allow mobiles to determine an access point to the system that can provide them with the services for their current location. Whereas contextualisation determines the base context of the device, naming resolves the name of that base context into the respective AROUND servers. Figure 3 depicts the functional model of a system based on the AROUND architecture and illustrates how an application interacts with the various components of the architecture to make its behaviour depend on the set of services available at each given location environment.

When the contextualisation process detects a change in the current location context, it notifies the application of that change, indicating the name of the newly entered context. The application will then contact the name service to resolve the name of new base context into one or more references to AROUND servers that provide access to that particular location context. Having obtained such references, the application sends a service query to the respective AROUND server and

[1] Whether or not services are actually available depends also on other factors, such as system policies (to be described in section 2.4.1).

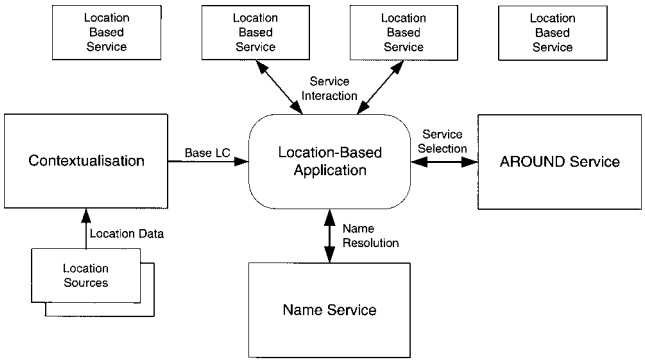

Figure 3. Functional model of a system based on the AROUND architecture.

obtains a number of references to location-based services that are specifically associated with the current location of the system. The application can then interact with those services and obtain the information it needs to reflect the particular information space of the new environment. This interaction can itself be location-dependent, in which case the application can communicate its location to the service and obtain an even more specific response.

2.4. AROUND service

The AROUND service is provided by AROUND server instances, each managing a set of location contexts. AROUND servers can be federated by creating relationships between location contexts maintained by different AROUND servers. These external links enable servers to share their respective service offer spaces and form a larger distributed context space in which queries may traverse multiple servers and possibly multiple administrative domains before being completed. The federation structure that results from this linkage mechanism contains two important properties: Firstly, all the services potentially relevant for a specific location are in a reduced set of AROUND servers. Secondly, from its base context, a client is able to reach all the other contexts that may provide it with services relevant for its location.

2.4.1. Querying the AROUND service

The interface supported by AROUND servers is largely based on the interface of the CORBA Trading service [12]. The basic parameters of a query are: (i) the name of the location context at which the query starts, (ii) the types of services to select, and (iii) a set of constraints that the attributes of the selected services must satisfy.

A query to the AROUND service may also include a set of *Policies*, which determine the location contexts that the query will traverse. Policies are thus the key mechanism for controlling the execution of queries that go through multiple contexts and servers. Even though the service location infrastructure provided by a federation of AROUND servers may be seen as a global service, it does not support global queries. Queries, thus, are always scoped, either explicitly, by policies included with each query, or implicitly by default polices defined at the AROUND service. The policies supported are basically the same as in CORBA Trading, but new spatially-based policies have been added to embody the spatial characteristics of the AROUND service: the *Query_type* policy defines the spatial direction of the query, i.e., whether to follow adjacency or containment links; the *Context-bound* identifies a specific location context as the limit for the query. The query terminates when it reaches the specified context or a context that is not contained within the specified context; and the *Type-bound* identifies a maximum location type for the query. The query terminates (without selecting the local services) when it reaches a location context with a type that cannot be contained in a context of type *Type-bound*.

The other query parameter is a *Preferences* expression that can be used to request a particular ordering of the set of matched services. Preferences are also based on CORBA Trading, but we added the new "*nearest <position>*" expression to support the ordering of services by their distance to the specified position. This feature, together with adjacency links and location attributes, provides the key mechanism for enabling service selection using distance-based proximity.

