‘Forget About It’: Signage at Preview Day 2003

Ian C. Millard, Gareth V. Hughes, Danius T. Michaelides, David E. Millard, m. c. schraefel, Mark K. Thompson, Mark J. Weal

Intelligence, Agents, Multimedia Group,
School of Electronics and Computer Science,
University of Southampton

{icm02r, gvh, dtm, dem, mc, mkt, mjw}@ecs.soton.ac.uk

In this report we present a brief introduction to the Signage Project, a multidisciplinary research collaboration exploring the physical-digital space, and detail our first experimental system which was deployed during a University ‘Preview Day’ for Sixth Form students. This system allowed the students to ‘tag’ posters and demonstrations on display that they found interesting during the day, enabling an adaptive, digital after-experience that provides access at the user’s convenience to further information on those topics which were of interest. We describe the motivation for our work, explain the Preview Day system, and discuss the future work stemming from this initial effort.

1 Introduction

The Signage project at the University of Southampton is an interdisciplinary effort within the IAM Group in the School of Electronics and Computer Science. Signage is designed to consider how the digital information space can augment human activity carried out in the physical space. Signage is uniquely situated in Southampton as a ‘bridge’ project between two EPSRC funded IRCs: AKT[1] in the semantic web space, and Equator[2] in the infrastructure/interaction space. Signage also has participant researchers from the fields of AI and e-Science.

Our focus for the project is on the Visitor. The concept of the visitor is a rich one for focusing interdisciplinary research and collaboration in the area of exploring the physical-digital space. The visitor focuses us on the multiple aspects of physical/digital requirements: that the visit takes place in a physical space; that a visit is a temporal event; that there are different classes of visits and visitors.

Each of these visit attributes has implications that must be solved concurrently, from which systems will be offered when to whom, to software architectures to support new information and services, to multiple devices accessing the system, to the services provided themselves. We must also consider the ways the visitors can engage with the information and services provided, and how their privacy is protected in any such exchange.

For instance in our case, that of a university research school, we have a range of regular visitors: UCAS students who visit at different times of the year to evaluate

[1] http://www.aktors.org/

[2] http://www.equator.ac.uk/

universities for their studies; industrial visitors who are looking for possible research cooperation; collaborators from different institutions, and of course, drop in visitors from elsewhere on the campus or local area, not specifically associated with the School. For each of these visitors, a visit system will optimally afford distinct information and services appropriate to the number and kind of visitors within the space/system. What we are exploring in Signage, through the visit scenarios are questions like the following:

- How can we augment the environment to support visitors' understanding of the site, appropriate to the kind of visit in which they are engaging?
- How can we support their information needs, from navigation to communication, in the physical context?
- How do we assist collaboration between internal and external information sources while the visitor is on site?
- How can we sustain persistence of their interactions with the site, so that participants can recover relevant parts of those events on demand?

In order to understand the role of pervasive computing in this context, we are considering questions from four related streams:

- **Infrastructure:** infrastructure to support multiple devices, services and information interacting concurrently
- **Semantic Web Technologies:** for delivery of information and services
- **Hypermedia:** for representation and delivery of information
- **Interaction:** consideration of affordances, constraints and requirements for service deployment.

In the following sections, we describe our first effort to explore the integration of these areas in order to support one class of visitor to our School. We describe the motivation for work, the system we developed and the future work stemming from this initial effort.

2 First Prototype

2.1 Motivation

We were challenged to provide a service for Preview Day visitors; these are once-a-year visits by Lower Sixth Form students who visit a variety of University Schools over the course of a day in order to gain a sense of areas of interest to them and their parents. The Computer Science School was tasked to provide a demonstration area where students could see examples of the different research projects currently being undertaken.

Based on a previous study we have carried out with UCAS visitors, we know that the school handbook, listing modules in various streams in the school, is not regarded as particularly engaging material. In addition, we have learnt from the academic staff

running the event that students are more interested in "gee-whiz" demonstrations rather than those which address deeper research questions.

2.2 'Forget About It'

We came up with a design we call "Forget About It" [3], allowing students to 'tag' posters/demonstrations for later retrieval. By supporting tagging, we hypothesised that we could help students engage with poster/demonstration presenters without having to worry about taking down notes about posters or people of interest. Students simply 'tagged' posters of interest (explained below). Later, they could visit a web site that presented a page of just the posters/demonstrations they had visited. Further information relating to these posters/demonstrations was presented, and augmented with information about the project, project URLs, and in particular, a list of undergraduate courses they might like to take based on the content of the demo. The listed courses were linked to the appropriate Calendar entries on the University web site. In this way, we hypothesised that we had provided a context for students to learn about some of the courses they might take, as well as having a concrete reason for taking them. In other words, we used the demos as an event/context for learning more about the research and the associated curriculum.

