

A context inference and multi-modal approach to mobile information access

David West
School of Information
Technologies
University of Sydney
Sydney, Australia
dwest@it.usyd.edu.au

Trent Apted
National ICT Australia
Australian Technology Park
Eveleigh, Australia
trent.apted@nicta.com.au

Aaron Quigley
School of Information
Technologies
University of Sydney
Sydney, Australia
aquigley@it.usyd.edu.au

ABSTRACT

Multimodal context-aware computing offers system support that incorporates the environmental, cognitive and computational state of an individual while allowing them to compose inputs and receive outputs across a range of available modalities. Here we present two support planes from our ubiquitous system architecture, entitled *Nightingale*. These include a multi-modal application framework and context-management and inference approach based on ontologies. In *Nightingale* the devices in use form a personal area network (*PAN*) with access to *local* (*PLAN*) or *remote* (*PWAN*) devices or services as they become available. Our proposed architectural elements aim to simplify multimodal and multi-device interaction, as the proliferation of small personal and inter-connected computing devices increases.

ACM Classification Keywords

H.3.4 [Information Storage and Retrieval]: Systems and Software; H.5.2 [Information Interfaces and Presentation]: User Interfaces—Interaction styles; I.2.4 [Artificial Intelligence]: Knowledge Representation Formalisms and Methods

INTRODUCTION

Typically the goal of a pervasive computing system is to allow people to access their information or information services regardless of the actual technology currently available [5 16 18]. Such a goal presents a number of research challenges including security, mobile data management, human computer interaction, and distributed computing. In particular “multimodal interfaces” and “context awareness” are emerging as major trends in the realisation of pervasive computing systems. *Multimodal interfaces* allow individuals to interact with systems using a *combination* of their innate faculties (e.g. touch,

sight, hearing) or common skills (e.g. speaking, reading, writing) [3 7]. *Context awareness* allows individuals to interact with systems which are *aware* of the environmental state (e.g. location, workgroup, activity) and computational state (e.g. applications, devices, services) of an individual [6].

Consider a home environment, such as that demonstrated in the EasyLiving project[4], which has been instrumented to collect contextual information and offer multi-modal applications. In such a set-up there are embedded computers and control elements, identification and authentication elements, a home network, an entertainment or media hub, wireless networking, service discovery, middleware and operating system support, handheld computers for data access throughout the home. Within the home environment an example multimodal application may incorporate a speech, universal remote and gestural input to a stereo system. An action such as a button press, a pointing gesture to the stereo and a spoken command of “Set this preference button to that stereo” will be resolved by the home system, to correlate to a certain button press with a given set of preferences for the home stereo. Within the EasyLiving environment “disaggregated computing” allows for a user’s location and preferences to determine which set of inputs and outputs, across a set of computers, were connected to the currently active applications. While managing, resolving and fusing the inputs is a challenging task, so too is the problem of coordinating the devices, preferences and services across the home for a set of users[5 8]. Further complicating this issue is the ability for a user to cognitively deal with a large permutation set of options both within the home and while on the move [21]. In such an environment a context-aware system manages and stores environmental data, device profiles, resources and user preferences. The goal is to simplify, through automatic (learnt or specified) or semi-automatic means, the multi-device, multi-user, and multimodal system support issues.

Our motivation for this research is to realise a pervasive computing system for supporting applications through non-desktop interfaces for accessing, organising and interacting with your own suite of “personal server” devices. The rest of this paper is organised as follows,

This research is funded through the Nightingale Project and is supported by the Smart Internet Technology Co-operative Research Centre and the National ICT Australia. National ICT Australia is funded through the Australian Government’s *Backing Australia’s Ability* initiative, in part through the Australian Research Council.

the background section describes related and ongoing work in multimodal and context-aware computing, the Nightingale section describes our multimodal application framework and our approach to context representation and inference; the final section outlines our conclusions and future work in this area.

BACKGROUND

Multimodal Interfaces

Seminal work on multimodal interfaces includes Bolt's "Put That There system" in 1980 [3] that involves the *fusion* of speech and pointing gestures to create, move, copy, name and remove objects on a large 2D screen. Input fusion attempts to combine the inputs from multiple modalities, either directly or through recognition, to form an aggregate or disambiguated input. For example in [3], a user can issue a command such as "create a yellow triangle there". Deictic references were resolved by referencing the object at the x, y coordinate indicated by the cursor at the time the reference was spoken. In addition, the user could name objects for future referencing by issuing a command such as "call that *my object*". When the system recognises command utterance it immediately switches to training mode so the name of the object can be learnt by the system.

