Merino: Towards an intelligent environment architecture for multi-granularity context description

Bob Kummerfeld [a], Aaron Quigley [b], Chris Johnson [c], Rene Hexel [d]

[a] *Smart Internet Technology Research Group, School of Information Technologies, University of Sydney, Australia*

[b] *Smart Internet Technology Research Group, School of Information Technologies, University of Sydney, Australia*

[c] *Department of Computer Science, Australian National University, Canberra, Australia*

[d] *School of Computing and Information Technology, Griffith University, Nathan Campus, Australia*

Abstract

The Intelligent Environment consists of ubiquitous connectivity in sensor rich environments coupled with adaptive technologies that allow access from multiple devices with varying capabilities. Context information is drawn from what the Intelligent Environment can sense about its current physical and computational environments. This context information relates to aspects of the environment which include, who is in the environment, what they are doing, what they have done, and what their actions are. Multi-granularity context description provides a mechanism to describe, at increasingly more abstract levels, such context information gleaned from the Intelligent Environment. The proposed architecture for the Intelligent Environment is central to determining methods for capturing such information at the lowest level, interpreting by historical reference, and aggregating into increasingly more abstract forms.

Key words: Context; Personalisation ; Smart-Sensors ; Personas ; Smart Environment Agents.

1 Introduction

Ubiquitous computing requires computers to have more knowledge of the real world, the environment surrounding users, than computing did previously. The ubiquitous computing environment therefore has to handle large numbers of sensors, and to hide the physical sensors and communications under layers of abstraction. The sensor devices range in complexity, from devices as simple as a binary on-off, up to sensors that can decide "a meeting is now happening in this room". Their communications abilities range from the simple on-off binary signal to supporting Web services.

The raw data produced by the sensors has an enormous range of speed and volume – it may be as simple as a one bit value on a frequency of hours or days, up to a continuous stream of video data. To enable higher-level inferencing to be performed with sensor information, that information must be consolidated, in many cases, and must be distributed to the computing devices and programs that wish to deal with it.

Adaptive sensors and integrating sensors are therefore an essential part of bridging between the raw devices and an intelligent ubiquitous computing environment that includes user modelling[Kautz et al., 2003].

The issues include both the representation of data coming from raw sensors in a form suited to programs that are capable of reasoning with it, including sensors that have a low level of intelligence but deal directly with other sensors' information; and the efficiency, security and privacy related to transmitting the information from sensors and user models over any kind of network[Schreck, 1997,Busboom, 2002].

The scale of the ubiquitous computing system is significant to the design of both the distribution of sensor information and the way in which sensors and information are represented. To cater for even one enterprise of the scale such as a manufacturing plant, a shopping mall, or a university, it must be scalable to the extent

Email addresses: bob@it.usyd.edu.au (Bob Kummerfeld), aquigley@it.usyd.edu.au (Aaron Quigley), Chris.Johnson@cs.anu.edu.au (Chris Johnson), rh@cit.gu.edu.au (Rene Hexel).

of allowing tens or hundreds of raw sensors, in hundreds of rooms in a building, with hundreds of buildings to a campus [Schmidt and Beigl, 1998].

1.1 Layers of context

Sensors are assumed to be connected to the net and capable of communication. The sensors that connect directly to the environment (such as movement detectors, proximity detectors, light level meters, thermometers) typically produce raw data, at high frequency or at irregular intervals. The data comes as primitive bit streams or simple values. As the first stage of making sense of this data, each sensor's information must be capable of being

- integrated against its own history
- adapted from one representation to another
- combined with other sensors into richer forms of sensor information (where conflicts between the different sensors can be resolved).

For example, one of the standard motivating cases is that of detecting that a meeting is in progress in a particular room, allowing context-aware applications to modify their behavior accordingly. Such a sensor is clearly not a simple one: we will refer to it as a *smart* sensor. To approximate the human-based metrics that are used to make a decision like this, such as "number of people present" and "noise level", we might consider a location-based face-detecting and -counting sensor (a sensor that is based on a camera, but not transmitting images onto the network), combined with a noise-level sensor – in the same physical vicinity. The combination of values of these sensors coming above some preset threshold, within certain time periods, may trigger a smart sensor to indicate the information "meeting in progress". A computer application might use this sensor, but further confirm this indication for its own purposes, by inference using the daily calendar or user model of one or more of the participants, or the room-booking system of the owning organisation. Such a sensor might be used by the participants' own laptop or PDA computer applications to modify its way of handling of incoming messages, or by the room-booking system itself to confirm that the booking has in fact been taken up, or to control the lighting and heating levels and the access to other equipment of that room.