We supported the tagging process with iButtons[4], see Figure 1. As students arrived at the demonstration they were given an iButton and encouraged to optionally register their name and email address. As the iButton has a unique identifier, any information registered and all subsequent events with that iButton were associated with that id, hence still allowing interaction without having to specify any personal details.

Figure 1 – An iButton and key fob

Students were given the iButton to keep, as a persistent reminder of the event. They were also given a handout explaining where the URL for their web page would be and how to get to their customised page.

The following sections of this paper present a discussion of the infrastructure, storage, devices and systems issues with deployment of this Signage visit demo. We conclude with some lessons learned and next possible steps for Signage as outcomes of the process of deploying this demonstrator.

3 Infrastructure

The infrastructure on which the Preview Day experience was hosted consists of a distributed client/services architecture as described in Figure 2, with multiple processes interacting via the Elvin content based messaging system[5].

[3] http://signage.ecs.soton.ac.uk/visit03.html

[4] http://www.ibutton.com/

[5] http://elvin.dstc.edu.au/

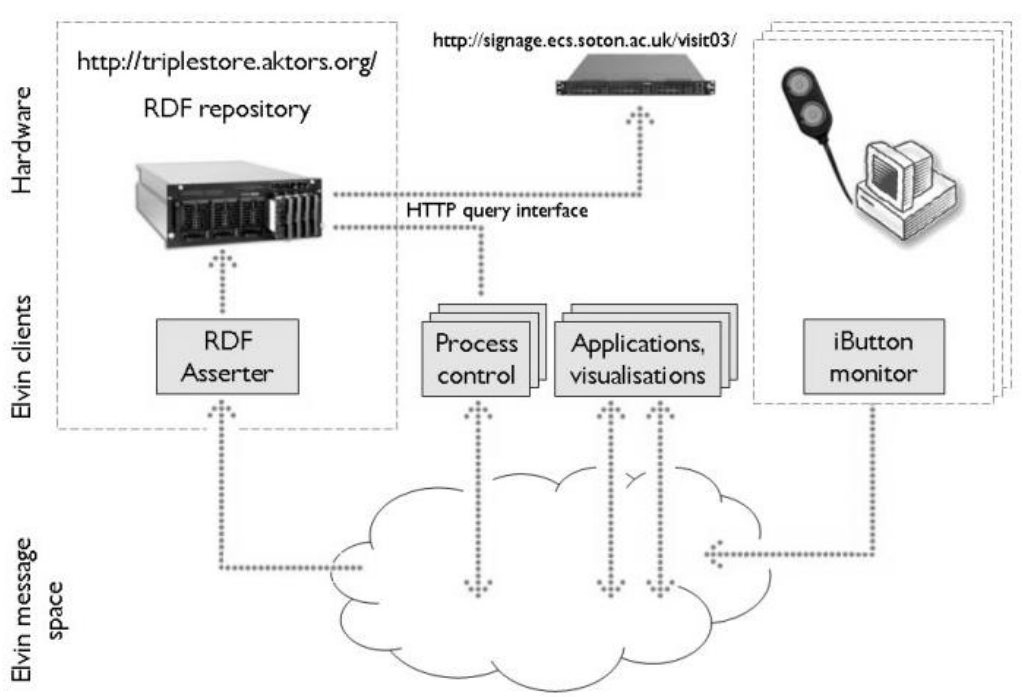

Figure 2 – Conceptual 'high level' architecture diagram

Each of these asynchronous services connects to a central Elvin daemon, which allows them to publish and/or subscribe to particular Elvin notifications. These notifications consist of a group of key-value pairs, and they are routed by the daemon to all clients that have registered subscriptions for those types of message. These interactions are depicted in Figure 3, and the messages passed are listed in detail in Appendix A.