Since this early work a number of multimodal applications [14] and architectures [7 10] have been developed. One common infrastructure in multimodal architectures is a multi-agent architecture, in which the various components of the multimodal system, such as speech recognisers, gesture recognisers, application processing logic, etc., are implemented as semi-autonomous *agents* which communicate through a support infrastructure. Where an agent is a software element that acts on, or has the power or authority to act on behalf of another element. The QuickSet system [7] is the basis for a number of military research applications, integrating a pen and speech multimodal system. QuickSet goes beyond using a pen for simple pointing commands, recognising more complex gestures, such as lines and arrows, via a neural network and a set of Hidden Markov Models (HMM). The system employs a logic based approach to multimodal fusion involving the use of *typed feature structures* and *unification-based integration* [14]. Subsequent research based on the QuickSet approach and others has shown that combining the input from various input modalities can reduce error-rates in interpreting user intent by as much as 40% in multimodal based systems as opposed to systems using only one modality for input [14]. In effect, there can be mutual disambiguation between input signals, which can be used to stabilise individual inherently error-prone recognition technologies, like speech and hand-writing recognition.

In parallel to many of these research efforts, the W3C have specified a multimodal interaction framework [2] to support the deployment of multimodal content on web-

enabled devices. The framework identifies the major components of multimodal systems, and the markup languages used to describe information required by components and for data flowing between components. The input component consists of various elements for the recognition, interpretation, and integration of input modes, such as pen, speech, hand-writing, etc. Raw input signals from the user are sent to the recognition components specific to each modality. The recognition component may use a grammar described by a grammar mark-up language to produce useful input to the interpretation component. The interpretation component involves semantic interpretation of the output from the recognition stage.

The W3C are standardising an XML based language called the Extensible MultiModal Annotation Language (EMMA) to represent the input to the interaction manager in an application defined manner. An optional integration component takes the EMMA results from various modalities, unifies them, and sends the combined result to the interaction manager. The interaction manager coordinates the dialogue between the application and the user. This hosts the interface objects (input and output) as DOM based interfaces. Application state is stored in the session component, allowing the interaction manager to support state management, and temporary and persistent sessions for multimodal applications. The system and environment component supports application knowledge of device capabilities, user preferences and environmental conditions. Application output passes from the interaction manager to the generation component. The generation component is responsible for which mode, or modes, will be used to present information to the user. The styling component for a particular modality accepts output from the generation component. This converts the modality neutral output into a format suitable for display by the particular output modality, using a style sheet mark-up language (such as CSS). The styled output is finally passed to the rendering component which renders the output to produce the output signal.

Context

While originally from linguistics, *context* is generally taken to mean "that which surrounds, and gives meaning to, something else". However, early research in context-aware application programming focused almost exclusively on *location* and *time* [5]. Schmidt et al. presented some ideas for furthering these notions of context at a low level using *Sensor Fusion* and later presented the European Project TEA (Technology for Enabling Awareness) [20], which expanded the notions of context to include light level, pressure and proximity to other people, as well as a *resolution* layer to determine a user's *activity*.

Before this work, the narrow view of context meant that sophisticated modelling, resolving and naming techniques were not required. For most applications,

all that was required was the choice of a *symbolic* or a *geometric* model of location. Schilit [19] was the first to present an *architecture* for context aware computing, but again the focus was on location and it did not involve *resolution* into higher levels of context. CyberDesk deals only with a different type of context information – a user’s selected text – but took a modular approach to context acquisition, abstraction and client services that later inspired the Context Toolkit [8]. However, in the Context Toolkit, context data is not separated from the sensors, and the path context flows through the system must be manually specified by application developers [9].

In order to share context between applications, there must be a standard way of modelling the context information and standard protocols for the exchange of this information. There are many tools to accomplish the exchange of generic data between heterogeneous applications including CORBA and Jini. However, an infrastructure should be as agnostic as possible with respect to hardware platform, operating system and programming language [9]. Furthermore, Ranganathan and Campbell [16] observe that such generic middleware falls short of providing ways for clients to be context-aware. They proposed another layer to facilitate the acquisition and reasoning of context, as well as the modification of clients’ behaviour.