It is clearly a powerfully simplifying design decision to view sensors of both simple and complex kinds in a common framework, from the point of view of the computational elements of the ubiquitous computing system. Organising the whole system into layers of complexity is also necessary, given the scale of the number and variety of sensors involved.

A further requirement of a system on this scale is that the network must be capable of efficiently distributing the sensors' information and user model information beyond the immediate vicinity, but at the same time, the data and information flowing on the communication network must be made secure for both personal privacy and safety reasons.

1.2 Security and privacy

Even raw sensor information must be made secure as soon as it goes over any kind of network beyond the very local. Where sensors are reacting to the proximity of people and reporting their identity and their precise location, there are issues of privacy even in the relatively secure working campus (who needs to know that I am consulting a medical doctor in my lunch break?) and of personal security, where there is a possibility of stalking. Even the very lowest layers of context handling must provide security against inferencing that might reveal the location of a person at risk, or violate privacy policies by revealing the details of office workers' movements in fine detail within a suite of offices and its service areas.

Encryption is only part of the security solution: locations and personal identities may need to be obfuscated by depersonalisation and location fuzzing, incorporated at the level of simple sensors.

Pragmatic issues of implementation therefore lead us to have at least two layers. The complexity of handing data from raw sensors and making simple inferences from this data lead us to a further layering of the handling of sensor information.

2 Background

Early works in ubiquitous computing already touched the complexities of context gathering, processing, and representation as well as privacy and security issues [Weiser, 1991]. Despite this insight, most approaches were based on case studies and projects. In light of the complexity that arises from having multiple, distributed layers between the sensor and system levels at the bottom and the application layer at the top, most projects and case studies were aimed at a very narrow, vertical path through these layers. As a consequence, the techniques chosen in most approaches matched the particular problem addressed by the study and not so much the general case.

2.1 Major Case Studies and Projects

The Aware Home project [Kidd et al., 1999] is a conglomeration of projects aimed at building a context aware, intelligent home. The goal of the project is to create a home environment that is aware of its occupants' whereabouts and activities.

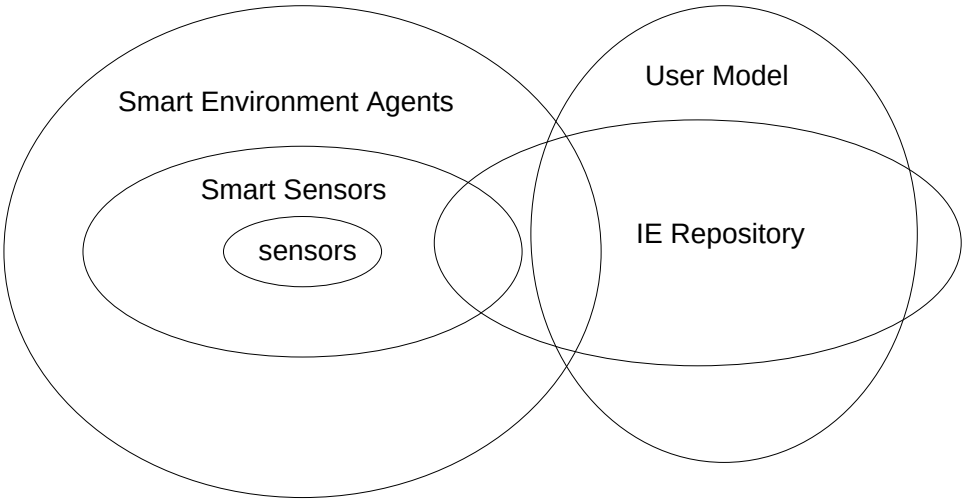

Fig. 1. Section of intelligent environment service layer description

The Massachusetts Institute of Technology houses two different context aware computing projects. Project Oxygen [Brown, 2001] is an attempt to interact with computing environments on human terms. The core emphasis of this project is to use speech and vision for user interaction. The Intelligent Room [Phillips, 1999] is an AI centred multi agent approach to embed computers into ordinary office or home environments.

