MIDDLEWARE SUPPORT FOR CONTEXT-AWARE MULTIMEDIA APPLICATIONS

Hani Naguib, George Coulouris and Scott Mitchell
Laboratory for Communications Engineering, Cambridge University
http://www-lce.eng.cam.ac.uk/qosdream/
{han21,gfc22}@cam.ac.uk, Scott.Mitchell@mail.com

Abstract We describe QoSDREAM, a middleware framework for the construction and management of context-aware multimedia applications. The contributions of QoSDREAM include (1) a novel approach to the handling of location data derived from sensors in the physical world which integrates sensor data from a variety of sources into streams of application-relevant events and (2) a component-based architecture for the construction of real-time multimedia and other context-aware applications. The component architecture supports the construction of application models that are used for quality of service analysis and management purposes. Working distributed applications are derived from the models.

1. INTRODUCTION

Many of the components found in conventional analogue audio and video systems (for example, VCRs, switches, mixers and editing desks) can now be found as software components running on high-performance desktop and mobile computers and using high-speed networks with QoS guarantees instead of point-to-point analogue cables. Other components (cameras, microphones, displays, loudspeakers) are computer peripherals whose operation is managed by software. Digitally stored audio and video streams are already widely exploited in the production and use of programme material.

Another important development is the emergence of *sentient computing* [17] as a valuable adjunct to mobile computing devices and wireless network technologies, enabling applications to behave in a context-aware manner that is, their behaviour takes into account sensor data such as the locations of people and things in order to better support users' tasks. The locations of people and equipment can be tracked using a variety of *location technologies*. These technologies employ infrared, radio, sonic or optical sensing techniques. They provide information about the *proximity* of a locator object to a sensor or they give *coordinates* of the locator object to some degree of precision. Some also provide information about the orientation of the locator object. Active Badge [15], Active Bat [16], and tag technologies [25] are representative examples of location technologies that depend upon the detection of small dedicated tags or badges. GPS and mobile phone technologies and handheld computers plus beacons can be exploited as location technologies when the handset is able to communicate with the relevant systems.

As these developments proceed, there is an increasing need for flexible application development and management environments, enabling context-aware software for the manipulation and management of multimedia streams to be developed and deployed quickly and cheaply. There is a profusion of potential application domains for such systems. They include factory and plant management, security surveillance, support for clinical workers, airport and seaport control and major incident coordination. Applications are likely to emerge wherever there is a need for multi-party communication employing video or audio links between users, many of whom are engaged in multiple tasks and migrate according to the needs of their work, within a workplace or more widely.

In Section 2 we define and discuss the goals of the QoSDREAM framework. Section 3 reviews some related work. Section 4 describes the architecture of the framework, emphasising the features for the management of location data and those that support extensibility, configurability and scalability. Section 5 reports briefly on our initial experience in using the framework to build a context-aware application. Section 6 concludes the paper.

2. DESIGN APPROACH

We have defined the following goals for the QoSDREAM framework:

1 It should be simple for applications to construct multimedia flows in terms of a uniform set of high-level components (sources, sinks, transforming elements and communication channels). The framework should mask differences in terminal equipment and communication links and enable applications to add and delete multimedia streams and modify their configurations in response to changes in resource availability, application needs and users' contexts.

2 The framework should maintain application-defined constraints on the integrity and quality of service properties of multimedia flows during configuration changes.

3 Location information should be made available to applications with reference to single spatial model of the physical world, regardless of the sensor system that was used to generate it.

4 The multitude of raw real-time events that are reported by context sensors should be filtered as required and presented to application components in a uniform format.

5 A set of persistent data describing the current set of running applications and a synopsis of the real-world model should be available, so that applications or components can be added or brought up to date in a manner that is consistent with the existing state.

Space does not allow us to describe fully our approach to all of these goals. Therefore in this paper we focus on goals 3,4 and 5. Please refer to [20,21],for details regarding our approach to goals 1 and 2.

3. RELATED WORK

We are not aware of any integrated middleware platform that aims to achieve all of the goals described above. In this section we review several projects that address major portions of the requirements space we have identified.