The work of Ranganathan et al. on the GAIA project [17] involves development of their own middleware layer built over CORBA. They use ontologies to describe context predicates in their middleware in order to facilitate various types of inference, including rule-based and machine learning approaches. This enables their agents in different ubiquitous computing environments to have a common vocabulary and a common set of concepts when interacting with each other, while their Ontology Server assists with configuration management and context-sensitive application behaviour.

Strang et al. [21] investigation of ontologies for pervasive computing has inspired our research into this area. Rather than defining a single, monolithic language, they decompose their *Context Ontology Language (CoOL)* into a collection of several, grouped fragments and reach a high degree of formality by using ontologies as a fundament for their *Aspect-Scale-Context (ASC)* model. This model allows context information to be *characterised* and thus shared (and queried) independently of its representation. Wang et al. [23] are also looking at ontologies for modelling context in pervasive computing environments; specifically to leverage existing Semantic Web logic-based reasoning techniques. They have investigated the scalability of ontology-based context reasoning (using a framework they have called *CONON*) compared to user-defined, rule-based reasoning.

NIGHTINGALE

Project Nightingale aims to explore pervasive computing interaction mechanisms beyond the classical screen, key and mouse. As such, this research touches on aspects which include natural and adaptive interfaces, mobile data management, trusted wireless networking standards and personalisation. In this section we discuss two of our higher level architectural elements; namely multimodal interfaces and context-aware computing.

Contribution

Most current multimodal applications and frameworks have been designed for operation on a single device or a limited number of closely coupled devices [14 10]. A number of these systems take an agent-based approach to support distributed processing, where agents are closely tied to a single or small number of applications. Agent-based approaches are typically structured around an input agent providing an interpretation of users’ intent, involving application-specific logic, and the knowledge of which application agents they must bind to in advance.

Providing support systems that offer context awareness and multi-modal interaction has received little attention. When considering mobile multi-device scenarios there is a clear need to create scalable and reusable architectures that can cope with sensed environmental context data along with the the myriad devices, input modes and signals that will be routed to various applications – themselves residing on arbitrary devices; for example, applications running on a user’s mobile phone accessing services from their PDA.

Such architectures should support *perceptual spaces* [13], in which the input provided by users is interpreted within the pervasive computing environment’s overall computing context. A goal of such environments will be to minimise the number, size and configuration settings for the devices that a user has to carry. Ideally, the environment will contain ambient, embedded devices, such as embedded cameras and directional microphones. Pervasive environments involving context-aware multimodal computing will move away from the assumption in present systems that a single user provides all input to a device in a given period. Instead, the context-aware environment will need to determine each user’s individual contribution to the input signal.

The primary motivating goal for the Nightingale multimodal system has been to decouple multimodal applications from particular input devices; giving the user the ability to roam seamlessly between her PAN, PLAN, and PWAN environments, allowing her to access her full suite of application services, while taking advantage of the input and output devices that she encounters in a seamless manner. Our architecture allows a user to approach any input device or devices of arbitrary modality available in her surroundings, and to immediately use

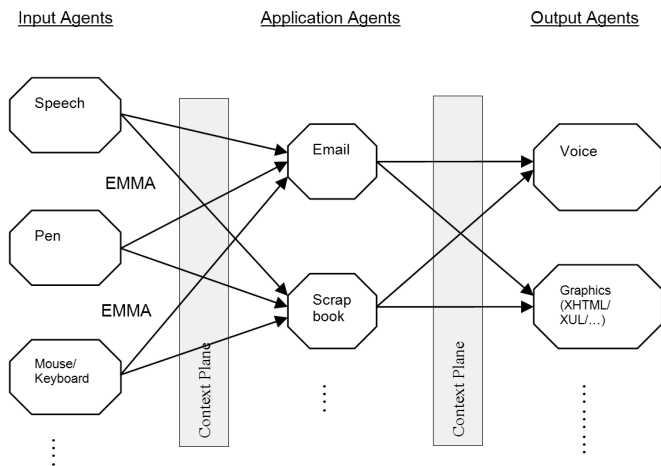


Figure 1: Mediated Input and Output in EMMA form

them to continue her multimodal dialogue. After the invocation of a user identification component, the input device will be automatically tailored to provide an interpretation suitable for her current application processing context. As her application processing context changes, the interpretation of her intentions will be modified on the fly. The following section describes our agent based architecture tailored for such pervasive computing environments.