Typically, these projects provide toolkits [Salber et al., 1999] or APIs that allow some form of context information to be utilised by applications. Other projects such as Cooltown [Kindberg et al., 2000] attempt to implement a loose framework that is based on the notion of a web presence for people, places, and things. It uses the existing internet and world wide web infrastructure as a glue for communication between these entities.

2.2 Context Infrastructures and Environments

An approach that uses an existing network infrastructure is the Ambient Interaction Framework [Fitzpatrick, 1998], which uses an I/O oriented framework to connect different sensors and actuators. The framework relies on a content based message routing system called Elvin [Fitzpatrick et al., 1999].

Based on their experience with the Aware Home project, Abowd et *al* presented an architecture for context aware applications that uses a widget abstraction model to represent sensors and actuators [Salber et al., 1999]. A similar architecture was presented by Grimm et *al* that uses a task/tuple abstraction model to handle context information [Grimm et al., 2000].

2.3 Context Distribution and Scalability

While the architectures presented above are suited to the well defined and constrained environments they were designed for, they lack flexibility in open, distributed environments. A first attempt to address part of this issue was presented by Kantor and Redmiles [Kantor, 2001]. Their approach is to introduce a group based distributed subscription service that allows applications to locate and subscribe to context information provided by other applications at various abstraction levels.

Despite multiple attempts to solve parts of this problem, at the moment, the biggest challenge remains scalability in large scale, dynamically changing, distributed environments. The system architecture introduced in this paper attempts to address this problem at the very foundation of sensor and actuator interaction with intelligent, user centred applications.

3 Merino Architecture: Layers of context

The proposed architecture for context layers for the sensed Intelligent Environment is shown in Figure 1. This architecture consists of five distinct elements, sensors, smart sensors, smart environment agents, the Intelligent Environment repository, and finally the user model.

At the lowest layer, the sensors are mechanisms in both hardware and software to interrogate both the physical and computational environment. These sensors include, hardware to measure the physical environment such as temperature, motion, light, vibration, and pressure, along with software sensors which detect user login, network activity, and application activity. Sensors are assumed to be net connected devices, where connectivity ranges from conventional fixed line to wireless sensor networks. Existing work in ubiquitous computing has often focussed on providing such voluminous data as *context* which an application engineer must then incorporate and interpret in an ad-hoc application specific

manner [Hawick and James, 2003,Salber et al., 1999]. In contrast, here the lowest layer of abstraction an application engineer can use is the *smart sensor layer*.

3.1 Smart sensor layer

Smart sensors, provide the first layer of context abstraction in our intelligent environment architecture, where the repository also acts as the secure and scalable distribution mechanism for context information. Rather than provide the sensor data at the lowest level of detail (raw data) to an application programmer, here the smart sensor layer is provided. The smart sensor layer includes methods for aggregating, filtering, and eliding the raw sensor data into forms which are made available through the repository interface. For example, a location context agent would harvest data, possibly contradictory, directly from a range of physical and software sensors. This data is used to determine a person's physical whereabouts, as being present in a identified location. This "location" information is then deposited into the IE repository by the context agent.

3.2 Smart environment agents

Smart environment agents provide the second layer of context abstraction. These agents form into two classes, *rich-context* providers and *performance enhancers*.

Rich-context providers, access a range of information from the IE repository to form higher level context information or "rich-context". For example, a meeting smart environment agent may access location context information, calendar context information, and noise context information to form a high level piece of rich-context information, that a meeting is taking place. In general such agents build such high level abstraction by the use of first level smart sensor information, user model information, and historical smart sensor and user model information.

Performance enhancers, are the second class of smart environment agents which act with environmental knowledge over the smart sensor layer improving their individual and collective performance. Such agents interact with both the smart sensor layer and the computing environment to determine methods of improving the performance of the global smart sensor layer. One approach is for these agents to include learning and reasoning algorithms to discover patterns which enable the agents to improve the performance (eg. maximize battery life) and scalability (eg. minimize communications) of the underlying smart sensor layer.