The SPIRIT project [16] at AT & T Labs, Cambridge, has built a substantial Corba-based application framework to manage and exploit location data from the Bat high-resolution location technology. The SPIRIT system holds real-world descriptions in an Oracle database to provide persistence and query ability, with a cache of Corba objects holding more up-to-date data about currently-active real-world entities. High-precision geometric modelling is used to support queries and invocations that request the locations of objects, whereas QoS-DREAM aims to limit the computational load on application hosts by building a spatial model in terms of a (potentially large) set of possibly-overlapping 2.5-D regions and to support a wider variety of location technologies.

The Glagow Context Server [18] is designed as a low-cost context-aware architecture for mobile applications. It exploits PDAs connected by a wireless network and receiving data from static infrared beacons that emit identity, location or other information. The bandwidth of the handheld devices and the network technology cannot support multimedia streams, but like QoS DREAM, it aims to support clinical applications to provide information on the locations of patients, staff and equipment and to enhance communication between clinical staff. Such an architecture can substantially reduce the cost and time required to deploy context-aware applications.

The Cooltown project [5,19] is an architecture for the support of applications that span the physical and virtual worlds. It exploits the world wide web to hold the data modelling real-world objects. It does not provide any special support for multimedia streams and it is not clear to us how Cooltown addresses the problems of system loads in large-scale applications.

Other component-base multimedia middleware systems include Sumo-ORB [8] and CINEMA [2]. CINEMA also uses a model-based approach to reconfiguration; however QoS DREAM is novel in its use of an active runtime model encapsulating QoS and event management, as well as reconfiguration. In addition, the atomic action mechanism encourages the construction of reliable, yet highly dynamic, applications. Researchers at Lancaster University have been investigating mobile multimedia support for emergency services [10], such as equipping ambulances with videoconferencing systems.

4. ARCHITECTURE

Figure 1 shows the major constituents of the QoSDREAM architecture. It consists of four major parts. At the lowest level we have the **Location Service** It is in charge of gathering location information from specific location-technologies and presenting it to the rest of the system in a location-technology independent fashion. The Location Service's functionalities also include managing and filtering the potentially large amount of location information that can be generated. This information is made available to the rest of the system and applications through an **Event Messaging System.** The Event Messaging System distributes events to interested parties and provides a number of filtering services.

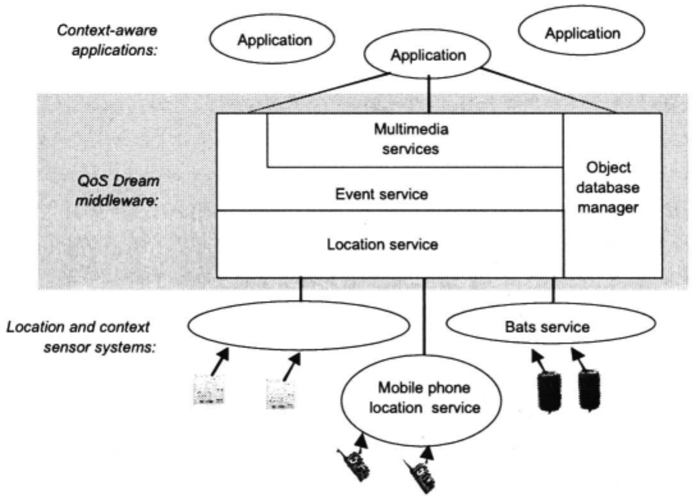

Figure 1. Overview of QoSDREAM's Architecture

Applications are exposed to three major APIs:

- **The Event Messaging Service,** which allows applications to subscribe to receive specific events (including those generated by the location service) and also to send their own application specific events.

- **A Distributed Multimedia Service,** which allows the application to construct and manage, distributed multimedia components. Multimedia applications are built as a collection of interconnected components, such as cameras and displays.

- **A Distributed Object Database Management Service,** which provides a means of sharing system wide properties such as information about known physical objects. The information held by the database is largely 'static' (i.e. it is not modified very often), although some location-specific data may also be stored with it.

4.1. Location Information

A wide range of location sensing technologies already exist and more are under development [25]. Some, such as the Active Badge system [15], provide information about the presence of a user in a region or a room. Other location technologies, such as the Active Bat system [16] are much more precise, providing information that is accurate to a few centimetres. All location technologies generate sensor events in response to changes in the locations of locatable objects. The rate at which they are generated is high, especially for high precision location technologies. Only a fraction of the sensor events are of interest to any specific application.