Multimodal Architecture

The Nightingale model consists of an agent based architecture spanning multiple autonomous computing devices in a user's PAN, PLAN, and PWAN, as illustrated in Figure 1. Application agents reside on the user's home server, his office computer, or other computers. Agents currently coordinate through a simplified context-plane service. The aim is to make available through the context-plane the following aspects of service: details of the devices and services available, the various inputs and outputs that are available, historical usage patterns, current sensed computational and environmental context along with aggregated and inferred contexts. For example, when the context-data indicates the user is in the minimal connectivity PAN environment, application processing will typically take place on his personal server. Input and output agents may reside on embedded systems within the user's environment (an embedded microphone in his vicinity for example), or they may reside directly on his personal server (an agent controlling pen-based input through a paired Bluetooth Anoto pen for example).

Input agents send application defined, modality neutral input to application agents in the form of EMMA, part of the W3C multimodal application framework. EMMA is a language for exchanging data between the interpretation and integration/interaction manager components. It consists of:

Instance data consisting of an application specific interpretation of user intent. The format of the instance data is meaningful to the consumer of the EMMA document. Because of the ambiguity of interpreting various input sources, there may be more than one interpretation. This facilitates mutual disambiguation through fusion with other input modalities or the use of historical or current context data.

Data model may be implicit. Specifies constraints on the format of the instance data. E.g. an XML schema or DTD for the XML instance data.

Metadata allows the association of various pieces of information (annotations) about the produced instance data. E.g. timestamps (for multimodal fusion), the producing media (e.g. speech), confidence scores in the interpretation, information about the process that produced the interpretation, etc.

Applications agents have a generation component to distribute modality neutral, application defined output to suitable modalities. We are exploring fission mechanisms for multimodal output, taking into consideration user context such as current location, ambient noise level, available modalities, user preferences, etc., as discussed in the following section.

The upper half of Figure 2 shows how the raw input signal from the user is converted into application defined input as EMMA documents. Applications supply modality specific grammars and interpreters to input agents. The grammar controls input recognition by constraining what is recognised in the input signal. Grammars can consist of context free grammars such as for speech. We anticipate in the future that the use of context will drive both how the recognition is refined and how the subsequent fusion is achieved. Recognised input is passed to the semantic interpretation component in the input agent. The application supplied interpreter converts raw recognised input into application specific meaningful input, in the form of EMMA.

The lower half of Figure 2 shows how application defined output is converted into an output signal delivered to the user. The styling component converts application defined output into a format suitable for rendering by the modality specific rendering engine. An application must supply a style sheet to specify this process. For example, a multimodal email application may send a list of headers to an output agent. The style sheet will convert this to a format suitable for output using a text-to-speech renderer, say, by creating a string suitable for the text-to-speech engine to dictate. In addition to providing interpreted input and styled output, applications can also gain direct access to the raw signal through a *raw interface* component of each input/output agent. Thus applications which would like to record audio directly from the user can access the raw microphone input from a speech agent. As with input, we expect to