2.4.2. Replication

The AROUND architecture supports two distinct replication mechanisms. *Replication of location contexts* consists of having multiple instances of the same context managed by different AROUND servers, and plays a key role in the availability of the system. Firstly, it provides a fault-tolerance mechanism, as a location context is no longer associated with a single server. Secondly, it can be used to support load balancing in contexts with intensive use. Finally, an appropriate distribution of replicas can also increase the availability of the system in the face of network partitions. *Service Caching* allows an AROUND server to cache the service registrations from a location context managed by some other AROUND server. The caching mechanism benefits considerably from the locality of reference that is inherent to the scope model on which the architecture is based, as the set of cacheable location contexts in an AROUND server directly results from the links of the location contexts maintained by that server and not from arbitrary user requests. The cache of an AROUND server will typically include location contexts to which it maintains some containment or adjacency relationship, thus allowing it to answer locally to queries that would otherwise have to be sent to other servers.

Both replication mechanisms assume a loose, or converging consistency model. The adequateness of loose consistency as a model for replicating service registrations is based on the idea that inaccurate information about services can either be detected (e.g., the service is no longer available) or will have no major consequences (e.g., there is a new service available but some parts of the system still do not know about it). The scalability of service caching is also based on two key assumptions about the patterns of service registration. The first is that the read/write ratio increases at the higher levels of the hierarchy of location contexts. The dynamic of service registrations is assumed to be inversely proportional to the size of the respective scopes, and thus the set of services

registered with large scopes is expected to change slowly. The other assumption is that service registrations will be reasonably distributed among the various levels in the containment hierarchies, i.e., there will be some administrative control capable of promoting a balance between the higher visibility provided by a registration at a higher level and the administrative cost associated with that registration.

2.5. Implementation

In order to provide the basis for the evaluation of the system, we have created a prototype implementation of the AROUND architecture. Considering the nature of this work, we have decided to focus on a rapid and flexible prototyping of the AROUND service functionality rather than on a low-level implementation optimised for performance. We have chosen to developed a JAVA implementation based on the Jini framework [15], in which Jini lookup services are used as service registries, and service definitions, e.g., service types and attributes, are based on the Jini programming environment. The system has been developed using JDK 1.2.2 and the SUN reference implementation of Jini (version 1.0).

3. Applications

In order to evaluate the feasibility of the approach and gain some insight into its implications, we have developed a number of prototype systems based on the AROUND architecture. We will now describe our two major prototypes: the AROUND client and the Hypergeo portal.

3.1. The AROUND client

Within the context of the AROUND project [1], we have developed a test-bed in the town of Guimarães for validating the architecture. This case study has included the creation of a generic service infrastructure, with multiple AROUND servers, location contexts and location-based services, and the development of multiple client applications over that common infrastructure.

The service infrastructure was based on an heterogeneous set of location contexts, ranging from rooms at the University campus to areas of various sizes in the town and its surroundings. Since the case study was thematically focused on applications for assisting mobile users with useful travelling and guidance information, and in particular public transportation information, these location contexts were populated with location-based services of the following types: a *BusInfo* service that gives information about the bus services for a specific area; a *BusStop* service that acts as an Internet counterpart to the information typically available at real world bus-stops; a *Map* service that provides maps in several formats for specific areas; a *Weather* service from which several formats of weather forecast can be obtained; and a *SpatialInfo* service that provides structured information about the current location, e.g., postal code, address, or a simple description.

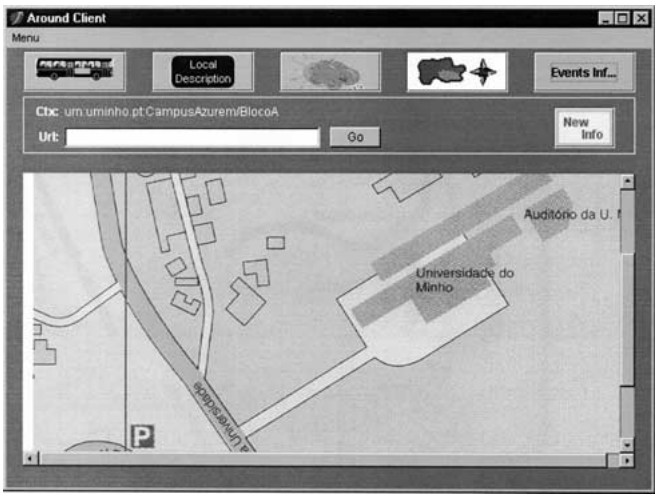

Figure 4. AROUND client application.

