3-31-2009

Ubiquitous Information Systems (UBIS): A design research study of intelligent middleware and architecture

David Bell
School of Information Systems, Computing and Mathematics, Brunel University,UK, david.bell@brunel.ac.uk

UBIQUITOUS INFORMATION SYSTEMS (UBIS): A DESIGN RESEARCH STUDY OF INTELLIGENT MIDDLEWARE AND ARCHITECTURE

David Bell
School of Information Systems, Computing and Mathematics
Brunel University, Uxbridge, United Kingdom.
Email: david.bell@brunel.ac.uk

Abstract

Ubiquitous information systems (UBIS) adapt current Information System thinking to explicitly differentiate technology between hardware devices and software components in relation to people and process. More recent ubiquitous computing approaches provide the means to link Web content and services to a number of mobile devices (evolving from earlier Palm Computers to more recent smart phones and ambient screens), adapting information to provide mobile business solutions. In general, these approaches focus on providing the means to improve specific information access and transcoding but not on how the information can be discovered and accessed on-the-fly. This paper explores how a number of investment banking systems can be re-used to provide the invisibility of pervasive access and uncover more effective architectural models for strategies of this type. A proof-of-concept intelligent middleware Web service is built to further test and explore how human-devices-application connections can be made sporadically and not limited to pre-configured access to specific applications and data.

Keywords: Ubiquitous/Pervasive Computing, Middleware, Design Research

1.0 Introduction

The Ubiquitous computing (Ubicomp) goal of an enhanced computer that makes use of the many computers embedded within the physical environment - effectively invisible to the user - impacts all areas of computing, including hardware components, network protocols, interaction substrates (e.g. software for screens and haptic entry), applications, privacy, and computational methods (Weiser 1993). Invisibility within the physical environment is a central theme in Ubicomp. Computer scientist, economist, and Nobel Prize recipient Herb Simon calls this phenomenon "compiling“; Philosopher Michael Polanyi calls it the "tacit dimension"; Psychologist TK Gibson calls it "visual invariants“; Philosophers’ Georg Gadamer and Martin Heidegger call it "the horizon" and the "ready-to-hand“; John Seely Brown at PARC calls it the "periphery" (Weiser 1991). “All say, in essence, that only when things disappear in

this way are we freed to use them without thinking and so to focus beyond them on new goals" (Weiser 1991 p. 933). In order to de-couple the information that we associate with our current applications and move it into the periphery, a means to transform the content and support the user in their current place is required. Large numbers of devices, variation in how information should be changed and the large number of source systems combine to make architecting such system challenging. In response, a number of vendors have simplified the problem by focusing on specific parts of this complex network. AvantGo were early entrants into the market, focusing on a single device (the Palm Pilot) and a standard data format. IBM unsurprisingly supplemented their application server (WebSphere) with a number of supporting tools that address both data transcoding and pervasive device support. More recently, smart phones (such as the Apple iPhone, Google G1 and Blackberry Storm) are proliferating large numbers of small applications that each connects to a range of internet data services (e.g. combining barcode readers and search).

The vision of Ubicomp is also well represented in the research community with many practical and futuristic projects – notable projects have included human-centred computing in Project Oxygen at MIT, wireless device access in HP Cooltown and in-house telecare for the elderly at Brunel University. Difficulties arise when trying to overlay such a vision on top of current information system architectures. One could be disappointed that Weisner's Ubiquitous Computing vision has not been realised, especially since many of the hardware and network bottlenecks have been overcome. Instead we have a number of intelligent mobile devices each competing in a narrow market segment (media downloading, e-mail integration etc.). Only by widening (and connecting) research at all levels from hardware design to information system strategy can substantial progress be made.

The web is awash with intelligent recommender services, but the same level of ubiquitous intelligence has still to percolate into the main stream business environment. In order to illustrate some relatively simple applications of intelligent ubiquitous information systems (not currently utilising ubiquitous technology in this manner), two banking scenarios are presented. It is also worth highlighting that the underlying devices, databases and applications already exist and are in use within the working environment described - their ubiquitous re-use is not however.

Scenario A: After flying from New York to London, a sales manager is entering the lobby in the regional headquarters of a global investment bank. Swiping their ID card through the turnstiles allows them to enter the waiting area for the lifts where they are faced with a large flat screen display. Instead of repeating the corporate marketing advertisements a report of the regional sales data is rendered – displaying data that is relevant to an imminent meeting. In addition, urgent messages are displayed (taken in part from the US cellular telephone).