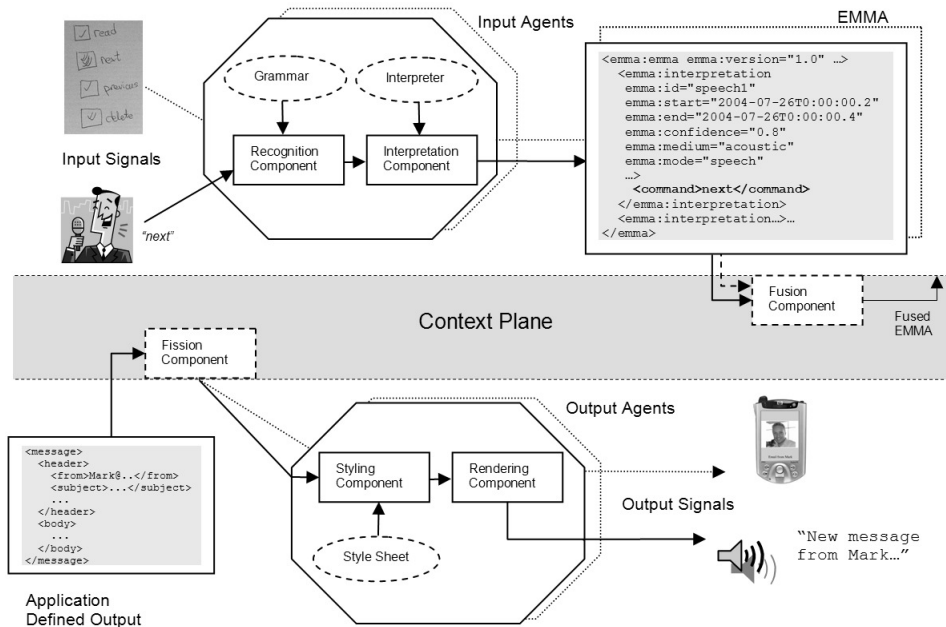


Figure 2. Application supplied components for input and output agents

offer context services to guide the application outputs as the context plane develops.

From the above description, it is apparent that the application must supply an application/modality specific grammar and interpreter to input agents, and an application/modality specific style sheet to output agents. The application writer must write these components. Once the identification component of an IO agent determines which user is to use it, the agent will load the necessary components for the user's current active application by mediation with the context and data planes. Our current implementation of the context plane uses a shared tuple-space. This provides the advantages of temporal and referential decoupling [15]. The bootstrapping process is described for a pen input agent, email application agent, and text-to-speech output agent in Figure 3.

The first step (not shown on diagram) is for the user to indicate they wish to use an IO agent. Currently, we use a manual GUI based login mechanism for this. We are investigating using context data in the form of biometric and signal identification mechanisms (e.g. recognising a particular user by their speech-/hand-writing). Agents must obtain the necessary application supplied components and application binding information for a new user. The other steps are as follows:

1. Applications register themselves with the user's command agent in advance. The command agent typically runs on the user's personal server. The pen agent requests the *ApplicationWithFocus* context for the given user. The command agent responds with

the unique agent ID of the email agent.

2. The pen agent requests the data plane location of the required grammar and interpreter for the email agent using the pen modality. The email agent responds with the locations.
3. The pen agent loads the grammar and interpreter from the data plane.
- 4-6. A similar process loads the styling component into the voice output agent. This also triggers the email agent to locate the binding information for the output agent by similar mediation with the context plane.
7. When a user speaks, commands relevant to the email application e.g. "read my email", "next message", "delete this message" are recognised. The grammar constrains the recognised commands, and the interpreter produces a modality neutral, application defined input to send to the email agent.
8. Fusion of the input signal with multiple input modalities may occur in the email agent. The command is executed, and modality neutral output is generated. The generation component sends the output to the relevant attached output agents, in this case the voice agent. The output is styled by the styling component, and rendered to produce an output signal.

The user's application processing context is controlled by the command agent. When an input agent is bound to a new user, in addition to loading an application control grammar/interpreter, a command grammar/interpreter is loaded. This separate grammar recognises

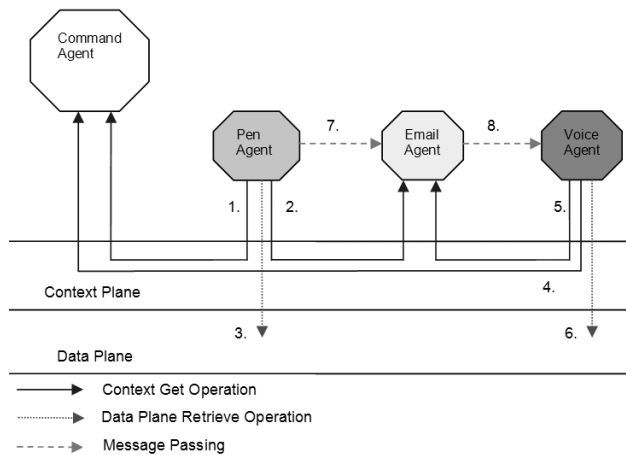


Figure 3: Mediated Input and Output agent configuration

commands which change the application processing context; specifically, the user’s current active application. In our current implementation, the user changes active application by explicitly requesting an application focus shift. For example, they may say, “Switch to my scrapbook application.” This will trigger all currently bound input and output agents to reload their application specific components and bind to the new application. We are investigating alternative strategies to allow users to control multiple application simultaneously using different modalities. In addition, we are investigating a model whereby applications can update the command grammar on the fly in response to changes in their internal state. For example, on response to a new email, the email application agent may update the active command grammars to allow the user to listen to the new email immediately, even if email is not the currently active application. We are also planning to provide a mechanism for applications to refine their own control grammars in response to changes in application state.