Some of these services, e.g., *SpatialInfo*, are available at various levels of the hierarchy of location contexts with various degrees of specificity. Others are available at specific types of location context, e.g., the *BusStop* service can typically be found in contexts representing small town areas.

With this prototype infrastructure operational, we have begun to explore the use of location-based services for the creation of location-dependent systems. Our main application, simply called AROUND client [11], was engineered for a medium-size handheld device equipped with GSM or IEEE 802.11 connections, and is represented in figure 4.

The application was designed as a flexible client of the AROUND architecture, capable of enabling many different patterns of information access. To achieve this flexibility, the application core supports only generic functionality, such as contextualisation and user interface, whereas specific functionality is encapsulated in thematic modules that act as dedicated sub-applications within the framework provided by the overall AROUND client. Each module may have its own behaviour in terms of the service types to use, query scoping, and service selection criteria. Given the thematic focus of the case study, the AROUND client has been tailored as a travel assistant by creating a set of modules that provide the user with complementary information about visited locations, more specifically *Transportation*, *Local Description*, *Weather*, *Maps* and *Events*. The user can activate and select modules by clicking the icons at the top of the application screen. The HTML output of the currently selected module is displayed in the central display area. When a new context is entered, all the active modules request the services they need, interact with those services, and generate a new location-dependent output, so that when they are selected they are able to provide immediate results. The availability of new information is signalled to the user by flashing the respective module icon on the top of the screen, and in the case of the selected module, by activating the "*new info*" button, which indicates to the user that the currently displayed information is no longer correctly contextualised. The shared application environment makes prototyping of new access

patterns easier as modules can share the common functionality of the application. Even though we have not implemented such functionality, the dynamic downloading of new modules would be a normal extension to our current implementation.

3.2. The Hypergeo portal

The Hypergeo project [8] has developed a web portal system for providing mobile users with several types of tourism related information. The system, which has been deployed in Toulouse and Karlsruhe, allows a person equipped with a wirelessly connected PDA to access several types of location-dependent information. At the exception of maps, which have their own module, most of that information is obtained using the AROUND architecture. When receiving a request for information associated with a location, the portal uses the AROUND architecture to select services with relevant information about the current location of the user, and then accesses those services to generate a personalised page with location-dependent content. Figure 5 represents the components of the Hypergeo Information Server that implement this functionality, and in particular the Hypergeo LBS component, which in this case plays the role of the client of location-based services.

The User Profile Manager (UPM) maintains information about registered users, including the user's identity, an history of past positions, preferences, and configuration. Information about the location of the user is managed by two different components. The Position tracker component parses and validates GPS positioning information sent over the mobile GSM network and updates the corresponding user profile. The Location Trends Extractor combines information about speed and sinuosity of the user with contextual data, such as transport networks, to derive a fuzzy reasoning about conjectured activity (e.g., shopping) or transport characteristics (e.g., on a train). Location-based services are not represented as part of the Hypergeo Information Server because, although they have to register with the AROUND service established for the Hypergeo server, they are assumed to be provided by third-party entities. The services created provide information about the Weather, Restaurants, Hotels, Monuments, Points of Interest, Events, YellowPages and Local Description.

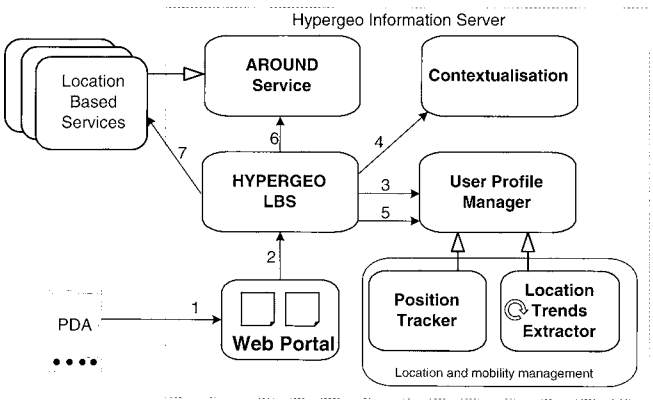

Figure 5. Architecture of the Hypergeo infrastructure.