Scenario B: Three traders are standing within viewing distance of one of their trading workstation and discussing the current economic climate. Their radio frequency identifiers (RFIDS) detect their proximity to the display where software automatically starts and displays financial news that is relevant to each of them.

This paper presents an intelligent, loosely coupled approach to ubiquitous information systems and is structured as follows. Section 2 covers some of the characteristics of Ubicomp and literature on relevant integration middleware. Section 3 presents the design research method deployed in this research and the *constructed* artefacts generated whilst undertaking the design. Section 4 reports on progress with this early research and identifies a number of possible avenues for continued research within the area.

2.0 Devices to Information Systems

2.1 Web of Devices

A future Web containing a heterogeneous mix of software services and embedded, mobile devices offers many opportunities to more effectively interact with enterprise applications (extracting information for rendering on appropriate devices at optimal times in appropriate places). One approach is to extend existing technologies and platforms to support such complexity and IBM's Websphere Everyplace Suite (WES) has followed this strategy. IBM extended their application server platform to include a number of tools for pervasive synchronisation of e-mail to mobile devices, transcoding data (converting data for specific devices) and radio frequency identifier (RFID) management. Another early middleware provider is AvantGo; providing software services to synchronise content from the Web onto a Palm Pilot mobile

device. AvantGo middleware supports both general purpose access to Web sites such as news as well as corporate data distribution (e.g. financial analysis provided by banking institutions to specific clients). IBM and AvantGo both provide clearly targeted technical innovations – enabling a pre-configured service that has been identified and constructed for a user community. In order to realise an increased level of *invisibility* in the UBIS vision we need to explore the interface between the human (with their mobile or ambient lens) and the source system to which they may want to interact. To undertake a task of this type requires appropriate theoretical underpinning and we must first understand ubiquitous architecture as a whole. Mukherjee and Saha (2003) provide a comprehensive schematic of the entities and layers of a ubiquitous architecture. Figure 1 is an extended version of their architecture with an additional focus on software services – both alongside applications and resident on the mobile device.

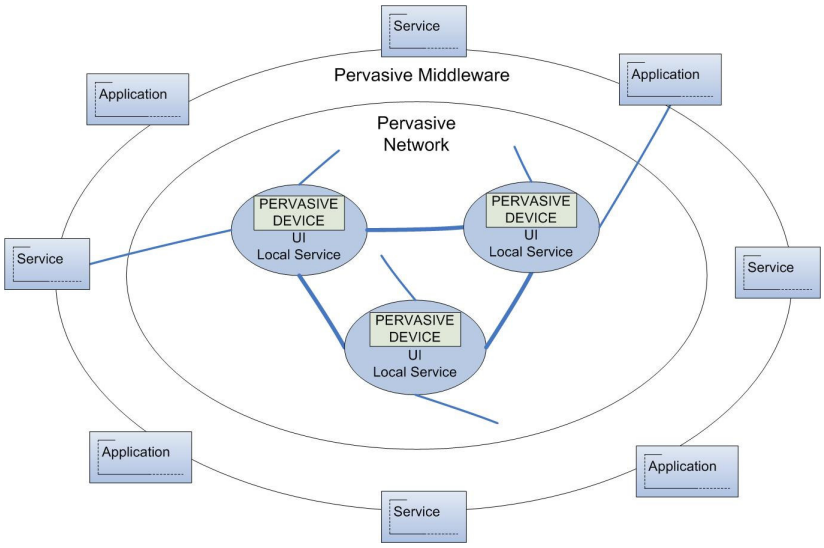

Figure 1. Ubiquitous Architecture (adapted from Mukherjee and Saha 2003).

Working outwards (in Figure 1) the pervasive network is made up of a number of devices (sensors, actuators, user interfaces etc.) that are interconnected on one or more networks. These devices can interface together in order to undertake specific tasks, for example peer to peer content distribution, or with remote applications and services. In order for the mobile devices to access remote services or applications they must first utilise some pervasive middleware. Original middleware definitions describing a middleware service as general purpose services that sit between

platforms and applications (Berstein 1996) appear too general (in light of a UBIS environment with application components sitting on many devices and applications) and requires additional clear and usable constructs to progress further. To start this exercise, a number of approaches to integration middleware, from the research literature, are contrasted. Before proceeding though, it is worth emphasising the positive (and much needed) contribution that Information Systems is able to provide to the research area, bringing together a much needed focus on human, technological and societal dimensions.