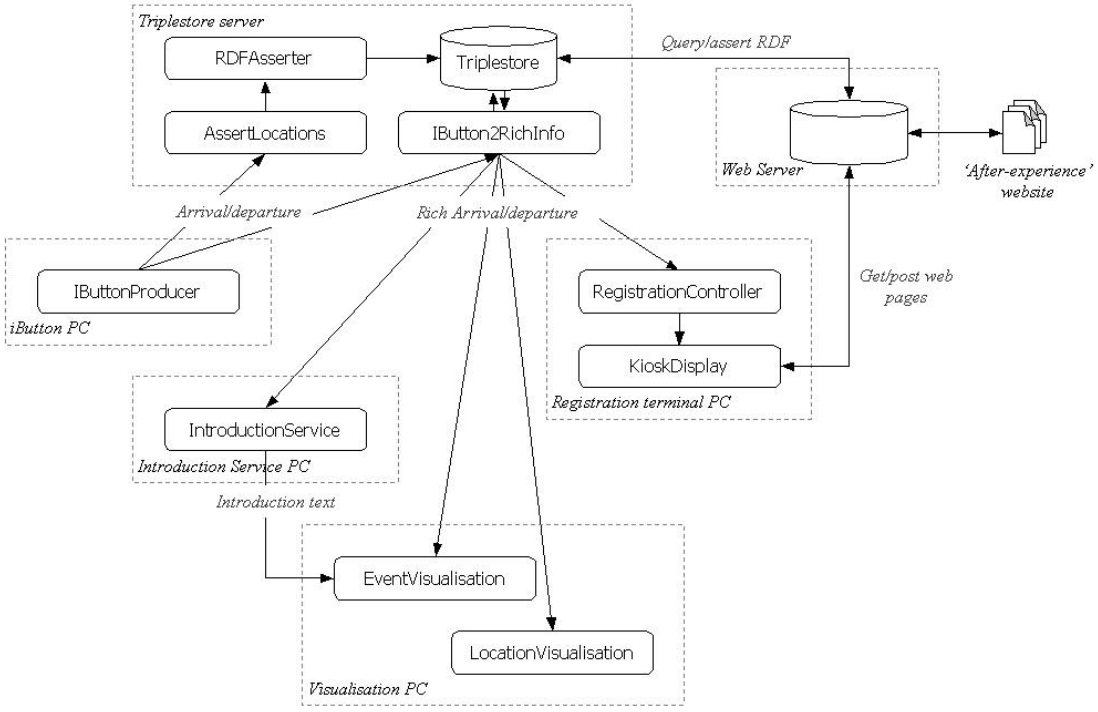

Figure 3 – Interactions of Elvin clients in the Preview Day infrastructure

The data storage mechanism underpinning Preview Day experience is the AKT Triplestore[6], a specialised RDF repository and inference engine. Java utilities were created to provide access to the Triplestore's HTTP RDQL interface, allowing queries to be made and their results returned. In addition, the `RDFAsserter` Elvin client ran on the Triplestore server to allow the assertion of RDF in real-time when an 'assert' notification was received. Each time such a notification arrived, a file was opened on the server and RDF data written to it. Subsequently, the Triplestore import program was executed such that the data in the new file was incorporated into the repository.

3.1 iButton docks

iButtons are a class of 1-Wire[7] device that both communicate and receive power over a serial bus. These busses can be driven by an interface connecting to a standard PC serial port. In the signage implementation, a separate 1-Wire bus was routed for each docking station, isolating each station from bus shorts, but also providing a mechanism by which location could be cued: each bus, and therefore each PC serial port, represented a discrete location.

For the Preview Day experience, one PC served as a 1-Wire bus master for all of the docking stations at posters and demonstrations, and also for some of the visualisation stations. In order to demonstrate that different hosts could participate as message producers of equal stature, 1-Wire bus masters were deployed on each of the Registration Terminals. For each 1-Wire bus master, a Java thread would poll the bus for device arrival or departure events. Upon observing such an event, an Elvin notification was generated comprising the identifier of the device observed, the nature of the event (arrival or departure), and the physical location keyed on the bus on which the event was observed.

Where iButton docks were placed alongside hosts that were running applications for visualisation, registration and demonstration, the coupling between message source (e.g. an iButton being docked in its receptacle) and sink was distributed, an artefact of the nature of the iButton device.

The physical characteristics of certain iButton docks meant that under certain conditions, the arrival or departure of a device would electrically short the bus, resulting in the process monitoring the bus wrongly reporting that there were no devices present: any other device on the bus would be reported as having departed for the duration of the short, and then re-arriving once the short resolved. This is a point of confusion for some of the applications that registered interest in device events.

