DEVELOPMENT OF LOCATION-AWARE APPLICATIONS

The Nidaros framework

Alf Inge Wang[1], Carl-Fredrik Sørensen[1], Steinar Brede[2], Hege Servold[3], and Sigurd Gimre[4]

[1]*Dept. of Computer and Information Science, Norwegian University of Science and Technology, NO-7491 Trondheim, Norway,* [2]*Telenor R&D, NO-7004 Trondheim, Norway,* [3]*Bekk Consulting AS, NO-0102 Oslo, Norway,* [4]*CIBER Norge AS, NO-0103 Oslo, Norway*

Abstract: This paper presents the Nidaros framework for developing location-aware applications that provide location dependent functionality based on the current location of the user. The framework can be used to develop location-dependent advertisement, city guides, guides for tourist attractions, etc. The framework consists of three main components: *A runtime system* that manages user locations and the interaction with the user clients; *a creator tool* that is used to map information and multimedia content to locations; and *a logging tool* that is used to log the movement of users to monitor the interest for certain locations. The paper also describes an implementation of a location-aware tour guide for the Nidaros Cathedral in Trondheim that can run on different mobile devices. Further, the paper describes experiences from installing, configuring, and running a location-aware tour guide in a real environment. A demonstration of the tour guide was tested on PDAs and mobile phones.

Keywords: Context/location, case studies and experience, applications, tour guide

1. INTRODUCTION

In 2005, it is estimated that there will be more than 1.5 billion wireless subscribers worldwide. Although mobile computing gives challenges to the application developer like handling wireless networks, heterogeneity, limited screen size, input device CPU, memory and battery (Satyanarayanan, 1996), it gives possibilities to develop new types of applications. One such type is location-aware applications. Location-aware applications are useful for

several reasons: Firstly, the limited screen size of mobile devices can be better utilized by providing user interfaces that are related to the context of the user. By using the location, the parts that are not relevant to the current location can be left out. Secondly, the user experience can be improved by providing the user with information and interaction that are relevant to the current context. Here the context is necessarily not only location, but can also be time, weather, temperature, altitude etc. Thirdly, a system can collect context information from users to further improve the location-aware system. For instance in a tour guide system for an art gallery, the system can log which paintings the visitors spend most time by. This information can be used to publish additional information about the most popular paintings in the location-aware guide system.

Several location-aware systems for tour guiding have been developed like the Lancaster GUIDE (Cheverst et al., 2000), CyberGuide (Abowd et al, 1997) and MUSE (Garzotto et al., 2003), but most of these systems are tailored for a specific location-aware application. In 2003, the Norwegian University of Science and Technology (NTNU) started together with Telenor, the largest telecom company in Norway, to develop a general framework for creating, running, and analysing mobile location-aware systems. The motivation for this work was to enable rapid development of location-aware systems that can provide the user with information or multimedia content dependent on the user location. Another important aspect of the framework was to enable support for different mobile clients with different characteristics from the same system. From similar projects, we have found that location-aware systems use various client devices from rather big portable PCs down to small PDAs (Sørensen et al., 2003). Also we noticed that some location-aware systems use customized hardware to get the required characteristics. In addition, the evolution of mobile devices makes it necessary to be able to adapt to future devices with new and useful capabilities. From talking with people managing a PDA-based tour guide (not location-aware) at the Nidaros Cathedral, we understood that theft was a serious challenge. Letting people use their own mobile equipment (like mobile phones) for such services was found to be very interesting.

Another shortcoming for many of the existing systems is that they are tailored to support only one type of positioning technology like GPS, GSM, Bluetooth, IR or WLAN positioning. We used in the Nidaros framework a location server to fetch the user positions from various sources and to send this information back to the system when needed. The location server examines position technologies available to return the most accurate position.
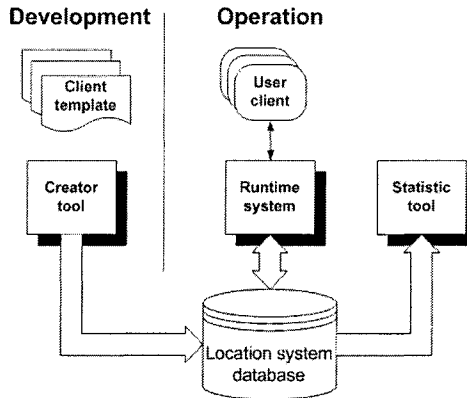
Development | **Operation**