In a more pragmatic vein the architecture in Figure 1 can be used to construct a high level architecture. Indentifying the applications, services and devices that are within scope of the design before mapping networks and middleware required to create connections between chosen components. Traditionally, these connections would be envisaged at design time. Connections between specific applications and devices would be designed – choosing or developing appropriate middleware and network systems. This approach is clearly demonstrated in the commercial strategies already described.

2.2 Integration Middleware

In addition to some of the commercial platforms already discussed a number of middleware specific research project have been undertaken. Three themes are clear in the literature – software engineering, architecture and data support.

The AMUN middleware (Trumler et al. 2006) uses autonomic principles and the JXTA platform to investigate an office of the future. The middleware itself focuses on events and central control and configuration. Coronato and Pietro (2005) used grid middleware to provide access to services. Again, a heavy reliance on a specific software platform limits the fluid and invisible emergence of data and services. The importance of awareness and transparency is highlighted by Yau et al. (2002) when presenting their RCSM middleware. Their approach uses the Interface Description Language (IDL) as a means to support a number of underlying technologies. Extending this abstraction further to use XML would appear promising and increase the realisation of some of the benefits they state.

A number of researchers have used agent based middleware to interact in a context sensitive way with services. Soldatos et al. (2007) present a taxonomy of ubiquitous components: transparent ad-hoc communication, capture and transfer of sensor streams, raw signal processing, context acquisition and decision making. In addition they propose a middleware of agents (ranging from perceptual components such as face recognition to service agents); developed in C++ and JADE (Java Agent DEvelopment framework) and providing a memory jogger system. The technology centric approaches demonstrate a valid means of realising a UBIS architecture, but often lack wider architectural cohesion.

Soldatos et al. (2007) highlight the difficult balance between transparency and context awareness with technology pre-configuration (both hardware and software). In addition to technology centric middleware, research focusing on the planning of component and service execution (Rouvoy et al. 2008) has been undertaken, optimising the utility of applications (a customer relationship management scenario in this case) as context changes occur. At the heart of the planning system is a quality of service (QoS) plan and an associated plan repository (requiring that specific interactions are modelled up-front).

In order to realise a UBIS vision, a wider architectural framework is required that lends itself to grounding subsequent technology choices and activities. A substantial amount of current Ubicomp research focuses on tailored technology solutions and not how they are best able to fit into a larger architectural vision.

3.0 Design Approach

3.1 Design Framework

Recent activity around design research methodology has provided a number of frameworks for research artefact generation – with artefacts often categorised using March and Smith's (1995) terminology of concepts, models, methods and instantiations. Artefacts are wide ranging and examples include data sets, methodologies and digital media. A consequence of focusing on the iterative design and implementation of more effective products and processes are a catalogue of interlinked design artefacts each generated over the life of the research project. Subsequent analysis at both a holistic level and of each artefact's unique life-cycle is

limited without detailed recording of artefact characteristics over time (including their origination). Expanding analysis to evaluate against existing artefacts (typically applications or databases that are external to the design exercise) has additional difficulties, typically the result of many external artefacts being documented at a high level or not at all. In summary, the effective use and reuse of design research artefacts is heavily reliant on comprehensive capture of artefact detail (the same knowledge capture limitations are compounded when re-using artefacts across research projects).

The foremost question in ubiquitous information system architecture is how component synthesis should be undertaken more effectively – bringing together hardware devices and software/application services. The research described in this paper initiates this process by scoping the use of intelligent middleware and associated device, application and service knowledge. The research follows a design research approach, which is a "search process to discover an effective solution to a problem" (Hevner et al. 2004 p.88). The relevance of the problem for the research community must be demonstrated and the solution must be effective to a satisfactory level. The effective solution may not (and generally does not) coincide with the "best" or "optimal" solution however - generally the effectiveness of the solution must be demonstrable through an iterative evaluation of the designed artefact(s). The research process can be seen in Figure 2.