There are other techniques to providing tree-like 1-Wire bus structures that are resilient to bus shorting, and addressable to the extent that different sections of the bus could represent different physical locations. Whilst interesting to research for larger scale, more permanent deployments, the approach of a single 'dock station master' PC with 16-way serial port multiplexer where each port masters its own 1-Wire bus sufficed for the Preview Day experience.

---

[6] http://triplestore.aktors.org/

[7] http://www.maxim-ic.com/1-Wire.cfm

4 Preview Day Applications

4.1 Registration Terminal

When students arrived at the Preview Day, they were given an iButton and requested to enter their details at a registration terminal. This terminal combined an iButton docking point and a `KioskDisplay` – a simple Java web browser that could also be instructed to display specific pages by means of publishing an Elvin notification. The `RegistrationController` process detected iButton arrival messages, and advanced the display from a welcome screen to an input form, allowing the user to enter their name, school and optionally their email address. On submission of the form, the web server asserted this information as RDF in the Triplestore, associated against the unique iButton identifier, and displayed a confirmation/thank you screen at the terminal. If at any point during the registration process the iButton was removed, then the `RegistrationController` cancelled the registration and returned the screen to the welcome state.

When iButton arrival or departure notifications were received by the `IButton2RichInfo` client, the Triplestore was queried to return any information associated with the ID of the iButton and the location from which it had occurred. This was then republished as another Elvin notification for use by other applications, such as the `IntroductionService` and `EventVisualisation`.

In addition to the mogrification of arrival and departure notifications, the `AssertLocations` client stored a time-stamped record of each arrival by asserting an RDF resource in the Triplestore detailing both the iButton ID and the location of the visit.

4.2 Visualisations

To provide some exposure of the underlying infrastructure to the visiting students, two visualisations were constructed and displayed in the room on a projected screen, in conjunction with an explanatory poster. The `EventVisualisation` gave a real-time scrolling ‘log’ of the events received, detailing arrivals, departures and other events within the room. In addition, the `LocationVisualisation` provided a dynamic map of the room, emitting circular ‘pings’ representative of the location of the source of each Elvin notification. Although crude, studying this display gave a rough representation of which demonstrations were popular, as they could be seen to emit pings more frequently than those which were less popular. A screenshot of these visualisations can be seen in Figure 4, below.