Figure 1. The Nidaros framework for development of location-aware applications

Figure 1 shows an overview of the Nidaros framework for development of location-aware applications. The framework covers development and operation of a location-aware system. The development phase is supported in the framework with a creator tool to add and map location-dependent content, an XML client-server interface and templates for creating clients. A runtime system and a statistic tool support the operation phase. In addition the framework make use of a location system database and a location server.

2.   THE FRAMEWORK

This section presents the main components in the Nidaros framework.

2.1   The Development Phase

The development phase of the framework is supported by three components: A creator tool, client templates, and the XML interface between client and server.

The Creator Tool

The creator tool is a system for managing information and multimedia content related to locations that are to be part of a location-aware application. The tool provides a simple user interface that facilitates describing areas hierarchically into maps, zones and objects as a tree structure. A map represents the whole area of the location-aware application. A map can be divided into one or more zones that represent some specific

areas in the map. Within a zone, you can add several objects. These data objects typically represent physical objects and may contain information, audio-clips, video-clips and similar. The maps, zones and objects are mapped to a local Cartesian coordinate where the x- and the y-axis are represented with a 32-bit integer. The coordinate system can be mapped to various geographical positioning systems and makes the framework independent of the actual coordinate systems. By using a Cartesian coordinates in the application, it is easy to visualize maps and objects on the screen of the client device. In the local coordinate system, a map is represented as a rectangle. A zone is represented as a polygon of four coordinates. An object is represented by a specific coordinate and with a hotspot area represented by a polygon similar to a zone. The hotspot area is used to determine if a user is close to an object to trigger some location-aware event.

In the tour guide application we developed for the Nidaros Cathedral (described in Section 3), the map represented the whole cathedral, the zones represented the main parts of the cathedral, and the objects represented physical objects like alters, the pipe organ, paintings, etc.

The creator tool offers a user interface for how various devices like laptops, PDAs and mobile phones are mapped to the local coordinate system. The mapping identifies the type of device by name, the size of the device, an offset coordinate (X_{offset}, Y_{offset}), a rotation coordinate $(X_{rotation}, Y_{rotation})$ and a rotation angle R_{angle}. These data are used to compute the transformation from real world coordinates to the local coordinate system.

The creator tool can be used to graphically visualize the results of using the tool. The visualization shows the map you have created with named zones and objects. The zones and the objects are shown in different colours. Hotspot areas are shown for the objects.

The Client Templates

The framework provides a template for reducing time to implement clients for location-aware applications. The template is intended for clients that run Macromedia Flash applications. Flash makes it possible to make advanced and dynamic interfaces, and can run on various operating systems like MS Windows, Mac OS, Linux, Unix and Pocket PC. The Flash player is also capable of audio and video playback.

The template offers the basic functionality needed to implement location-aware clients. The template includes functionality for server communication using XML, graphical highlighting of zones and objects, management of hotspots (event-triggered actions), and initialization of multimedia playback. When the template is used, the developer has only to design the user

interfaces and possibly additional functionality if wanted. The template makes the implementation of clients easier and possible misunderstandings of the XML-interface with the server are avoided.

The XML Interfaces

The Nidaros framework provides an open XML interface to the runtime system that makes it possible to create clients for different devices using different technologies like Flash, Java 2 Micro Edition, HTML, .NET compact framework, etc. The only requirement is that the client is capable of managing XML communication and adheres to the specified XML interface. The client application can send a request to the runtime system in several different ways, but it has to follow a predefined XML syntax. The response will likewise follow a predefined syntax. If the client application requires downloading of multimedia content, this is done by an HTTP-request to a file server (see Section 2.2).