Our Location Service provides location-dependent information. The Location Service is a collection of objects that retrieve location information from existing location sensors and make this information available to QoSDREAM applications. The type of location information available is highly dependent on the location-technologies being used. QoSDREAM's Location Service interprets this information and presents it in a location-technology independent format. This is achieved by representing location information in terms of regions and the interactions between regions.

4.1.1 Modelling Location Information. Within QoSDREAM, applications are presented with a simplified model of the real physical world. This model contains the following abstractions:

- **Locators,** which represent objects whose location can be determined by a given location-technology. Active badges and Bats are examples of Locators.

- **Locatables,** these represent objects whose locations need to be tracked. Examples include people and equipment. Locatables must have at least one associated locator, so that their location can be inferred from its locator.

- **Locatable Regions,** Each locatable will in general have one or more regions. These regions define various location specific characteristics of Locatables. For instance in the case of a person, they may include in addition to the person's location, their visual and audible fields.

The main reason for using regions as a way of presenting location information is the varying degree of precision of location technologies. Active Badges will only place a badge within a room, Active Bats provide more fine grained information. Expressing location information as regions allows the incorporation of a wide range of location technologies. Furthermore the use of regions also aids in the management of location information particularly with regards to filtering this information in order to reduce location-related traffic. This is discussed in the following subsection.

Location related information is presented to applications as interactions between the various regions in the model. In particular the change in overlap status between regions (For example a person walks into a room, his 'location region' overlaps with the room's region).

4.1.2 Management of Location Information. Our Location Service performs management of location information. Figure 2 depicts the various abstractions that form this service.

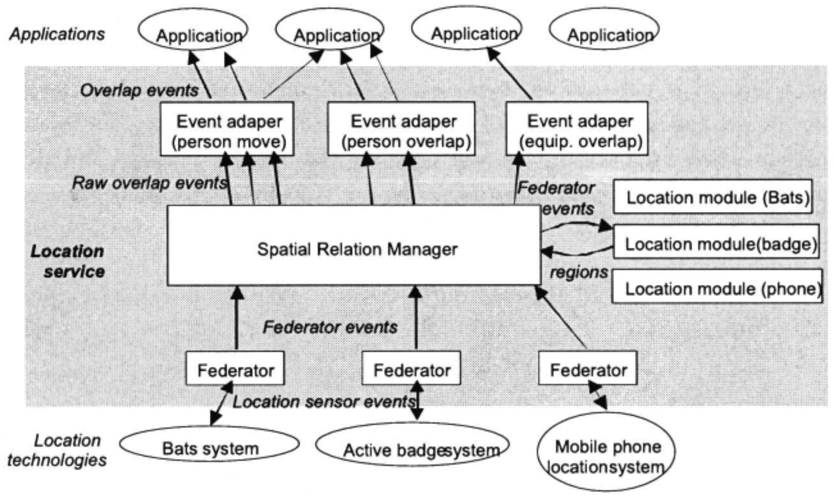

Figure 2. Location Service Architecture

At the lowest level we have **Federator** objects, these interface with specific location technologies and can detect location information. In our lab we have an 'Active Badge Federator' which is capable of interpreting the data being generated by the Active Badge System. The information gathered by Federators is exported through light-weight events called **Federator Events.** These contain generic information such as the type and id of the federator as well as federator-specific data.

These events can be monitored by applications that require very detailed location information. In general applications are only interested in specific location-related information and so can use other types of events (described below) to receive only events that it is interested in. Federator events are also sent to our **Spatial Relations Manager** (SRM). The SRM's task is to convert information gathered by Federators into overlapping events between the regions in our world model. Before the SRM can reason about the regions and their interactions, **Location Modules** translate Federator events into regions. For example a Location Module for an Active Badge System, would generate a location region for the given locator in the Badge Federator Event. The shape of this region would be the same as that of the room where the locator was detected. This can be derived from the sensorID field in the Federator Event.

Location Modules are Federator-type-specific, and can be loaded from the database by the SRM. The SRM uses the federatorType field of Federator Events to determine the required Location Module.