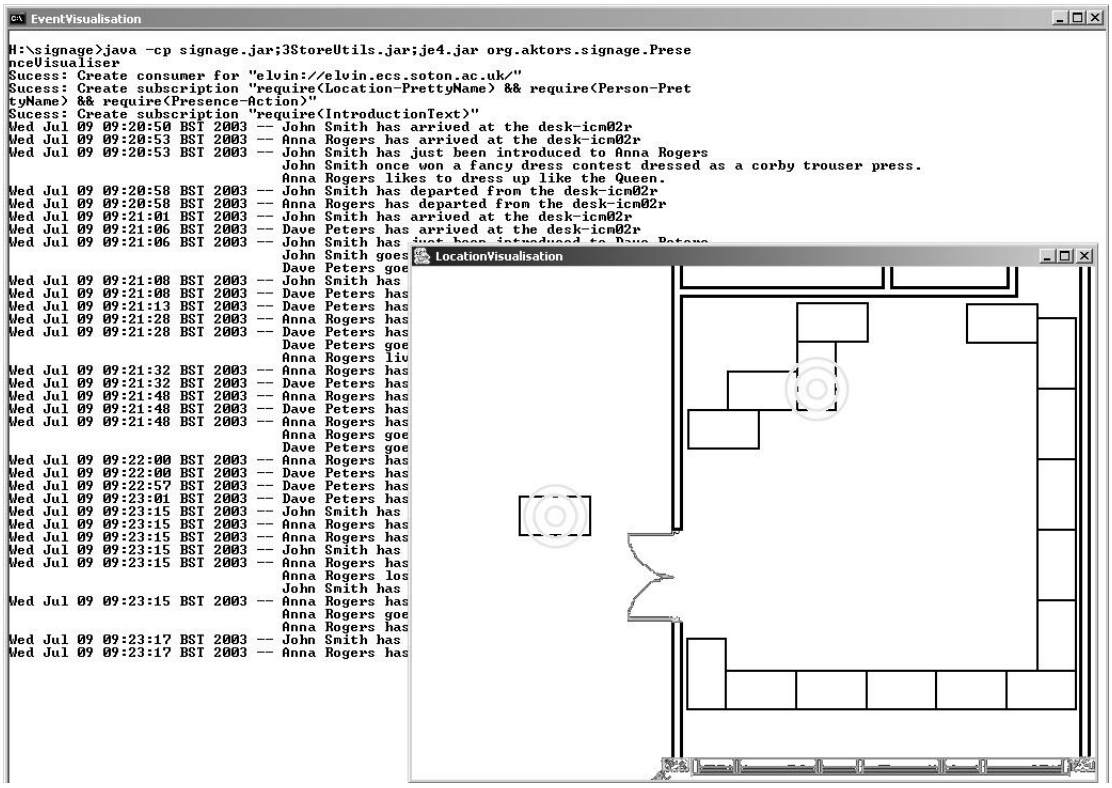

Figure 4 – Visualisations of iButton dockings

In addition to the user-facing visualisations, an Elvin sub-system health monitor visualisation tool was deployed, as shown in Figure 5, which assisted in the 'comfort' level of those researchers responsible for the liveness of the system. The nature of interaction using Elvin means that no individual component needs to understand all the various notification elements, their name, purpose or type. However, because the Preview Day was a closed system in that all of the code was developed by a small team of researchers, the meaning of the various components of the notifications (such as which attributes appear together, what the purpose of each attribute and notification was) was known. Leveraging this knowledge, a set of subscription expressions was identified that captured the entire spectrum of possible notifications, which was then used to seed the eMatrix Elvin traffic visualisation tool[8]. Each different type of notification is associated with a character, which scrolls down the screen as such events occur. This system was utilised to resolve the difficulty of attempting to watch a raw log of the Elvin notifications in real-time, as there are so many events occurring that they cannot be seen before they have scrolled off the screen. Watching the comparatively slow 'trickle' of characters moving down the screen allowed the researchers to monitor the events occurring at a more abstracted level, with any lack of a particular type of event indicating a possible problem or failure.

[8] http://elvin.dstc.edu.au/projects/ematrix/

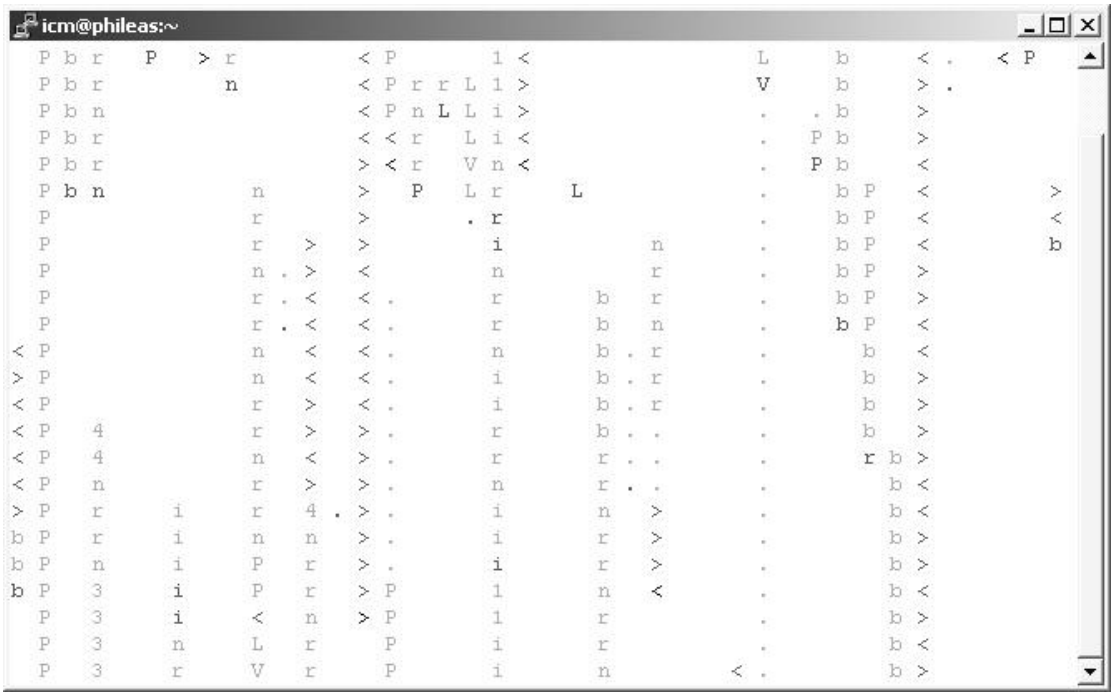

Figure 5 – eMatrix Elvin traffic visualisation tool

Whilst not particularly usable by non-specialists, the visualisation enabled a simple mechanism by which the overall health of the various components participant in the experience.