The root element in every XML request is a *locationRequest*. This root element can contain several different elements depending on what kind of information that is wanted. Every *locationRequest* must contain an element called *mac* that holds the mac address of the client device. The runtime server needs the mac address to identify and find the position of the user. The following information can be requested from the server: **Position** will get the current position of the user, **simulatedPosition** will get a simulated position of the user (useful for demonstration and testing), **friendsPosition** will return positions of other users registered as friends, **dynamicInfo** will return objects that the user is within the hotspot area of, **tracks** will return some predefined routes that the user might want to follow, and **messages** is used for sending and retrieving messages between users. When a request is sent to the server, the client will get a response from the server with the necessary information depending on the request type.

2.2 The Operation Phase and Runtime System

The main components used in the operation phase are the user clients, the runtime system and the location system database. The user clients are developed using the client template and the XML-interface described in Section 2.1. The runtime system is the heart of the Nidaros framework that brings location-aware applications alive. The runtime system manages the information in the database including maps, zones, objects, users, and etc. The main task of the runtime server is to feed the clients with correct information according to the client position.

Figure 2 shows the physical view of the runtime system. The runtime system supports several types of clients and identifies three client types we have implemented. The figure shows that wireless LAN is used between the server and the clients, but also other types of wireless networks have been used. Currently, our mobile phone client uses GPRS for communication between the client and the server. The runtime system itself consists of four main components described below.
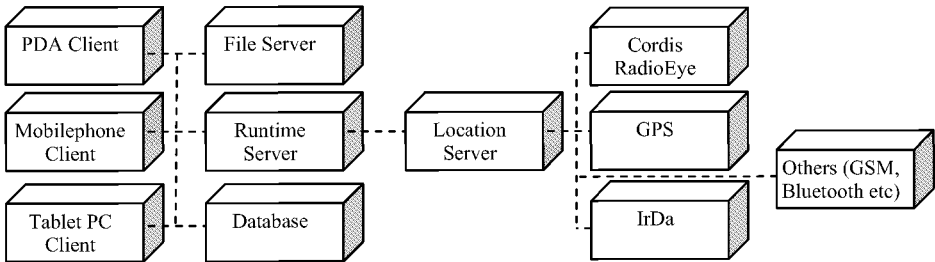

Figure 2. The Physical view of the runtime system

The **file server** stores media files accessible for mobile clients. A file server is used because mobile devices are not likely to store all media files locally because of the limited memory. How much the file server is used depends on the magnitude of multimedia used and the storage capability of the mobile device. Multimedia elements used often should be cached on the device.

The **runtime server** is responsible for handling requests from the clients and providing the services requested. A more detailed description of this component is given later in this section.

The **location system database** stores all information used by the clients and the server. This database is also used by the creator tool when creating location-aware applications, and by the statistical tool for analysis of user patterns.

The **location server** gets the current positions of the clients through various interfaces to different position technologies like WLAN positioning, GPS, IrDA, Bluetooth etc. A more detailed description can be found in Section 2.4.

From early scalability tests we have found that the wireless network between the clients and the server will be the main bottleneck of the system. If the GSM network is used, only audio streaming is supported. If WLANs like IEEE 802,11b are used, video streaming is supported. The number of simultaneous users to be served depends on how well the physical network is implemented. Another possible bottleneck can be the file server. However, such servers can be duplicated to achieve better performance.

Figure 3 shows the logical view of the runtime server architecture. The system is communicating with client applications through the *GLocServlet* class. Data is exchanged as XML, and the *XMLTransformer* class interprets and transforms the information sent between clients and the server. For the runtime server, we decided to use an architecture based on a centralized control model. This means that all information flow through the *MainController* class. By using centralized control, it is easy to analyse control flows and get the correct responses to the given client requests. It also makes it easy to substitute the servlet class with another class for handling the client communication and to add new interfaces to the system as needed. The *UserManager* class is responsible for maintaining information about the users. This task includes storing the user's last position and deciding whether a person is allowed to communicate with another person (defined as friend). The *PositionManager* class is responsible for returning the user position, adjusted to the type of mobile device used. The *TrackManager* class is responsible for keeping information about the available predefined tracks. Each track has a unique name, so the client application can either request all tracks or one particular track. The *DbManager* class is responsible for all communication with the database.
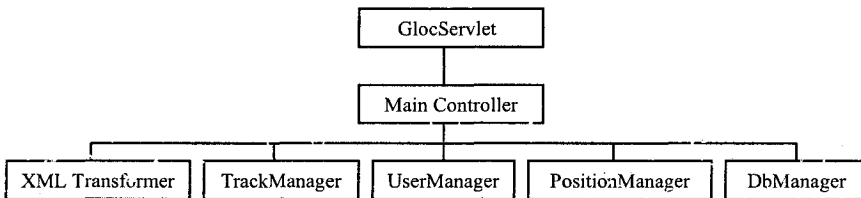