Once the SRM has obtained updated regions from Location Modules it analyses them looking for changes in overlaps between regions. If those are found then the SRM produces what we call **Raw Overlap Events.** These contain the type of overlap between two regions as well as their previous relationship.

Applications can register to receive these events, but again these events are too general for most applications and cannot be filtered. Instead application register to receive specialised overlap events produced by **EventAdaptors.** These types of events can be filtered and relate to higher level abstractions, such as people and equipment.

EventAdaptors are objects that receive Raw Overlap Events generated by the SRM and can filter and transform these into events that are of interest to applications. The events generated by EventAdaptors are called **OverlapEvents,** and are transmitted by QoSDREAM's Message Service. New EventAdaptors can be easily created and dynamically incorporated into QoSDREAM. By default we provide three types of EventAdaptors:

- **PersonMovement Adaptor:** This generates overlap events when a person's location region overlaps with a geographical location. This event contains a personID and a LocationID fields.

- **PersonOverlap Adapter:** This generates overlap events when a person's location region overlaps with that of another. This event contains a person1ID and person2ID fields.

- **EquipementOverlap Adapter:** This generates overlap events when a person's location region overlaps with that of an equipment. This event contains a personID as well as a equipmentID fields.

Applications are free to choose what events they wish to receive. They can further filter OverlapEvents on the value of the fields in the event. For example, an application interested in the arrival at work of person George can register to receive PersonMovement events whose personID field equals George's id.

To aid with the scalability the representation of the physical world is divided into zones. These zones can be used by the SRM to cut down on the number of regions it must analyse when looking for overlap events. Zoning also allows for a federation of SRMs to be set-up, each of which is responsible for specific zones (as long as those don't overlap). For instance floors in a building might each be handled by a separate SRM.

4.2. Event Messaging Service

The multiplicity of raw events occurring in context-aware systems determines the need for an event management system. Even after the processing of location information in the manner described above, the potential network and processing loads are substantial. A mechanism is required that will enable applications to see those events that are relevant to its responsibilities while filtering out those that are not. The event mechanism should be distributed to avoid the problem of hot-spots at network nodes where event traffic is concentrated.

The QoSDREAM framework provides an event abstraction that enables applications to register for events by event type. Further selection is performed by event filters which the application can instantiate. Options for event queuing, communication, fault tolerance, event persistence, reliability and security will be offered in a future version of the DREAM framework. The event handling abstraction outlined here is compatible with most general-purpose event handling systems including Corba Events [11], Elvin [1,13] and Herald [3,24].

QoSDREAM uses events primarily for delivery of contextual information to applications. The messaging service allows application objects to send and receive events and is also used by the Location Service to deliver events generated by EventAdaptors.

The Messaging Service facilitates the management of location information in a number of ways. It allows clients to request the delivery of events without having to know anything about the sources of these events. Similarly event sources do not require any knowledge about their clients. The messaging service also allows applications to specify filters on the events that are sent to it. This coupled with the location service greatly reduces location-related traffic. Without the location service and messaging service, applications would need to monitor all location information generated by the various underlying location technologies.

The Messaging Service is itself independent of the event delivery system used to transfer events from event sources to event clients. It provides the following abstractions:

- **Event Sources:** Are objects that produce events.
- **Event Clients:** Are objects that register an interest in receiving events.
- **EventDescriptors:** These are descriptions of the events generated by event sources. This includes information about the fields of an event as well as the event type name.
- **Message:** These are instances of events, and contain values for the fields in an event.

EventDescriptors are stored in the database by event sources and remain there while the source is registered to produce such events. Having EventDescriptors stored in the database in this fashion allows clients to easily discover the types of events currently available in the system. The database in this situation acts as an event broker.

Clients obtain a reference to an EventDescriptor from the database and through it can register their interest in receiving these types of events. Event-Descriptors also allow clients to further filter the events they receive through a set of methods that specify specific value for fields in the event.

Our current implementation of the Messaging Service uses an implementation of the Cambridge Event Architeture called Herald [24] as the event delivery system. Herald contains similar abstractions to those listed above. Other event delivery systems can be easily added to our Messaging Service by the implementation of a single interface. Event sources can specify which underlying event delivery system is to be used if they so require.