3.3 Rich Context

"Rich Context" information is provided on varying levels of granularity. At the highest level an application may want to only listen for "meeting" events but once this event is raised it needs to know the components of the information used to form this decision, and so on. In this way, the Intelligent Environment allows application designers to build risk-adverse context aware applications depending on the granularity of the information to be provided to the end user. Instead of your "smart phone" automatically redirecting an incoming call because it "thinks" you are in a meeting, the context information for both the sender and receiver are provided to both parties to facilitate a smart decision being made.

Smart environment agents also provide a means for the intelligent environment to update information about people using the environment as their associated context or rich-context changes. For example, as a person moves through the physical environment their associated user model is updated with the changing location information [Riché and Brebner, 2003].

3.4 Architecture: User Modelling

One class of agent assessing the Intelligent Environment (external to Figure 1), is the Smart Personal Assistant which takes care of a number of chores for the user. In practice, such Smart Personal Assistants access and manage much of the user model, along with accessing context and rich-context from the IE repository to customise and personalise interactions on behalf of the user.

4 Realisation

In our architecture, context information is stored as fields of context objects that represent particular sensors or higher level context entities. For example, an environmental sensor may have an object with fields for temperature and light level. A higher level context entity such as a room may have a list of people that are currently in the room.

Context objects are modified by sensor agents or by smart environment agents. A sensor agent may have the task of monitoring a low level environment sensor and updating a field in the relevant context object. Smart Environment Agents in turn may monitor context objects and add or modify fields of other context objects. A Smart Environment Agent may monitor sensor objects for a room and infer that several people are present and meeting is in progress. This information is stored in a context object for the room.

The context repository is implemented as a type of distributed object database. The objects are stored at a set of servers distributed across the network in administrative domains (Fig 2). Servers are arranged in a tree and can locate parent and child servers.

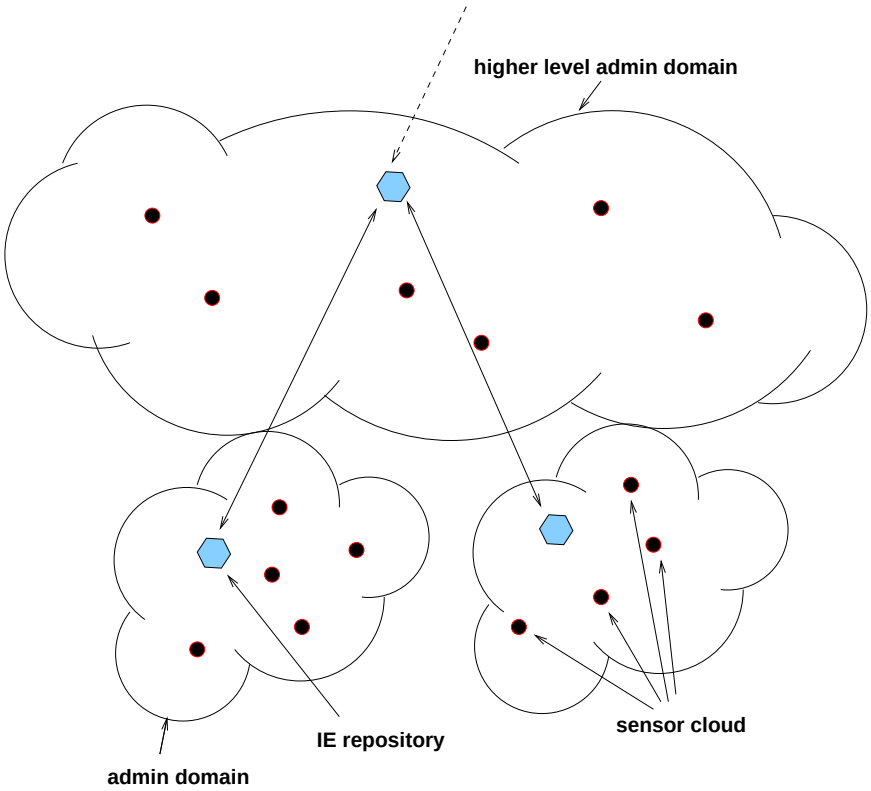

Fig. 2. Intelligent environment sensor cloud distribution and aggregation

Objects are referenced using a unique ID or URN. In order to find a given object a search is carried out starting at the local server and expanding through the tree in a similar way to Gloss [Dearle et al., 2003] and Globe [Ballintijn et al., 1999] systems. The key difference is that our search has a configurable limit by administrative domain. Beyond this limit the search switches to a distributed hash table approach. The reason for this approach is to restrict most searches to the local area where they are most likely to be satisfied and where the natural ownership of context data lies but still allowing a scalable solution that doesn't default to global flooding. Once the server for an object has been located, operations are then carried out directly with the server. Key features of the context repository are that each object has a unique ID and resides at a single point (server) in the system, and is located using a distributed search.