The sequence of interactions starts when a client sends a request, as an URL, to the Web portal (1). When the portal identifies a request as being for local information, it passes the request to the Hypergeo LBS component (2). The LBS will then request from the UPM (3) the current position of the user and from the contextualisation component the corresponding base context (4). Before formulating a query to the AROUND service, the LBS goes back to the UPM and obtains further information about the user (5), particularly the user preferences and location trends. This information plays a key role in the behaviour of the LBS by determining the type of information needed and how it is going to be presented to the user. The objective is to create a personalised content that reflects the combination of the user interests with its current context (place, time of the day, conjectured activity). After selecting the services it needs from the AROUND service (6), the LBS component interacts with those services (7) and generates an XML output that is then sent to the Web Portal, where it will be formatted according to the characteristics of the mobile device.

4. Evaluation

Our evaluation of the AROUND architecture has been essentially qualitative and oriented towards the understanding of the main issues raised by the use of this particular approach as a generic mechanism for enabling location-dependent systems. We first analyse some of the main issues raised by the case-studies, and then discuss the system from the perspectives of openness, scalability and security.

4.1. Analysis of the case studies

The two prototypes created have demonstrated the practical application of the concept of location-based services in developing location-dependent applications, and allowed us to gain a valuable insight into the implications of developing systems based on the AROUND architecture, into the nature of the applications that the architecture can support, and into the advantages and disadvantages of the approach. The AROUND case study has mainly exercised the creation of a generic infrastructure of location-based services, and the development of multiple location-dependent applications on top of that common service infrastructure. The Hypergeo case study has exercised a more focused use of location-based services within the framework of a server-side portal and showed how the openness of the AROUND architecture can be used to extend the functionality of a system that would otherwise be self-contained. The overall results suggest that location-based services can effectively be used as a generic approach for supporting the development of location-based applications, but have also raised the following issues.

Definition of location contexts. Regarding the definition of location contexts, it has become evident that such definition is much simpler for location contexts indoors and in urban areas than it is in areas of reduced human presence. In the latter

case, it is often difficult to clearly identify suitable zones to be used as symbolic locations, i.e., areas whose boundaries are easy to identify. Our typical approach has been to rely on existing administrative divisions of space, such as postal codes, but this approach alone can lead to very shallow hierarchies and to areas that do not have enough symbolism to function as location contexts.

Contextualisation. The most obvious limitation in the contextualisation process concerns the inevitable existence of overlaps between location contexts, meaning that a mobile device can effectively be located in two location contexts at the same time. This is a generic issue with symbolic models, but in the case of the AROUND architecture there is also the possibility of having multiple independent spaces of location contexts overlapping on the same physical space. Despite assuming that the contextualisation process may result in the indication of more than one base context, the AROUND architecture does not include any mechanisms for handling such situations, except when one of the contexts is contained in the other, and thus it is possible to select the most specific of them. All other situations must be handled by the client application without any assistance from the contextualisation mechanisms. Our development of client applications has shown that this is not usually a satisfactory solution, as client applications have no access to the inner functionality and location sources of the contextualisation process, and thus have no means to make an informed decision regarding the relevance of the various base context indications. Their only option is either to use them all or to require user assistance.

User interface. The development of interactive applications has raised several user interface issues. Some of them are common to context-aware applications in general [3]; e.g., how to make the user interface reflect context changes or how to combine explicitly retrieved information with context-aware information, but others were prompted directly by the use of location-based services. One of those issues has been how to present the user with its options in terms of the service selection process, and particularly in what concerns the spatial scoping of queries. An adequate representation of the space of location contexts should allow users to easily perceive the currently available alternatives in terms of the spatial scoping of queries and take advantage of the multiple proximity models supported by the AROUND architecture. However, the lattice structure of the location context space is not conveniently represented by common navigation metaphors. Another user interface issue concerns the representation of contextual factors associated with information. In our applications, the user does not directly select his sources of information, and is even unaware of which sources are being used. Furthermore, an application may combine information obtained from multiple sources and present it as a single piece of information. This prevents the user from assessing contextual factors that are normally associated with information, such as its source, the way it is presented, or where it is referenced. Without such elements, the user may fail to evaluate the background and trustworthiness of the available information.