4.3. Database Management System

While an application is running, its components in the physical world (cameras, screens, people, rooms) and the virtual world (conferences, phone connections, patient records) are modelled as objects in a distributed program and their attributes are manipulable by the application program. But the attributes of these components are also needed at application start time and whenever a new component is added to an application. For example, a ‘locate a specialist’ application might be invoked to find a cardiology specialist when a heart patient arrives in the emergency department of a hospital. The persistent data includes a synopsis (which is not necessarily up-to-the-second) of the locations of all of the doctors, nurses and patients in the hospital. It also includes details of doctors' specialisms. The persistent data should be queryable in order to resolve requests such as the example just mentioned.

The Database Management System (DMS) plays a central role within the QoSDREAM architecture. It is used to store static information about the system such as geographical information and information related to people and equipment. It is also used as an event broker, storing information that allows event sources and clients to interact, and contains a synopsis of the location-related information.

QoSDREAM exposes the DMS through an interface called the **QueryManager.** This interface allows for the objects stored within the DMS to be manipulated. Apart from providing a means to access the database the QueryManager also decouples the platform from the underlying database technology used (With the restrictions that it must support objects and be capable of handling OQL queries).

Our current implementation uses Castor from Exolab [9]. Castor is being used as an object-to-relational database mapping tool, and supports any relational database that is JDBC compliant and has full transactional support. (We currently use MySQL[14]).

The QueryManager interface still needs further refinement, for example it lacks direct support for transactions (which currently is only possible by obtaining a reference to the underlying castor Database object). Furthermore the database is a single point of failure and may become a bottleneck. We are currently investigating these issues.

5. EXPERIENCES

This section describes some of our preliminary experiences in building applications using the QoSDREAM architecture. The application described in this section is called **ActiveLab,** and is about to be deployed at our laboratory as a basis for our evaluation. Members of the laboratory will use it as a communications aid, allowing them to locate one another, establish and participate in audio/visual conferences. It exploits location information provided by the Active Badge System (through the QoSDREAM architecture), that is installed in our laboratory. Features of ActiveLab include:

- The ability to locate/track people, rooms and equipment.
- Reroute conferences and desktops as users move around the laboratory.
- Set alarms providing warnings for when given people enter or exit specific regions of the laboratory. (Useful to PhD students)
- Allow users to define default actions depending on the context in which they find themselves. For example to disable incoming phone or video calls when in a meeting.

The following sub-sections describe some of the experiences we have found while developing specific areas of the ActiveLab application.

5.1. Application Modelling

The ActiveLab application's model of the physical world contains a number of abstractions. These include people, computers, devices, rooms and regions as well as representations for the Active Badge System (badges and sensors). Creating representational objects for these abstractions was very simple, although our use of Castor requires that a mapping file be produced in XML. This file specifies the classes that are to be made persistent-capable and the names and types of fields that need to be kept persistent.

Additionally once the database objects have been designed, the database needs to be populated. We found this to be a slightly tedious task since measurements of our laboratory's rooms were required. Furthermore badge ids and people and equipment information were also gathered, although in this case we were able to import much of this information from pre-existing software.

We found that the use of OQL queries to interface with the database greatly simplified the retrieval of database objects.

5.2. Multimedia Modelling

Our approach to modelling the conferencing tools within the ActiveLab application was to build a composite component which we named **Conference.** A single instance of Conference is created for each of the ongoing conferences. This composite component contains a set of **Participant** components as well as an interface for adding/removing and connecting conference participants. A Participant is also a composite component in charge of controlling the multimedia components needed by a single conference participant. These include a camera, microphone, speaker and displays.

We have found a number of benefits from using our multimedia framework. Its component-based approach to application construction promotes reuse (In fact the Conference component can be used by any application which requires a conferencing ability with no modifications). Most of the atomic components used in the model were taken from pre-existing applications. This also meant that any preoccupations we might have had from platform-specific coding for multimedia devices were removed.

Unfortunately it is still too early to perform a thorough performance evaluation of QoSDREAM. Table 1 gives an indication of the typical location-specific traffic found in our lab. This traffic was measured over a five minute interval and at the time there were six members of the lab wearing badges.