Software Architecture

The Nightingale multimodal architecture is an object oriented framework and is structured as a series of layers, beginning with the physical local data store and network. The data plane, not described here, is structured above this layer and provides persistence and distributed data management, including proactive caching strategies and application session support. Currently, the context plane consists of a tuple space for storing user, application, and IO agent context. The application plane consists of a generic agent based infrastructure, incorporating command agents, generic input and output agent support, and user identification support used by IO agents. The next layer contains the modality specific code for input, output and user identification. The application programmer writes the topmost layer, consisting of application specific

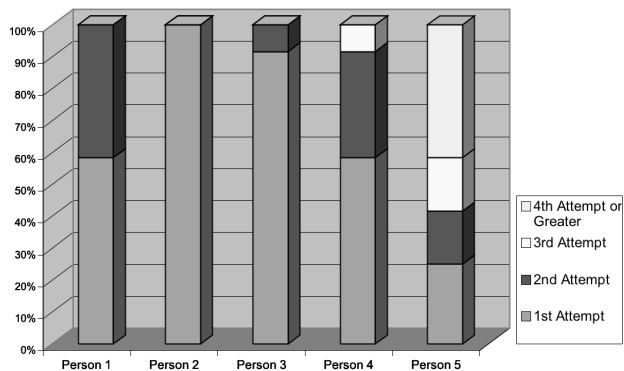


Figure 4: Gesture recognition results from draw-able user interface experiment.

logic. They define the structure of modality neutral application input and output. In addition, they must write the interpreters/grammars/stylers for the specific input and output modalities they wish to support.

Implementation

As a point in the design space for this architecture, we have developed an implementation incorporating pen and speech as input modes, and text-to-speech as output mode. We are planning a graphical IO mechanism based on XHTML/XUL. We have developed two applications to test our architecture, a multimodal email system, and a physical/digital scrapbook application. Our current implementation is written in Java to exploit cross-platform and byte code migration capabilities. Currently, LIME [12] is used to implement the context plane. By communicating on a shared multicast address, LIME offers the abstraction of the shared tuple space. All agents in multicast range of each other may coordinate in this fashion. To maintain scalability, agents should only coordinate with other agents in the user’s immediate vicinity. For this reason, the limited range of wireless technologies such as 802.11, or using the Bluetooth PAN profile, actually provides an advantage: only agents in range of each other, and consequently only those useful within a PLAN for the user, will coordinate their functions by merging their tuple spaces.

Application provided grammars, interpreters, and stylers have been implemented as Java classes. Application writers must define these components. The agent framework stores these classes in the data plane, and registers their locations in the context plane. When an input or output agent requests the location of the relevant components, the application agent framework responds with the data plane location. A custom Java class loader is then invoked to load the class as byte code directly from the data plane.

In addition to speech, we have developed a novel pen and paper based *drawable user interface*, using

the Anoto pen and paper system [1], as a means of controlling applications. Users may draw their own interface, consisting of labelled *buttons* i.e. squares and other shapes with hand-writing annotations. As such, a user drawn email interface might consist of buttons labelled *read*, *next*, *previous*, *delete*, etc (Figure 2). They “click” a button by ticking the box they drew, and this is interpreted by the pen agent to produce application defined input to application agents. The results of an early user study of this form of interface are shown in Figure 4. They record the number of attempts users took to draw correctly recognised interface items. While these results are promising they do point to the need for the use of historical context data along with other inference and gesture learning techniques.

In addition we have developed a physical/digital scrapbook using the Anoto pen and paper system to promote reminiscence and memory sharing activities among the elderly. Users use the physical scrap book in the same way the use a regular scrapbook; they paste in photos, newspaper clippings, etc., and they annotate them with the pen. In addition they can add digital items, such as audio annotations, into their scrapbook. By drawing a square labelled *audio* they will be prompted to record a piece of audio. By ticking the box later, the audio will be replayed to them using an appropriate audio output agent in their vicinity. All digital items, including pen strokes, are stored in the data plane to automatically publish the scrap book as a web page. To synchronise physical items in the scrapbook with the digital version, the scrapbook can be placed under the gaze of a high resolution digital camera.