Figure 3. The Logical view of the Runtime system

2.3   The Analysis Phase

The statistical tool is useful for analysing the use of location-aware applications. The tool uses data logged by the runtime system to look at user behaviour and what objects that are most popular. Four classes implement the statistical tool: GUI, LogTool, DbManager and DbLog. The GUI class presents the different services. The LogTool class is responsible for calculation and manipulation of the data stored in the database. The DbManager and DbLog classes handle database issues.

2.4   The Location Server

The location server (see Figure 4), developed by Telenor R&D, is a framework for uniform, geometric-based management of a wide variety of

location sensor technologies. The goal of this framework is to have one server that can get positions of mobile devices through multiple location sensing technologies. By using the location server in the Nidaros framework, we do not need to tailor our system to use a specific positioning technology. We can also use different positioning technologies within the same application.
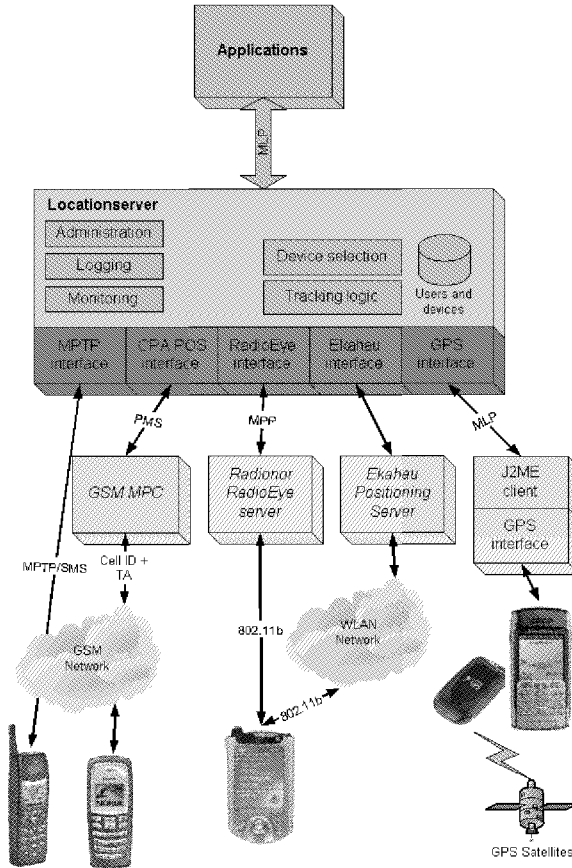

Figure 4. The architecture of the location server

The server includes a middleware protocol specification and a specification of quality-of-service parameters. Further, the server has support for event-driven position reporting (i.e. for change of position) and support for methods for merging and position enhancements. Figure 4 shows an overview of the location server architecture. The architecture is layered and shields the application from the details of collecting and merging location information from a variety of sources. The devices that are positioned by the server are identified by the MAC-address of the device. The location server also manages authorisation for accessing the position of a device. Further,

the architecture provides support for monitoring and tracking of mobile devices. The architecture provides several interfaces to various positioning technologies. The location server currently supports positioning using GPS, GSM, and WLAN.

One advantage from using the location server is that this server handles the complexity of merging locations that might include partly overlaps of positioning systems and seamless transitions between different position systems. This is especially useful for location-aware applications that cover both indoor and outdoor areas.

2.5 The Location System Database

The location system database stores location data, log data, and user data. The *location* data is represented in five tables describing maps, zones, objects, preferences, and mapping. The motivation for these tables is that one location can have several maps covering different territories, each map can cover several zones, each zone can contain several objects, and each object can be connected to several preferences. The preferences of an object are used to provide the dynamic menu service. The user can state the preferences in the attraction, and only objects matching his preferences will be displayed in the client menu. In addition, there must exist a table with mapping information to transform locations from world coordinates to coordinates adjusted to fit the client map.