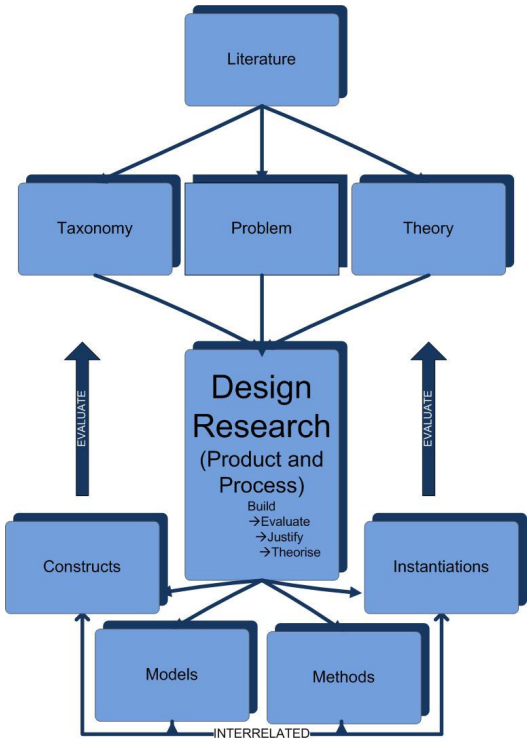

Figure 2. Design research framework (terminology from March and Smith 1995).

The design research process presented in this paper, and depicted in the diagram above, is methodologically based on and adapted from the approach described by Nunamaker et al. (1991) and the guidelines presented by Hevner et al. (2004). The research outputs are also described according to March and Smith's (1995) terminology for design research. The resulting inter-related artefacts will provide a theoretical model for progressing research with this area of middleware development.

- Theory building: The study is theoretically based on previous work conducted in the areas of ubiquitous middleware architecture and development. The proposed architecture builds upon this theory by covering an existing gap represented by a lack of integration between hardware device capability modelling and application/service software modelling.
- Constructs: Characterisation and categorisation of middleware at the heart of a UBIS has not been clearly articulated. The developed framework has been evaluated primarily through its conceptual

migration of a number of banking applications on a range of mobile devices. The scenario itself is not identifiable with a live project given the novelty of the research; however the applications and settings are drawn from live industrial projects. The application of the architecture to the scenario represents the development of a "proof of concept" project whose outcomes are to be evaluated in relation to the source applications from which they were taken.

- Models: Two sets of observations have been conducted. The first set of observations concerned the applicability of existing ubiquitous computing middleware architecture to fully represent the required migration. The development of component description and middleware was observed in order to understand how a typical software development process currently organises development components of this type. This observation allowed the research to understand the limitations of applications, services and devices described and developed with current industrial technology and method (largely based on XML). A second set of observations were carried out on the proof of concept development and used to evaluate the framework.
- Methods: Two design methods are in use: (1) the method for extending the current architecture to better reflect the intricacies of UBIS and (2) the method deployed when moving from architecture to realised system. Method 1 is a pre-requisite for method 2. The iterative development and evaluation of the architecture and intelligent middleware software is possible starting with a number of business applications and services and finishing with a proof-of-concept software artefact.
- Instantiations: A developed software artefact, connecting applications to devices, will be evaluated in the context of the Mukherjee and Saha original framework. The middleware will utilise current business practices – namely Java and XML.

The aforementioned strategies permeated the research as a whole. The strategies themselves should not to be considered as process steps, but rather as means of

organizing the researchers' processes. All strategies were influential during every step of the study. In terms of the iterative cycle adopted to materialize the various research artefacts (i.e., constructs, models, method and instantiations), the steps that were followed are schematically outlined in Table 1.

Phases of research	**Individual steps**
Identify problem relevance	Conduct literature review Analyse industry applications and middleware Identify gap(s)
Framework design	Define scope of architectural framework Define underlying concepts and constructs Define middleware artefacts (input and output)
Framework evaluation	Apply framework to a realistic scenario Observe framework in action with proof of concept
Improve and re-evaluate framework	Identify limitations or areas of improvement Refine (re-design) architecture (iterate previous two steps)
Communicate and discuss research	Identify limitations and further potential benefits Define directions for future work Disseminate (e.g., present and publish findings)

Table 1: The adopted design research process
(based on guidelines by Hevner et al. (2004))

The developed software instantiation, described in the context of earlier stages, is detailed in this paper. The previous section highlighted an existing gap in the current literature. The following section will address the identified gap, confirming relevance of the problem investigated, both in the architectural and software artefacts. The resulting artefacts are now presented.