Communication at the local server level is carried out using a multicast message-based protocol. For our initial implementation we use the Elvin system [Fitzpatrick et al., 1999]. Communication between servers is also message-based but unicast.

Agent programs may register with a context repository server and ask to be notified when a specified event occurs such as a particular object being updated. This allows, for example, an agent to monitor a room for changes. Notification occurs using the same message protocol used in other parts of the system

For a complete example of the system in operation consider a person carrying a Bluetooth-equipped phone entering a room with a Bluetooth sensor. The sensor detects the presence of the phone and notifies its controlling agent which converts the (unique) Bluetooth ID of the phone into a URN. The agent locates the local context repository server and requests the home server for that URN. The server may be the local server for a phone that normally resides in the building but if not a search is carried out, eventually returning the server address or a not-found indication. The home server of the Bluetooth device object is then contacted and information added to the relevant object to indicate that the Bluetooth device has been detected by the sensor. Agents that have asked for notification when this object changes are then notified. Among these may be higher level agents that in turn generate further context information and update other objects which trigger notifications.

User models are stored in objects in the same distributed context database. These are complex objects that contain user profile information in the style of Personis [Kay et al., 2002]. This system includes a "view" mechanism that allows a subset of the user model to be retrieved. This view definition can be specified by the query agent, can be stored in the user model, or

is determined according to the query agent's identity. This feature allows the user model to present a set of "personas" to query agents.

Another feature of the Personis system is that values for components of the user model are calculated from evidence for particular values [Kay, 2000]. Evidence is accreted and when a value of the component is required it is calculated using an application specific resolver. This accretion approach has many advantages when used with context data: interpretation is delayed until a value is required and historical data (evidence) can then be interpreted (resolved) appropriately for an application.

An important feature of context-aware systems of the future is that the user should be able to find out what the system is doing on their behalf. Context data and associated inferencing systems must be inspectable or *scrutable*. This is a key feature of our user modelling system [Kay, 1995].

5 Summary and Current Status

In many current context systems there is a strict distinction between low level, sensor context information and high level context information created as a result of an inference activity. In our system the context repository unifies low and high level context information along with the user model.

Our architecture also unifies the user model with context information through the use of the *accretion* approach to managing possibly unreliable context data and the *views* mechanism for presenting different personae to context aware applications.

A prototype repository has been built for a single domain. Bluetooth access points are being used as simple sensors of Bluetooth devices with a sensor agent gathering location information and publishing it to the repository. Some simple applications have been developed to demonstrate tracking of people carrying Bluetooth-equipped phones in our building. Work is continuing on integration of the Personis user modelling system, inter-repository communication, and the representation of context information in standard formats.

References

[Ballintijn et al., 1999] Ballintijn, G., van Steen, M., and Tanenbaum, A. S. (1999). Exploiting location awareness for scalable location-independent object IDs. In *Fifth Annual ASCI Conference*, pages 321–328, Heijen, The Netherlands.

[Brown, 2001] Brown, E. S. (2001). Project Oxygen's new wind. *Technology Review*.

[Busboom, 2002] Busboom, A. (2002). Delivery context and privacy. In *W3C Workshop on Delivery Context*, Sophia-Antipolis, France. W3C.

[Dearle et al., 2003] Dearle, A., Kirby, G., Morrison, R., McCarthy, A., Mullen, K., Yang, Y., Connor, R., Welen, P., and Wilson, A. (2003). Architectural support for global smart spaces. In *Proceedings of the MDM'2003*, pages 153–164, Melbourne, Australia. Springer-Verlag.