4.2. Openness

The AROUND architecture has been designed from start with openness as one of its main motivations. Our case studies, were thus based on the implicit assumption that their various elements, i.e., services, client applications, and location contexts, could all be independently developed and managed. As a result, both prototypes showed a considerable potential for evolution in terms of the sources of information used, the types of information supported, or the underlying infrastructure. All these extensions were possible, and simple, because the abstractions on which the system design is based avoids hidden dependencies between the various components of the system, providing a clear separation of concerns between key functionality, such as information provision, location modelling, service location and application development.

However, the evaluation of the prototype as an open architecture has necessarily been compromised to some extent by the lack of established standards for most of the system elements. This has forced us to create, specifically for this purpose, all the elements in our case studies rather than simply integrating truly independent components. If, as in our case studies, the creation of a single location-dependent system implied the creation of a service infrastructure to serve it, then the potential advantages of an open approach would become blurred when compared with the much more pragmatic approach of a vertical system. We believe, however, that this limitation is merely transitional and that in the near future the adoption of open standards for automatic information exchange, on which multiple industrial groups are now working [14,16,18], will lead the Internet to better approximate a truly service-based infrastructure. When these yet missing pieces start to be in place, the advantages of an open approach will become more and more evident.

4.3. Scalability

The effects of scale are a major concern for any system aimed at the Internet environment. In our architecture the potential growth in the number of queries, services, users and locations are handled mainly through the mechanisms of *distribution*, *replication* and *caching*.

Distribution is the key scalability technique used in the AROUND architecture. The service is distributed among multiple AROUND servers, each managing only a small portion of the overall space of location contexts. This distribution reduces the number of requests that must be handled by each server, allows different parts of the system to be managed by different administrative entities, and allows the information to be placed where it is most likely to be needed. Despite being a global system, there are not any global variables or system-wide states, except naming, and there is not any "root" location context. Each component of the architecture can function with only a reduced knowledge of the other components of the system. Furthermore, the particular scope model

on which the architecture is based introduces a strong locality of reference, as a result of which, interactions within the system typically involve only a reduced and stable set of elements that are likely to be close to each other.

Replication of location contexts is another important mechanism for the scalability of the system. It allows replicas of a location context to be placed on different AROUND servers, thus supporting load-balancing between those servers. The caching of service information to lower levels in the hierarchy allows the system to scale in the number of location contexts and respective network distance. It reduces the number of servers that need to be contacted to satisfy each query and removes considerable load from the servers that maintain the higher level location contexts.

4.4. Security considerations

Even though we have not actually implemented any security mechanisms we have conducted a careful analysis of the security services needed by the different components of the architecture and of how they could be introduced. The most obvious requirement is to be able to guarantee the correct operation of the system by protecting communication between the various system entities and by controlling resource usage. Additionally, we have also identified the need to integrate the services' security policies with AROUND service queries. When submitting a query to an AROUND server, a client should be able to specify its access authorisations in such a way that the AROUND server would only reply with services to which the client is able to access. Otherwise, clients would risk having to go through multiple successions of queries to the AROUND service and service access attempts until reaching a service without a limiting access control. Addressing this issue requires the ability to represent in service registrations the access control criteria of the services and to evaluate the authorisation of any potential clients of that service. Another issue is how to evaluate the integrity and trustworthiness of the information stored in the AROUND service, i.e., "is the service really run by the claimed entity and is that entity a trustworthy source?". All these issues can be addressed by standard security techniques for basic services such as authentication, data integrity, and rule-based access control, but their introduction without jeopardising the openness of the system remains an open issue at least until the use of horizontal security mechanisms becomes widespread in the Internet.

5. Related work

The motivation for this work has largely emerged from systems such as GUIDE [4], developed at Lancaster University, or CyberGuide [10] developed at the Georgia Institute of Technology. We share with these classic examples of location-aware systems the aim of building applications that react to changes in their location. What separates our approach is the use of location-based service discovery as the abstraction for modelling the association between resources and location. More than proposing a vertical approach for addressing the requirements of a particular application, we aim to provide a new horizontal approach that is capable of bringing the benefits of service-based architectures to the development of location-dependent systems.