3.2 Intelligent Connector Architecture

Before developing a proof-of-concept, a number of business applications were analysed (four investment banking applications from each of the categories – Trading Systems, Risk Analysis Systems and Market Data Systems) and mapped to a number of devices. Each device selected is currently used in the business environment and (1)

their relationship to the application and (2) *transcoding rules* required when connecting each application part to the one of the new devices required explicit description. For example, a bond calculator that generates a matrix of bond analytics calculated in real-time, and when considering a mobile phone a number of relationships are apparent. The mobile phone display is able to display 3x4 of the Bond Matrix; the phones owner is interested in Russian Bonds; the phone's connection is able to handle 100 analytic records per second etc. Transcoding the output of the bond calculator to satisfy the various relationships (or constraints) requires a number of transcoding scripts to execute. A pipeline of transcoding scripts are constructed in order to satisfy the relationship between the mobile phone and bond calculator. The objective is to allow this to happen invisibly, without manual intervention or pre-configuration. Consequently, the initiating event, application-device relationships and transcoding scripts need to be in place for this to happen.

The architectural extension to provide this level of automatic integration is presented in Figure 3. This level of automation is critical if a middleware pipeline is to construct itself on-the-fly when reacting to some recognised event (e.g. a person passing an ambient screen) in a new and novel way. The additional relationship between events and applications and devices is a pre-requisite for any intelligent initiation. The same exercise was carried out on a further 3 applications and 4 ubiquitous devices. The result of the exercise and subsequent analysis is an extended architectural framework (see Figure 3).

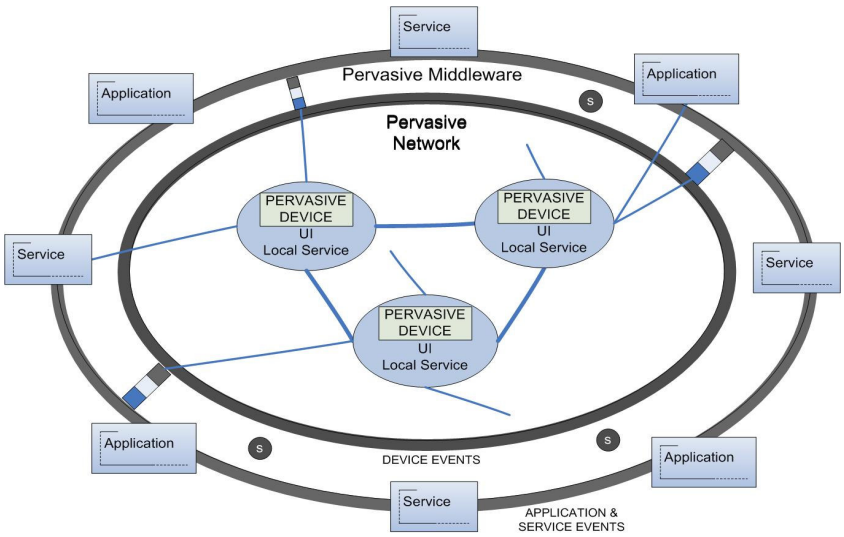

Figure 3. Extended UBIS Architecture

The first extension to the architecture is the addition of events to the middleware layer. Two event channels are added to enable: (1) Events to be consumed and generate the Services and Applications (e.g. informing that a specific bond analytic has changed) and (2) Events to be consumed and generated by the pervasive devices (e.g. Trader X is at location Y and is interested in Bond Z). It is this additional stage that allows the designer to explore both the capabilities and functionality of the devices and applications (inputs, outputs, requirement – when and where etc.). Consequently, the construction of linkages between the applications, services and devices are in response to specific events. The shared spaces ("S") are also included in the middleware layer as repositories of space based information and logic – embedded within a specific location. The space can be viewed as a *data cache* with integrated logic that is able to react to the cache and external events. The event message (an XML document) is read by the intelligent middleware who then identifies appropriate connections (interested parties). The second extension to the architecture is the connection middleware itself – comprising service adaptors, transcoders and device adapters. Initial analysis of the source systems (and their information provision) allows specific groupings of information (e.g. screen parts) to be identified as useful and described accordingly. Service adaptors extract this information in XML format (using the previous description as tags). A similar process is undertaken when describing particular devices, identifying and describing what they are able to render. The device adapters render XML on specific devices.