Context Awareness

To simplify the integration of new users and devices in our pervasive computing environment, and to allow our applications to be context-aware we are developing a context *infrastructure* [9] – the *Context Plane*. The Context Plane actively seeks devices in the pervasive computing environment and is able to prioritise them based, not only on computing context, but on user and application preferences as well as physical context such as location and co-location with other users.

For example, a room might offer a large, wall-mounted public display screen and a user might have a personal device with a smaller screen, such as a PDA. In the past, the user has indicated a preference for the wall-mounted display by asking for a *modality switch* when offered smaller screens. However, when the user walks into this room, the currently running application is an email application that has informed the Context Plane that the content is confidential; thus it might be inappropriate to display on a public display. But the Context Plane also knows that there are no other users co-located with the current user (i.e. in the same room). Note that for some applications, it might also be feasible to utilise audio output devices such as a speaker or Bluetooth headset. The application is currently using

the PDA as a display device, so the Context Plane must decide what is the most appropriate output modality after the user walks into this room.

Our approach to the resolution of this context looks at combining simple user-driven, rule-based, probabilistic and temporal logic inference techniques. Further, we aim to explore the use of an evidential reasoning model to simplify the way in which an *application* can provide context to the Context Plane. An ontological approach is also being considered because it provides a flexible and scrutable manner in which to provide inference rules, as well as describing a common *language* for applications to describe such rules and provide context. However, this alone is not enough; we require not only to find a *feasible* solution to the selection of modality but also to determine the *best* solution, given the current context.

As we intend to pursue a *risk-adverse* context awareness, the best solution may come directly from the user, rather than a fully-automatic approach. To achieve this, we require a method for the user to override the decision. In this case we would select the *next most appropriate* modality; thus we also need to *rank* the possibilities. Furthermore, we aim to implicitly *learn* from this decision as there is evidence to suggest that, in the current context, the initially chosen modality was inappropriate in some way. Our initial approach is to explore reinforcement learning techniques.

Another aspect being considered is how to make decisions for *new* applications. That is, applications for which we do not have any historical evidence to draw from. In this circumstance we would like to harness evidence from other, related applications in proportion to their usage by the current user. Our scenario is similar to those motivating other context-aware projects such as iROS [15] and GAIA [16].

CONCLUSIONS AND FUTURE WORK

Our early work focused on the development of a series of multi-modal demonstrators across a range of devices. This work has informed the specification and design of our multimodal, context and data management architectural elements. While this incremental approach to research and development isn’t suitable in a mature research field, we have found it beneficial in a developing area such as pervasive computing, without established standards, operating system support or data protocols.

Our future work aims to evaluate our current demonstrators with a group of elders, the development of a peer-to-peer multimodal application framework, the exploration of ontology based inference methods for context, a lightweight distributed database for mobile data management and SIP based protocols for handoff between managements elements in Nightingale.