[Fitzpatrick, 1998] Fitzpatrick, G. (1998). *The Locales Framework: Understanding and Designing for Cooperative Work*. PhD thesis, Department of Computer Science and Electrical Engineering, The University of Queensland.

[Fitzpatrick et al., 1999] Fitzpatrick, G., Mansfield, T., Kaplan, S., Arnold, D., Phelps, T., and Segall, B. (1999). Instrumenting the workaday world with Elvin. In *Proceedings of the ECSCW'99*, pages 431–451, Copenhagen, Denmark. Kluwer Academic Publishers.

[Grimm et al., 2000] Grimm, R., Anderson, T., Bershad, B., and Wetherall, D. (2000). A system architecture for pervasive computing. In *Proceedings of the 9th ACM SIGOPS European Workshop*, pages 177–182, Kolding, Denmark. ACM, SIGOPS 2000.

[Hawick and James, 2003] Hawick, K. A. and James, H. A. (2003). Middleware for context sensitive mobile applications. In Chris Johnson, P. M. and Steketee, C., editors, *Proceedings of the Australasian Information Security Workshop and the Workshop on Wearable, Invisible, Context-Aware, Ambient, Pervasive and Ubiquitous Computing*, volume 21 of *Conferences in Research and Practice in Information Technology*, pages 133–142, Adelaide, South Australia. Computer Science Association, Australian Computer Society.

[Kantor, 2001] Kantor, M. (2001). *Creating and Infrastructure for Ubiquitous Awareness*. PhD thesis, University of California, Irvine, California, USA.

[Kautz et al., 2003] Kautz, H., Etzioni, O., Fox, D., Weld, D., and Shastri, L. (2003). Foundations of assisted cognition systems. Technical Report CSE-03-AC-01, Dept of Computer Science & Engineering, University of Washington.

[Kay, 1995] Kay, J. (1995). The um toolkit for cooperative user modelling. *User Modeling and User-Adapted Interaction*, 4(3):149–196.

[Kay, 2000] Kay, J. (2000). Accretion representation for scrutable student modelling. In *Proceedings of Intelligent Teaching Systems 2000*, pages 514–523. Springer-Verlag.

[Kay et al., 2002] Kay, J., Kummerfeld, B., and Lauder, P. (2002). Personis: a server for user models. In *Proceedings of Adaptive Hypertext 2002*, pages 203–212. Springer-Verlag.

[Kidd et al., 1999] Kidd, C. D., Orr, R., Abowd, G. D., Atkeson, C. G., Essa, I. A., MacInyre, B., Mynatt, E., Starner, T. E., and Newstetter, W. (1999). The Aware Home: a living laboratory for ubiqutous computing research. In *Proceedings of the Second International Workshop on Cooperative Buildings*, Pittsburgh, USA.

[Kindberg et al., 2000] Kindberg, T., Barton, J., Morgan, J., Becker, G., Caswell, D., Debaty, P., Gopal, G., Frid, M., Krishnan, V., Morris, H., Schettino, J., and Serra, B. (2000). People, places, things: Web presence for the real world. Technical Report HPL-2000-16, HPLabs, http://www.cooltown.hp.com/.

[Phillips, 1999] Phillips, B. (1999). Metaglue: A programming language for multi-agent systems. Master's thesis, MIT, Cambridge, MA.

[Riché and Brebner, 2003] Riché, S. and Brebner, G. (2003). Storing and accessing user context. In *Proceedings of the MDM'2003*, pages 1–12, Melbourne, Australia. Springer-Verlag.

[Salber et al., 1999] Salber, D., Dey, A. K., Orr, R. J., and Abowd, G. D. (1999). The Context Toolkit: aiding the

development of context-enabled applications. In *Proceedings of the 1999 Conference on Human Factors in Computing Systems*, pages 434–441, Pittsburgh, PA.

[Schmidt and Beigl, 1998] Schmidt, A. and Beigl, M. (1998). There is more to context than location: Environment sensing technologies for adaptive mobile user interfaces.

[Schreck, 1997] Schreck, J. (1997). Security and privacy issues in user modeling. In *UM97: The Sixth International Conference on User Modeling*.

[Weiser, 1991] Weiser, M. (1991). The computer for the twenty-first century. *Scientific American*, 265(30):94–104.