The ActiveLab application registers interest only in PersonMovement events (in order to update its display). Thus in this application, the QoSDREAM

Table 1. Typical location related traffic

Badge Events (Federator Events)	671
	These are the events generated by the ActiveBadge Federator. They are generated whenever a badge is picked up by a sensor. Many of the rooms in the lab have more than one sensor and therefore a badge may be picked up by more than one sensor. Most of these events are as a result of a person's badge being detected, since equipment badges emit their signal less often, than badges worn by people.
Raw Overlap Events	204
	These are the events generated by the SRM.
PersonMovement Events	12
	These events are generated by the PersonMovement Adaptor in response to people moving in and out of rooms.
PersonOverlap Events	40
	These are generated by the PersonOverlap Adaptor. In this example those were caused by people moving into rooms where other people were present.
EquipmentOverlap Events	48
	These are generated by the EquipmentOverlap Adaptor. Those were caused by people moving into different rooms and their region overlapping with those of the equipment in the new room.

location architecture reduces the event traffic that the application process is required to handle by a factor of approximately 17.

6. CONCLUSION

We have described those parts of the architecture and design of QoSDREAM that are concerned with location information management, a general-purpose framework to support the construction of context-aware multimedia applications. The contributions of the work include:

- a novel approach to the handling of location data derived from a variety of location-sensing technologies, integrating and representing them in terms of overlap events that are constructed by geometric analysis, thus reducing the incoming event traffic to proportions that are manageable by applications by its aggregation into relevant relations.

- an implemented framework that integrates an application-level service to handle a variety of types of location data with a service to configure and manage real-time multimedia streams.

The architecture encourages the use of pluggable components. Thus the components of the framework that support event-based communication and persistence are hidden behind generic interfaces that provide the abstractions required for the class of applications that the framework aims to support.. This makes the platform largely independent of the specific protocols and mechanisms used in providing its services.

Work is planned on the extension of the framework to support large-scale application domains, including the intelligent hospital. This will call for the incorporation of fault tolerance and security mechanisms and for refinements to the platform following an evaluation of its performance under large-scale application loads. The extension of the federator/relation approach to other types of data source (e.g. biometric data) – is seen as a promising direction for further research.

Work currently in hand is expected to result in the early availability of a public release of the QoSDREAM framework.

Acknowledgments

The following people have contributed to the development of the QoS DREAM framework: John Bates, Mark Spiteri, Martin Harper, Sarah Gunn, Eli Katsiri, Kasim Rehman. We thank the following for valuable advice and support: Andy Hopper, Mark Wharton, Pete Steggles, Tim Kindberg. Last but not least, we are grateful to Dr Tim Coats and the staff at the A& E Department of the Royal London Hospital for giving their time to our study of their requirements. This research is funded by EPSRC, the UK Engineering and Physical Sciences Research Council.

References

[1] David Arnold, Bill Segall, Julian Boot, Andy Bond, Melfyn Lloyd, Simon Kaplan, "Discourse with Disposable Computers: How and why you will talk to your tomatoes", Usenix Workshop on Embedded Systems (ES99), Cambrdige Mass, March 1999. http://www.usenix.org/publications/library/proceedings/es99/full_papers/arnold/arnold_html/

[2] I. Barth, "Configuring Distributed Multimedia Applications Using CINEMA", Proc. IEEE MMSD'96, Berlin, Germany, Mar 1996

[3] J. Bates, J. Bacon, K. Moody and M.D. Spiteri, "Using Events for the Scalable Federation of Heterogeneous Components", Proc. ACM SIGOPS EW'98, Sintra, Portugal, Sep 1998

[4] F. Bennett, D. Clarke, J. B. Evans, A. Hopper, A. Jones and D. Leask, "Piconet - Embedded Mobile Networking", IEEE Personal Communications 4:5, Oct 1997

[5] John Barton and Tim Kindberg , The challenges and opportunities of integrating the physical world and networked systems, paper submitted to Mobicom 2001, http://www.champignon.net/TimKindberg/

[6] Bluetooth Special Interest Group, "Bluetooth Specification v1.0B", Dec 1999. http://www.bluetooth.com/developer/specification/

[7] J. Bacon, K. Moody, J. Bates, C. Ma, A. McNeil, O. Seidel and M.D. Spiteri, "Generic Support for Asynchronous, Secure Distributed Applications", IEEE Computing, Mar 2000

[8] Gordon Blair and Jean-Bernard Stefani, Open Distributed Processing and Multimedia, Addison-Wesley, Harlow, England, 1998.