The *user* data is represented in three tables describing users, user groups and user preferences. The user table contains an MAC-id of the user device and other information. The *user group* table is used to group users that are friends to allow services like tracking friends and messaging. The *user preferences* table stores information about the user's main interests.

The *log* data is represented in two tables. One table is used for storing user movements and one table for storing what kind of objects the user is interested in. The statistical tool uses the log data.

3. IMPLEMENTING A TOUR GUIDE USING THE FRAMEWORK

We created a location-aware tour guide for the Nidaros Cathedral in Trondheim to test the Nidaros framework in a real setting. The cathedral had an existing PDA-based tour guide, called Nidaros Pocket Guide (NPG) that was not location-aware. To show the flexibility of the Nidaros framework,

we decided to implement support for two different types of clients: A PDA and a mobile phone.

3.1   The PDA Client

We decided to develop our location-aware PDA client in Flash MX. This made it possible to reuse code from the NPG and the client template. Our PDA application provided functionality for selecting language, enable trace of own movement, show friends on a map, show zones and objects on a map (with highlighting of current zone and objects), displaying text about objects, and playback of audio or video about an object. The PDA used to run the client was an iPaq with Windows CE and the wireless LAN IEEE 802,11b network integrated. This made it possible to get the position of the device using WLAN-positioning. The location-awareness was presented in the client application in two ways. The first way was to show a map with the position of the user, position of possible friends, highlighting of current zone and nearby objects. The second way was optional for the user, and made the application able to initiate display of objects (text, audio or video) when the user entered the hotspot area of an object. A PDA client is shown to the left in Figure 5.

Figure 5. The system running on iPaq and SonyEricsson P900 and the WLAN tag

3.2   The Mobile Phone Client

A mobile phone client could either be implemented in J2ME or using HTML and the web-browser installed in the phone. We decided to do the latter by implementing an additional servlet component that communicates with the runtime server and produces HTML for the mobile phone. We used WAP push to send the appropriate web pages when the user was within a specific area to enable the client to react on the user location. This made it

possible to, e.g., get the phone to initiate playback of a video when the user was close to a specific alter. We used a SonyEricsson P900 in the test because of its support for WAP push and the built-in multimedia player. To the centre in Figure 5, the mobile phone client is running on a P900. The most common way to position mobile phones is to use GSM positioning. This method is too course-grained to be used for positioning inside a cathedral. To solve this problem, we came up with the idea of letting the user wear WLAN tags that could be positioned using WLAN positioning. WLAN tags are produced by RadioNor Communications, and are small WLAN radios that transmit a MAC-ID. A picture of a WLAN-tag is shown to the right in Figure 5. The size of the tag is about 4cm wide and 3cm high. For the mobile phone client, it was necessary to register the MAC-ID of the WLAN tag and the mobile phone number to link the phone to the tag.

3.3 The Position Technology Used

We used the Cordis RadioEye WLAN positioning produced by RadioNor Communications to position the client devices used in our location-aware application. The RadioEye is a small box with advanced antenna technology and a Linux-server that can determine the physical coordinates of every WLAN-terminal that are active within its coverage area. The sensors decode the MAC addresses of the network devices and determine their geographical position with a typical accuracy of 1-2 meters. A RadioEye covers a radios about 60 degrees from the centre and can e.g. be placed in the ceiling of a building.

3.4 Setting up and Running the Location-aware Application

The demonstration of the location-aware tour guide for the Nidaros Cathedral was performed May 14th 2004. Before we could start to run the demonstration, we had to install the infrastructure needed for running the system. Four RadioEyes were installed at the gallery of the cathedral to cover four different zones of the building. After the RadioEyes were in place, the coordinates from the RadioEye server had to be aligned with the coordinates used by the system. This was done by taking measurements from different locations in the cathedral. These measurements were made in a pre-test before the real demonstration. Approximately 20 people were present at the demonstration representing various technology-oriented companies as well as the media (television, magazines and newspapers). The demonstration of the system was very well received and especially the

mobile phone application attracted much attention. As far as we know, there are no similar location-aware applications running on mobile phones.