A whole range of new info-mobility services with location-based content is currently being launched within the context of cellular networks and allow customers to access information specific to their location via their handheld devices and WAP enabled mobile phones. These services typically take into account the current position of the mobile terminal to provide general interest information, such as nearby restaurants or local traffic reports. What mainly distinguishes our approach is where to introduce location-dependency. In those services, location-dependency is exclusively handled by the server, and does not contribute for service selection. As a consequence, each service must be pre-defined, must be able to deal with location information, and must be able to provide global coverage, i.e., it should be able to provide a convenient answer to requests from any of the possible locations of the client application. In our approach, location information is mainly used for selecting a relevant server, which may itself be location-dependent. Servers do not need to provide global coverage, as multiple services can be created to address the particular needs of each location.

Our work has also much in common with a great number of service location frameworks that are now available. The Service Location Protocol (SLP) [6] is proposed for service location in LANs under a single administrative domain. Jini [15] is a Java-centric technology that aims to support the association of groups of autonomous devices and software components into a dynamic system. The Universal Plug and Play (UPnP) [17] technology aims to simplify the transparent interconnection of appliances, PCs and services by leveraging Internet technology. One characteristic that all these systems have in common is a discovery mechanism strongly based on network proximity. We aim to build a system in which proximity is effectively based on physical locality, and is not constrained by administrative or network boundaries.

The CORBA Trading Object Service [12] supports a distributed service offer space and provides a model for sharing the offer spaces from various traders by creating links between them. Even though it is possible to select services based on location attributes, a link from a trader to another is simply a routing path for queries, and nothing else can be assumed about the nature of those interconnections. As a consequence, this model provides a weak specification about how search domains are specified, about when searches terminate and about where services are published [13]. The Service Discovery Service (SDS) [5] is a globally-distributed architecture for wide-area service location. SDS aims to support the selection of a server anywhere on the Internet, with location being just another potential search criterion. What mainly distinguishes the AROUND service is that the location space is itself organised by spatial criteria and is optimised for supporting spatially scoped queries. As a result, global queries

are not necessary for selecting the services that are relevant for a particular location.

The Cooltown project [2] is a platform for enabling a systematic correlation between the entities in the physical world and their respective web pages. The architecture proposed by Hodes in [7] supports the discovery of services associated with a particular physical location by separating the role of discovery, performed by a beaconing daemon, from the role of service selection, supported by a specific server. We share with both these systems the strong role played by physical location on the process of service selection. However, their discovery mechanisms are based on isolated infrastructures that support service selection within a restricted domain, whereas we propose a concept of location-based service based on abstract scopes that can scale to large areas and a globally available service location infrastructure.

6. Conclusions

6.1. Future work

In this section, we describe what we believe to be the main points for further research within the context of this work.

The current approach to the management of location contexts and their federated operation, essentially based on simple policies, is, admittedly, less than ideal. Also, in a realistic scenario of multiple overlapping location hierarchies managed by different authorities, the establishment of relationships modelled by purely spatial criteria may become a serious limitation to the collaborative development of multi-domain location context spaces. The administrative model should thus be able to support flexible relationships that, while keeping the essence of the spatial nature of the links, could also accommodate administrative decisions regarding the sharing of services between contexts. While these administrative aspects were not of great significance for the proof-of-concept infrastructure that we have developed, they would be of paramount importance in a widely deployed platform, and should thus be a key topic for further research. The creation of effective solutions in this area may, however, be subjected to the ability to foresee the potential business models that may arise in a service infrastructure of this nature.

We also aim to explore new application models that can better exploit the basic trade-off between generality, i.e., the application ability to satisfy many different information needs, and specificity, i.e., the application ability to adopt the most adequate behaviour for a particular task. Generality is important, particularly for applications aimed at providing information, as otherwise the diversity of people information needs would lead to a myriad of small applications each providing some very specialised content. Specificity is also important because it allows the knowledge about the selection and use of specific services to be incorporated into the application. This in turn allows the application to support in a pro-active way many of the service selection operations, bringing important benefits in terms of enhancing those operations, making them transparent to the user, and also additional optimisations such as background pre-fetching. We will therefore explore new approaches for combining these two elements in the design of client applications for location-based services.