[9] The Castor Project. 2000. http://castor.exolab.org/

[10] G. Cugola, E. DiNitto, and A. Fugetta, "Exploiting an event-based infrastructure to develop complex distributed systems", Proc. ICSE'98, pages 261-270, 1998.

[11] G. Coulouris, J. Dollimore and T. Kinberg, Distributed Systems: Concepts and Design, Edition 3, Addison-Wesley 2001.

[12] E. Coiera, Clinical Communication & shy; A New Informatics Paradigm, Technical Report HPL-96-64, Hewlett-Packard 1996.

[13] G. Fitzpatrick, T. Mansfield, S. Kaplan, D. Arnold, T. Phelps, and B. Segall, "Instrumenting and augmenting the workaday world with a generic notification service called Elvin", Proc. ECSCW'99, Copenhagen, Denmark, Sep 1999

[14] P. Gulutzan, T. Pelzer, "SQL-99 Complete". CMP Books, April 1999

[15] A. Harter and A. Hopper, A Distributed Location System for the Active Office. IEEE Network, Vol. 8, No. 1, January 1994.

[16] A. Harter, A. Hopper, P. Steggles, A. Ward and P.Webster, The Anatomy of a Context-Aware Application. In Proceedings of the 5th Annual ACM/IEEE International Conference on Mobile Computing and Networking (Mobicom '99), Seattle, Washington, USA, August 15-20 1999 1999. ftp://ftp.uk.research.att.com/pub/docs/att/tr.1999.7.pdf

[17] Andy Hopper, Sentient Computing, The Royal Society Clifford Paterson Lecture, 1999, http://www.uk.research.att.com/abstracts.html# 108

[18] Chris Johnson and Kevin Cheng, The Glasgow Context Server: a Wireless System for Location Awareness in Mobile Computing. Submitted to IHM/HCI 2001, http://www.dcs.gla.ac.uk/~johnson/papers/context_aware/

[19] T. Kindberg, J. Barton, J. Morgan, G. Becker, D. Caswell, P. Debaty, G. Gopal, M. Frid, V. Krishnan, H. Morris, J. Schettino, B. Serra, andM. Spasojevic. "People, Places, Things: Web Presence for the Real World". Proc. 3 rd Annual Wireless and Mobile Computer Systems and Applications, Monterey CA, USA, Dec. 2000. p 19.

[20] R.S. Mitchell, Dynamic Configuration of Distributed Multimedia Components. Ph.D. Thesis, University of London, August 2000. http://www-lce.eng.cam.ac.uk/qosdream/publications/

[21] Scott Mitchell, Hani Naguib, George Coulouris and Tim Kindberg, A QoS Support Framework for Dynamically Reconfigurable Multimedia Applications. In Lea Kutvonen, Hartmut König and Martti Tienari (eds), Distributed Applications and Interoperable Systems II, pp 17-30. Kluwer Academic Publishers, Boston, 1999. Also in Proc. DAIS 99. http://www-lce.eng.cam.ac.uk/QoSDREAM/publications/

[22] Scott Mitchell, Mark D. Spiteri, John Bates and George Coulouris, "Context-Aware Multimedia Computing in the Intelligent Hospital", In Proc. SIGOPS EW2000, the Ninth ACM SIGOPS European Workshop, Kolding, Denmark, September 2000. http://www-lce.eng.cam.ac.uk/QoSDREAM/publications/

[23] QoS DREAM Project, "QoS DREAM Hospital User Study", November 2000. http://www-lce.eng.cam.ac.uk/QoSDREAM/applications/hospitalstudy.php

[24] M.D. Spiteri, An Architecture for the Notification, Storage and Retrieval of Events, Ph.D. Thesis, University of Cambridge, Jan 2000. http://www-lce.eng.cam.ac.uk/QoSDREAM/publications/

[25] R. Want, D. M. Russell. "Ubiquitous Electronic Tagging". Distributed Systems Online, IEEE 2000. http://www.computer.org/dsonline/articles/ds2wan.htm.