4.   EXPERIENCES

This section describes experiences we gained from developing a location-aware application using the Nidaros framework. We have found the framework very useful for several reasons. Firstly, the server side of the system can be used directly without any modifications. The only thing missing is the information to be used in the location-aware application. This information can easily be added by using the creator tool. In addition, the creator tool can be used to make changes to the location information before and during the operation phase.

By using the available client template, the time to implement a client is rather short. Currently, client templates are only available for HTML and Flash. It would have been useful to provide templates for Java 2 Micro Edition and .Net Compact Framework. It does not require much work to write a client from scratch using the defined XML-interface. XML parsing might cause a challenge for a client implemented in J2ME because of the memory limitation in J2ME. However, the XML messages used in the Nidaros framework are relatively small and simple, and do not contain several levels. A possible extension of the Nidaros framework could be to make a client generator to ease the implementation of new clients for all client technologies.

The simulated movement of users was found to be a very useful feature of the runtime system and was invaluable for testing client applications. It takes several hours to set up a real environment with real sensors, and it would be time consuming to engage real users to debug the application.

The database used in the framework was found general enough for the tour guide application. However, there are limitations on what type of information that can be stored. This means that the database scheme might have to be extended to fit any location-aware application.

We introduced a file server that could feed the clients with multimedia contents because the limited storage available on client devices. In an ideal system, all the media files should be stored locally on the device for quick and responsive presentations. This was impossible for the tour guide for the Nidaros Cathedral if more than one language should be supported on the same device. Most of the multimedia files were audio files, but it was not storage space enough for more than one language. By introducing more videos, this would be a bigger problem. We found from the demonstration in the cathedral that the user had to wait from 5 to 7 seconds before the audio

or video was played. We stored the most used media files on the device to avoid such long waiting times. An extension of the Nidaros framework could be to have smarter media file communication management. This means, e.g., that the mobile client can start to pre-fetch files that are likely to be played in the near future based on the location and movement of the user.

The mobile phone client got tremendous attention when we demonstrated the location-aware tour guide in the Nidaros Cathedral. The main reason was to be demonstrated such advanced applications running on devices that many own. The current solution involves a WLAN-tag to make the system work. For a commercial tour guide for mobile phones, the WLAN-tag could be available in the entrance of a sight by paying the entrance fee. Sending an SMS that includes the ID of the tag to the tour guide server could initialize the setup of the mobile location-aware tour guide. The combination of using a WLAN-tag together with a mobile phone gives other new exiting opportunities for location-aware applications that can be used both indoors and outdoors. The WLAN-positioning technologies like the RadioEye can be installed in all public building like airports, hospitals, shopping centres etc.

The use of a location server makes it easy to adapt to new positioning technologies when they are available. The only required change of the system is to implement a new interface for the new position technology in the location server.

The choice of using XML for data exchange between client and server has many benefits. The main benefit is the extensibility of the interface and provision of an open interface to other systems. The main disadvantage of using XML is the overhead sending messages between client and server. For a system with many users, this can cause scalability problems because of the limited bandwidth in wireless networks. A problem we experienced running the location-aware tour guide, was the high demand on CPU and memory on the mobile clients for parsing the XML-data. Generally, the clients spent more time on parsing XML data than they used to send requests and receive responses. A high demand on the CPU will also make the battery run out faster. We foresee that this problem will not be so dominant for future mobile devices with improved performance and battery technology.

5. RELATED WORK

This section describes work on similar frameworks from location-aware and context-aware applications.

The NEXUS Project (Volz and Sester, 2000) has developed a generic infrastructure that can be used to implement all kinds of context-aware applications both for indoor and outdoor services. The NEXUS clients

access the server via a standardized user interface running on the mobile device carried by the user. The interface has to be adjusted to the different kinds of clients, especially concerning the different level of computing power, different amounts of memory and different size of displays. A NEXUS station can manage sensor systems that measure both global indicators (like temperature) and object related information (like location). The NEXUS framework uses separate components for sensor management and communication, and use spatial models with multiple representations to model the physical world stored in distributed databases.