Other points for further research include the best practices for the definition of location contexts, enhancements to the contextualisation model, and implementation of security services.

6.2. Concluding remarks

The work described in this paper has sought to investigate the use of location-based service selection as an open and generic approach to support the development of many types of location-dependent systems. To enable that approach, the novel AROUND architecture has been proposed as a service location infrastructure capable of allowing applications to select Internet services that are specifically associated with their current location. The results obtained from a number of case studies suggest that location-based services can effectively provide an adequate abstraction for the development of location-dependent systems. The resulting prototypes exhibit significant potential for evolution in terms of new information sources, new locations or the underlying infrastructure. The main limitation in terms of our initial objectives has been the current lack of open standards for automatic information exchange, without which the potential advantages of an open approach become compromised. We believe, however, that this limitation is merely transitional and that in the near future the adoption of such standards will allow the important benefits of openness to be fully realised.

Acknowledgements

Work on the AROUND architecture and prototype was carried out as part of the AROUND project, supported by the Portuguese Government Praxis programme (PRAXIS/P/EEI/14267/1998). Work on the Hypergeo portal was carried out as part of the Hypergeo project, supported by the European Commission IST programme (IST-1999-11641). Special thanks are also due to all the team members in both projects, and particularly to Filipe Meneses and Hélder Pinto.

References

[1] AROUND project web site, `http://www.dsi.uminho.pt/get/around` (2000).

[2] D. Caswell and P. Debaty, Creating web representations for places, in: *Handheld and Ubiquitous Computing*, Vol. 1927, eds. P.T. Gellersen and H.W (Springer, Berlin, 2000) pp. 114–126.

[3] K. Cheverst, N. Davies, K. Mitchell and A. Friday, The role of connectivity in supporting context-sensitive applications, in: *HUC99, International Symposium on Handheld and Ubiquitous Computing*, Karlsruhe, Germany, ed. H.-W. Gellersen (1999).

[4] K. Cheverst, N. Davies, K. Mitchell and A. Friday, Experiences of developing and deploying a context-aware tourist guide: the GUIDE

project, in: *Sixth Annual International Conference on Mobile Computing and Networking*, Boston, MA (2000) pp. 20–31.
[5] S.E. Czerwinski, B.Y. Zhao, T.D. Hodes, A.D. Joseph and R.H. Katz, An architecture for a secure service discovery service, in: *Fifth Annual ACM/IEEE International Conference on Mobile Computing and Networking, MOBICOM99*, Seattle, WA (1999) pp. 24–25.
[6] E. Guttman, C. Perkins, J. Veizades and M. Day, Service location protocol, Version 2, RFC 2608 (1999).
[7] T.D. Hodes, R. Katz, E. Servan-Schreiber and L. Rowe, Composable ad hoc mobile services for universal interaction, in: *3rd ACM/IEEE International Conference on Mobile Computing and Networking, MOBICOM97*, Budapest, Hungary (1997).
[8] Hypergeo Consortium, Easy and friendly access to geographic information for mobile users, `http://www.hypergeo.org/` (2001).
[9] U. Leonhardt, Supporting location-awareness in open distributed systems, Ph.D. thesis, Imperial College of Science, Technology and Medicine, University of London (1998).
[10] S. Long, R. Kooper, G.D. Abowd and C.G. Atkeson, Rapid prototyping of mobile context-aware applications: the cyberguide case study, in: *ACM/IEEE International Conference on Mobile Computing and Networking, MOBICOM96*, Rye, NY (1996) pp. 97–107.
[11] F. Meneses, A. Moreira and R. José, How do I design a location-dependent application?, in: *SPIE's International Symposium on the Convergence of Information Technologies and Communications, ITCom 2001*, Denver, CO (2001).
[12] OMG, CORBA services: common object services specification, Technical Report (1998).
[13] G. Outhred and J. Potter, An enterprise trader model for DCOM, in: *IFIP/IEEE International Conference on Open Distributed Processing and Distributed Platforms*, Toronto, Canada, eds. J. Rolia, J. Slonim and J. Botsford (1997) pp. 63–73.
[14] Parlay group web site, `www.parlay.org` (2001).
[15] SUN Microsystems, Jini web site, `http://www.sun.com/jini/` (1999).
[16] UDDI, Universal description, discovery and integration of bussiness for the Web, `www.uddi.org` (2001).
[17] Universal Plug and Play Forum, Universal Plug and Play web site, `http://www.upnp.org/` (2000).
[18] W3C, Web services activity, `http://www.w3.org/2002/ws/` (2002).