REFERENCES

1. Anoto. Development guide for service enabled by Anoto functionality. Technical report, Anoto, <http://www.anoto.com>, 2002.
2. M. Bodell, M. Johnston, S. Kumar, S. Potter, and K. Waters. W3C multimodal interaction framework. Note, W3C, <http://www.w3.org/TR/mmi-framework/>, May 2003.
3. R. A. Bolt. “Put-that-there”: Voice and gesture at the graphics interface. In *Proceedings of 7th annual conference on Computer graphics and interactive techniques*, 262–270, Seattle, Washington, USA, July 1980. ACM Press.
4. B. Brumitt, B. Meyers, et al. EasyLiving: Technologies for intelligent environments. In *Proceedings of 2nd International Symposium on Handheld and Ubiquitous Computing (HUC 2000)*, 12–29, Bristol, UK, Sept. 2000. Springer-Verlag.
5. G. Chen and D. Kotz. A survey of context-aware mobile computing research. Technical Report TR2000-381, Department of Computer Science, Dartmouth College, Nov. 2000.
6. H. Chen and T. Finin. An ontology for context aware pervasive computing environments. In Stuckenschmidt [22].
7. P. R. Cohen, M. Johnston, D. McGee, et al. QuickSet: Multimodal interaction for distributed applications. In *Proceedings of 5th ACM International Multimedia Conference*, 31–40, Seattle, Washington, USA, Nov. 1997. ACM Press.
8. A. K. Dey, D. Salber, and G. D. Abowd. A conceptual framework and a toolkit for supporting the rapid prototyping of context-aware applications. In *HCI* [11], 97–166.
9. J. I. Hong and J. A. Landay. An infrastructure approach to context-aware computing. In *HCI* [11], 287–303.
10. M. W. Kadous and C. Sammut. MICA: Pervasive middleware for learning, sharing and talking. In *IEEE PerCOM Workshops 2004*, 176–180, 2004.
11. T. P. Moran, ed. *Human-Computer Interaction Journal, Special Issue on Context-Aware Computing*, vol. 16. Lawrence Erlbaum Associates, Inc., Aug. 2001.
12. A. L. Murphy, G. P. Picco, and G.-C. Roman. LIME: A middleware for physical and logical mobility. In *Proceedings of the 21st International Conference on Distributed Computing Systems (ICDCS-21)*, 524–533, Phoenix, Arizona, USA, Apr. 2001.
13. S. Oviatt, P. Cohen, J. Vergo, L. Duncan, B. Suhm, J. Bers, T. Holzman, T. Winograd, J. Landay, J. Larson, and D. Ferro. Designing the user interface for multimodal speech and pen-based gesture applications: State-of-the-art systems and future research directions. *Human Computer Interaction*, 15(4):263–322, 2000.
14. S. L. Oviatt. *The human-computer interaction handbook: fundamentals, evolving technologies and emerging applications*, ch. 14: Multimodal interfaces, 286–304. Lawrence Erlbaum Associates, Inc., Mahwah, New Jersey, USA, 2003.
15. S. R. Ponnekanti, B. Johanson, et al. Portability, extensibility and robustness in iROS. In *Proceedings of IEEE International Conference on Pervasive Computing and Communications*, Dallas-Fort Worth, Texas, USA, Mar. 2003.
16. A. Ranganathan and R. H. Campbell. A middleware for context-aware agents in ubiquitous computing environments. In *Proceedings of ACM/IFIP/USENIX International Middleware Conference*, 143–161, Rio de Janeiro, Brazil, June 2003. Springer.
17. A. Ranganathan, R. E. McGrath, R. H. Campbell, and M. D. Mickunas. Ontologies in a pervasive computing environment. In Stuckenschmidt [22].
18. N. Reithinger, J. Alexandersson, T. Becker, et al. SmartKom – adaptive and flexible multimodal access to multiple applications. In *Proceedings of International Conference on Multimodal Interfaces 2003*, Vancouver, BC, Nov. 2003.
19. W. N. Schilit. *A System Architecture for Context-Aware Mobile Computing*. PhD thesis, Graduate School of Arts and Sciences, Columbia University, May 1995.
20. A. Schmidt, K. A. Aidoo, A. Takaluoma, U. Tuomela, K. V. Laerhoven, and W. V. de Velde. Advanced interaction in context. In *Proceedings of the 1st International Symposium on Handheld and Ubiquitous Computing (HUC’99)*, 89–101, Karlsruhe, Germany, Sept. 1999. Springer-Verlag.
21. T. Strang, C. Linnhoff-Popien, and K. Frank. CoOL: A context ontology language to enable contextual interoperability. In *Proceedings of 4th IFIP WG 6.1 International Conference on Distributed Applications and Interoperable Systems (DAIS2003)*, 236–247, Paris, France, Nov. 2003. Springer-Verlag.
22. H. Stuckenschmidt, ed. *Proceedings of the IJCAI’03 Workshop on Ontologies and Distributed Systems*, Acapulco, Mexico, Aug. 2003. Online proceedings: <http://www.cs.vu.nl/~heiner/IJCAI-03/>.
23. X. H. Wang, D. Q. Zhang, T. Gu, and H. K. Pung. Ontology based context modeling and reasoning using OWL. In *Proceedings of 2nd IEEE International Conference on Pervasive Computing and Communications*, 18–22, Orlando, Florida, USA, Mar. 2004. IEEE Computer Society Press.