A connector pipeline is constructed from service adaptors (Java code that extracts XML^1 from each service), a number of transcoders that convert XML^1 into XML^2 ready for passing to the device for consumption via device adaptors. For example, in the Bond Calculator example above, the intelligent middleware would read events (ΔBondZ, TraderX@Foyer, TraderX=iPhone) and construct two pipelines to convert the bond analytics into an XML format that can then be transcoded for consumption by the device adaptors for the trader's iPhone and Ambient screen in the foyer.

A final artefact was a developed instantiation of the connection software and associate Web service. This made use of Java, XML and XPath. The system and device capabilities, event interests and input/output parameters were described in XML (see Figure 4). The intelligent connector service reacts to events by constructing pipelines.

The XML documents are searched using XPath for interests in a particular event and then a pipeline between the event producer and interested party is constructed. The transcoder software is selected (based on the two ends of the pipeline) and the link is made. Information from the event producer is rendered on devices that are relevant to the interested party.

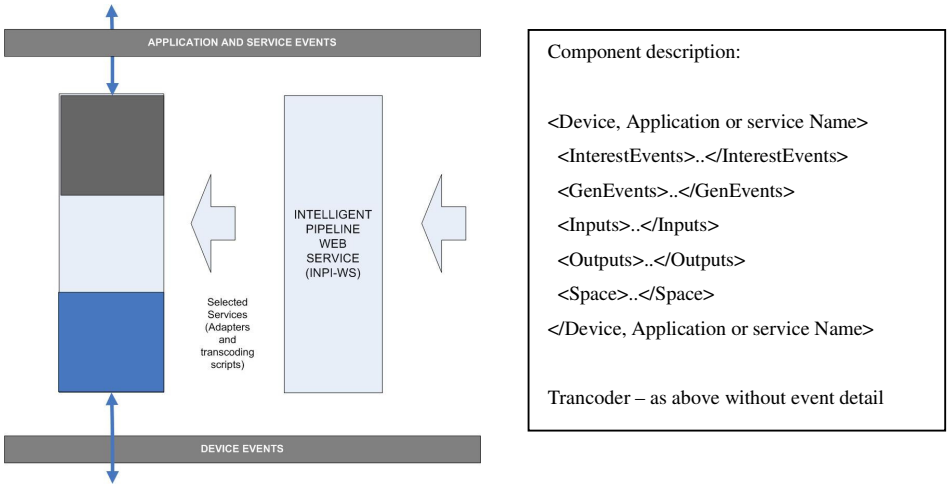

Figure 4. Connector Software and XML template

The transcoding scripts are selected to bridge the outputs of the sources artefact and the input of the recipient. The matching processing in this prototype is simplistic and not within the scope of the research aim of exploring and extending the architectural framework. The space tags in the XML are used to store contextual information about the device, application or service (e.g. proximity of a device etc.). The tags themselves reference shared spaces resident in the middleware layer (with universal resource identifiers URIs between the tags). In conclusion, the software system comprises a number of Java Web services that provide data conversion for particular devices, applications or services (adapters), XML conversion (transcoders), Pipelines construction (INPI-WS – using XPATH) and XML files containing component descriptions.

3.3 UBIS Architecture in use

Now that the architecture has been extended to better support the dynamic construction of ubiquitous business application integration (aiding *invisible* access), it is worth describing the process that emerged from the application analysis exercise. As more applications and devices were analysed and described an architecture design framework (articulating the population of Figure 3) resulted:

Process Step	**Details**
Indentify devices	Describe the characteristics of devices (rendering ability) that reside in the UBIS environment.
Identify applications and services	Describe the characteristics of applications and services (information provision) that reside in the UBIS environment.
Identify events and spaces	Events generated by applications, services and devices are classified (in relation to specific information). Interest in specific events can be added to the previously described devices, applications or services.
Select or design application, service and device adapters (and associated transcoding scripts)	Adapters provide an XML interface to the specific device, application or service. The inputs and outputs alignment determines the necessity to execute transcoding scripts. It should be noted that many applications provide an XML interface that will then only need transcoding.
Scenario Testing	A real-world use case can be used to test the information flow around the UBIS architecture. Further discrepancies in the description of architectural components can be removed or adjusted.
Realise physical architecture	Physical hardware components can be selected and software can be chosen or built.