Rui José is an Assistant Professor at the Department of Information Systems, University of Minho. He holds a D.Eng. (Licenciatura) in informatics and systems engineering, and an M.Sc. in informatics (specialisation in distributed systems, computer comunications, and computer architectures) both from University of Minho. In 2001, he obtained his Ph.D. in computer science (field of distributed systems) at Lancaster University for his work on location-based services. At the University of Minho, he participated in several funded research projects in the area of mobile computing, including the AROUND and Hypergeo projects, and he is currently the principal investigator of the Vade project, which explores new mobile application models combining services from mobile network operators with services from the local environment. Rui José is since 1997 a member of ACM and its Special Interest Group on Mobility of Systems, Users, Data and Computation (SIGMOBILE).
E-mail: rui@dsi.uminho.pt

Adriano J.C. Moreira is an Assistant Professor in the Department of Information Systems, University of Minho, since 1996. He received the Licenciatura degree in electronics and telecommunications engineering and the Ph.D. degree in electrical engineering, respectively in 1989 and 1997, from the University of Aveiro, Portugal. He has been a voting member of the IEEE 802.11 working group (Wireless Access Method and Physical Layer Specification) where he participated in the specification of the infrared physical layer. His research interests are in indoor optical wireless transmission systems, wireless local area networks and mobile and context-aware computing. He participated in many research projects, more recently in the AROUND – Supporting Location-Based Internet Services and the Hypergeo – Easy and Friendly Access to Geographic Information for Mobile Users. He is a member of the IEEE Communications Society.
E-mail: adriano.moreira@dsi.uminho.pt

Helena Rodrigues was born in Portugal. She did her undergraduate studies and received her Mestrado degree (M.Sc.) in computer science at Universidade do Minho, Portugal. She received her Ph.D. degree in computer science from the University of Kent, UK, in 1998, with her Ph.D. Thesis entitled "Cyclic Distributed Garbage Collection". Since 1998 she is an Assistant Professor at the Department of Information Systems, University of Minho. Her current research interests are in distributed systems, distributed applications, object-oriented technology and mobile computing. Recently, she has been a member of the research team of the AROUND project – Supporting Location-based Internet Services, ended in 2001 and funded by the Portuguese government, and also a member of the research team of Hypergeo – Easy and Friendly Access of Geographic Information for Mobile Users, an European funded R&D project (IST-1999-11641), ended in 2001.
E-mail: helena@dsi.uminho.pt

Nigel Davies holds a B.Sc. and Ph.D. in computer science, both from Lancaster University, UK. Having completed his studies he was a visiting researcher at the Swedish Institute of Computer Science (SICS) before returning to Lancaster in 1994 to help create the University's Mobile Computing Group. He has since managed numerous projects at Lancaster, including the MOST and GUIDE projects, both of which have been widely reported on in the academic literature and the popular press. During 1999/2000 he spent a year as a visiting researcher at Sony's Distributed Systems Lab in San Jose working on integrating mobile devices with home AV networks. In recognition of his work in establishing Lancaster as a major research center in the field of mobile computing he was awarded a personal chair in the Computing Department in 2000. He has participated actively in the mobile computing research community and has served in a number of roles including Program Chair for IEEE WMCSA 2000, tutorials co-chair for Mobicom 2000 and demo chair for Mobicom 2001. He currently divides his time between Lancaster and Tucson, AZ where he is an Associate Professor at the University of Arizona.
E-mail: nigel@comp.lancs.ac.uk