4.3 Introduction Service

As part of the trials, two iButtons were associated to an introduction service. This was a Director movie[9] that used the Xtrae4 plug-in[10] to subscribe to Elvin notifications. A screen shot of the introduction service is shown in Figure 6.

There were a number of objectives to the construction of the `IntroductionService`.

- To provide an entertaining demo for the visitors day.
- To look at the distribution of devices connected by a notification message format. The interface was separated from the iButton code. Ideally, the display could have been separated from the application as well, with a fixed display that simply receives notifications and displays the identified text. There wasn't time for that for this trial however.
- To begin to look at automatic configuration or indeed service discovery. The `IntroductionService` receives notifications to tell it the location identifier of the iButton dock for which it has to respond. This form of simple configuration by notification could form the basis of more automatic configuration strategies.

[9] http://www.macromedia.com/software/director/

[10] http://elvin.dstc.edu.au/projects/xtrae4/

- To look at how the Triplestore could be used to augment information flowing through the infrastructure. Initial ideas centred on the `IntroductionService` querying the Triplestore by asynchronous notifications. In the end, the `IButton2RichInfo` client intercepted the original iButton notifications and added the additional information from the Triplestore. Variations on these strategies could be tried and compared.

The architecture for this visualisation is fairly straight forward. The `IntroductionService` process listens for the rich arrival/departure notifications, which contain data extracted from the Triplestore as and when people arrive or depart from an iButton dock. When two people are docked at the Introduction Terminal, it generates an introduction and sends a notification to the rest of the system to indicate that it has introduced two people. After ten seconds the introduction service returns to its main screen and waits for two new people to dock with the system.

Once two people have docked and the notifications are successfully decoded, an introduction is produced on the interface. This is created from introduction information loaded from the XML data file [Appendix B] and combined with the details from the Triplestore concerning the two people to be introduced.

There are two types of template in the script. The first is a brief piece of introduction text, allowing greetings to be made in the format "`[name1] intro_text [name2]`". The script file contains these snippets between `<introducing>` tags.

The second type of template is an anecdote about the person. The anecdotes are intended to be non-gender specific and ludicrous to the point of non-believability. It was not our intention to offend anybody with these anecdotes. The format of the anecdote is always "`[name] anecdote`", with the anecdotes being selected randomly from those in the script file, encapsulated within `<anecdote>` tags.

There is a random chance that instead of an anecdote, if data is available, the school of the person will be mentioned in the form "`[name] goes to [school]`", in a pitiful attempt to at least occasionally ground the introduction in reality.

So once there is a `person1` and `person2` docked at the iButton the screen will introduce them as shown here in Figure 6.

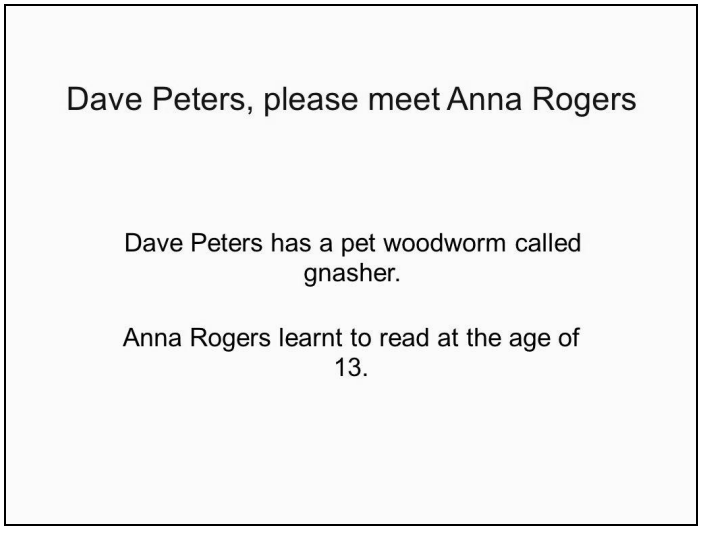

Figure 6 – An example introduction from the `IntroductionService`

It can be seen that all of the introductions appear to be different, even though each follows the described format –

```
[name1] introduction_text [name2]
        [name1] anecdote
        [name2] anecdote
```

The `IntroductionService` listens for, and is configured by an Elvin notification in the form

```
iButtonIntro: "reset"
iButtonName: <iButton_location_id>
```

Receiving this notification will tell the `IntroductionService` that there are no buttons currently docked. This became necessary as due to shorting problems with some versions of the iButton docks, the `IntroductionService` could get into a confused state where it believed a button was still docked when it had been removed.