Table 2: Architecture Design Process

The UBIS architectural framework now in place provides both the schematic and a supporting process. It is able to provide a basis for architecting intelligent business

system integrations that are able to better reflect human interests (in events and specific visualisations) and wider communities – prior to selecting and implementing specific technical components. It should also be noted that the framework is task independent, focusing instead on the environment (event description), the user lens (device description) and available information (device, application and service description).

4.0 UBIS Middleware Research Roadmap

The research presented in this paper is still at an early stage. Robust evaluation of the approach is required, systematically analysing further business applications. In terms of effectiveness the approach presented clearly improves on the current commercial approaches because: (1) the approach is not task centred and limited to specific information routings, (2) initial modelling allows the architecture to automatically adapt as new applications, devices or events appear and are recognised and (3) the approach is not platform or device dependant (utilising standards based XML and Web Service protocols). Existing Ubicomp research offers answers to some of these limitations, but not in the comprehensive manner that is able support wider business application architecture. The general nature of the architecture presented addresses many of the issues highlighted. Whilst undertaking the design it is clear that a number of potentially fruitful avenues exist:

- The modelling of application and device capabilities, relationships and events is limited by the syntactic and hierarchical nature of XML. Applying ontology languages (such as the Web Ontology language) of the Semantic Web may provide some additional benefits.
- The current approach relies heavily on a middleware layer that is able to process the transcoding pipeline. Limited scalability of the approach could warrant investigation into delegated processing with Clouds or Grids (traditional or mobile). The whole deployment of processing in a complex network of computers and devices is a topic in its own right.

5.0 Conclusion

In this paper the author presents a novel approach to ubiquitous information system (UBIS) architecture – including an intelligent middleware approach that reacts to an environment in which people and devices interact on-the-fly (not limited to responding to only specific requested tasks). An UBIS architectural framework is presented that extends work by Mukherjee and Saha to more effectively support ubiquitous information access in an enterprise setting - introducing ubiquitous computing architecture into the Information Systems discipline. A design research agenda is followed and includes both the extended architectural framework and a realised Web service instantiation (of intelligent middleware) that utilise Java, XML and XPath.

References

Bernstein, P.A. 1996, "Middleware: a model for distributed system services", *Communications of the ACM,* vol. 39, no. 2, pp. 86-98.

Coronato, A. & De Pietro, G. 2005, "Autonomic Pervasive Grids: A session Manager Service for Handling Mobile Users", *Self-organization and Autonomic Informatics (I),* .

Hevner, A., March, S., Park, J. & Ram, S. 2004, "Design Science in Information Systems Research", *MIS Quarterly,* vol. 28, no. 1.

March, S. & Smith, G. 1995, "Design and Natural Science Research on Information Technology", *Decision Support Systems,* vol. 15, pp. 251-266.

Nunamaker, J., Chen, M. & Purdin, T. 1991, "System Development in Information Systems Research", *Journal of Managemennt Information Systems,* vol. 7, no. 3, pp. 89-106.

Rouvoy, R., Eliassen, F., Floch, J., Hallsteinsen, S. & Stav, E. 2008, "Composing Components and Services Using a Planning-Based Adaptation Middleware", *Lecture Notes in Computer Science (Springer LNCS),* vol. 4954, pp. 52.

Saha, D. & Mukherjee, A. 2003, "Pervasive computing: a paradigm for the 21st century", *Computer,* vol. 36, no. 3, pp. 25-31.

Soldatos, J., Pandis, I., Stamatis, K., Polymenakos, L. & Crowley, J.L. 2007, "Agent based middleware infrastructure for autonomous context-aware ubiquitous computing services", *Computer Communications,* vol. 30, no. 3, pp. 577-591.

Trumler, W., Petzold, J., Bagci, F. & Ungerer, T. 2006, "AMUN: an autonomic middleware for the Smart Doorplate Project", *Personal and Ubiquitous Computing,* vol. 10, no. 1, pp. 7-11.

Weiser, M. 1993, "Hot topics-ubiquitous computing", *Computer,* vol. 26, no. 10, pp. 71-72.

Weiser, M. 1991, "The Computer for the 21th Century", *Scientific American,* vol. 265, no. 3, pp. 94-101.

Yao, S.S., Karim, F., Wang, Y. & Gupta, K.S. 2002, "Reconfigurable context-sensitive middleware for pervasive computing", *Pervasive Computing, IEEE,* vol. 1, no. 3, pp. 33-40.