The Framework for Location-Aware ModElling (FLAME) is a configurable, generic software framework for development of location-aware applications (Coulouris et al., 2004). FLAME provides support for multiple sensor technologies, provides a simple spatial model for the representation of locatable entities. In addition, FLAME provides a simple event architecture for the presentation of location information to applications, and a queryable location database. The framework and its applications are largely event-driven in order to accommodate the real-time nature of the location information that they handle. The database holds the initial states (like static regions), and it also holds a synopsis of the real-time location information. A region manager stores and retrieves regions from the database. A spatial relation manager generates application-related events to satisfy currently active subscriptions. The event adapters generate events, e.g., when a person has moved a ``Person Movement Event" is generated.

Dey and Abowd (Dey and Abowd, 2001) presents requirements and a conceptual framework for handling context information. The requirements to be fulfilled by the framework are *Separation of concerns, Context interpretation, Transparent, distributed communications, Constant availability of context acquisition, Context storage and history,* and *Resource discovery.* The conceptual framework for handling context by Dey and Abowd suggests the use of **context widgets** to provide access for applications to context information from their operating environment. The context widget is regarded as a mediator between applications and the operating environment, insulating applications from context acquisition concerns. Context-specific operations are addressed in the framework by four categories of components: interpreters, aggregators, services and discoverers. The framework defined by Dey and Abowd focuses more on the management of various context sources and to represent these context sources in an application.

6. CONCLUSION

Although there exist different types of location-aware applications, there are still many location-aware services that have not been explored. Many of the existing location-aware applications are tailored just for one purpose. In this paper, we have presented the Nidaros framework to be used to implement location-aware applications. Our framework provides support for the development phase, the operation phase and the analysis phase. In the development phase the creator tool is used to add needed information into the database to be used by the final application. Further, the client templates can used for faster development of mobile clients with some basic functionality. In the operation phase, the runtime system manages all interaction between clients and the server including communication of multimedia files and determination of clients' positions using a location server. For the analysis phase a statistic tool can be used to analyse the usage of the location-aware application, detect possible bottlenecks of the system, and see what objects that are most popular.

References

Abowd, G. D., Atkeson, C. G., Hong, J., Long, S., Kooper, R., and Pinkerton, M. (1997). Cyberguide: A Mobile Context-Aware Tour Guide. Wireless Networks, 3(5):421–433.

Cheverst, K., Davies, N., Mitchell, K., and Friday, A. (2000). Experiences of developing and deploying a context-aware tourist guide: the GUIDE project. In Sixth Annual International Conference on Mobile Computing and Networking, pages 20–31, Boston, Massachusetts, United States. ACM Press.

Coulouris, G., Naguib, H., and Samugalingam, K. (2004). Flame: An open framework for location-aware systems. http://wwwlce.eng.cam.ac.uk/qosdream/ Publications/flame.pdf. Submitted for publication.

Dey, A. K. and Abowd, G. D. (2001). A Conceptual Framework and a Toolkit for Supporting the Rapid Prototyping of Context-Aware Applications. Human-Computer Interaction (HCI) Journal. Special Issue: Context-Aware Computing, 16(2–4):97–166.

Garzotto, F., Cinotti, T. S., and Pigozzi, M. (2003). Designing multi-channel web frameworks for cultural tourism applications: the MUSE case study. In *Museums and the Web 2003*, Charlotte, North Carolina, USA.

Satyanarayanan, M. (1996). Fundamental Challenges in Mobile Computing. In Fifteenth Annual ACM Symposium on Principles of Distributed Computing, pages 1–7, Philadelphia, Pennsylvania, United States. ACM Press.

Sørensen, C.-F., Wang, A. I., and Hoftun, Ø. (2003). Experience Paper: Migration of a Web-based System to a Mobile Work Environment. In IASTED International Conference on Applied Informatics (AI'2003), pages 1033–1038, Innsbruck, Austria.

Volz, S. and Sester, M. (2000). Positioning and Data Management Concepts for Location Aware Applications. In 2nd International Symposium on Telegeoprocessing, pages 171–184, Nice-Sophia-Antipolis, France.