The `iButtonName` parameter is used to configure the `IntroductionService` to monitor a specific iButton dock, from which it listens for the arrivals and departures of people that are required in order to generate introductions.

In addition, when it introduces two people the `IntroductionService` gives details on the meeting of the two people, by sending an Introduction notification in the format

```
iButtonIntroduction: "introduced"
Person1: [name1]
Person2: [name2]
IntroductionText: [name1] [introduction_text] [name2]
Person1Text: [name1] [anecdote1]
Person2Text: [name2] [anecdote2]
```

5 Post-visit experience

After the visit the students were allowed to keep their iButtons and were given a web address where they could retrieve a personalised brochure for the school.

When they visited the site they were presented with a simple web form that asked for either their email address or the unique id written on their iButton. These were then passed to a Servlet that generated an adaptive brochure page, as shown in Figure 7.

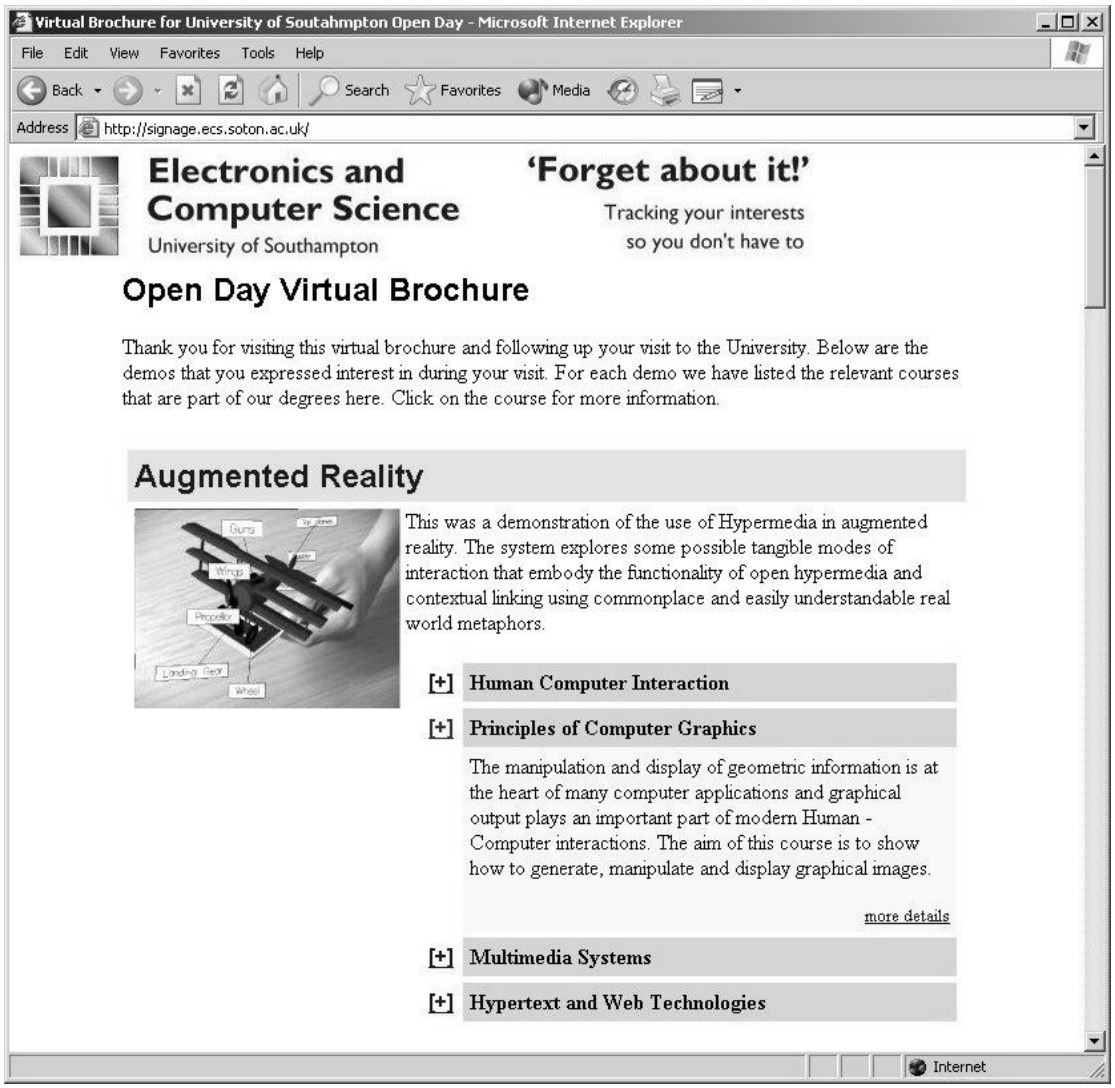

Figure 7 – An example 'after-experience' web page

The Servlet first queries the Triplestore to retrieve a list of all the demonstrations and poster displays that the student visited and selected on the day. It then builds those into a user profile. The brochure itself is stored as a FOHM structure [2] in Auld Linky contextual link server [1]. Each part of the structure has metadata attached to say in which contexts it is visible. When the Servlet queries Linky it does so in the context of the selected demos. This causes parts of the returned brochure structure (those demos that the student was not interested in) to be removed. The Servlet then crawls over the remaining structure and renders it into DHTML.

The top level of the brochure structure is a list of demos but each demo entry can itself be made up of other sub-structures. Thus the brochure is more of a tree than a list.

The sub-structures can be typed and the Servlet renderer uses the type of each structure to decide how to render it. For example, lists are rendered as a sequence. In addition to lists we also use LoD (Level of Detail) structures. These represent a set of content describing a single concept, but in increasing levels of detail. When it encounters a LoD the renderer uses the lowest level of detail but places a [+] next to it

and inserts DHTML such that the [+] expands into the next piece of content in the LoD structure.

Figure 7 shows how we used this to arrange the brochure. Each demo has a title, picture and small paragraph of explanation. These are then followed by LoD structures describing each school course that is applicable to that demo. If the user decides to expand a course name then a description is shown along with a link to the home page for that course.

In this first version of the post-visit Servlet the user context is used only to select which demos will be described. However, the context could also be used to alter the descriptions of the demos and the courses. For example, the descriptions could describe relationships with other demos if it was known that those had also been selected and course descriptions could include specific information on how selected demos were applicable to them.

6 Event playback

The definition of an agreed schema for all of the notifications distributed throughout the system [Appendix A] meant that an event capture process could be set up to record all device and process interactions. Each new event is marked by three dashes and the ISO8601 standardised time of the notification, with the body of the notification represented as a list of attribute-value pairs. For example,

```
--- 2003-06-18 10:57:45.417312+0100 ---
Presence-Location:
"http://www.ecs.soton.ac.uk/location/#B59.1257.registration"
Presence-Action: "Arrived"
Presence-Identifier: "urn:signage/ibutton#860000000B23080C"
--- 2003-06-18 10:57:45.814310+0100 ---
Kiosk-Identifier: "RegistratonTerminal1"
Update-URL:
"http://signage.ecs.soton.ac.uk/info.php?iButton=860000000B23080C"
```

In this fragment there are two events, the first of which originates from the `IButtonProducer` process, polling the 1-Wire bus pertaining to the iButton dock at the first registration station, and the second is from the `RegistrationController` process, instructing the `KioskDisplay` to navigate to the next page. As discussed previously, each of these notifications would be routed by the Elvin daemon to processes that had registered an interest in them.

Having a log complete with individual timestamps enables the replay of any part of the experience in order to analyse what processes worked well, visualise which stations were visited the most, or test new processes without actually requiring all of the 1-Wire hardware to be set up and running. This record-replay-reflect cycle not only assists in gleaning valuable research data from the experience beyond more typical user interaction studies, but it also readily enables visual demos of the system to visitors that couldn't make the 'installation demo', or perhaps wouldn't receive as much benefit from a videoed recording.

7 Discussion

7.1 Re-visiting

Students seemed enthusiastic about the process – comments like “cool” were heard at the registration desk. Comments like “this sucks,” happily, were not. To those students who registered an email address, we mailed out a note a day after the event to remind them of the URL to the web site and how they could access their page: either by entering the id number on their iButton or by entering the email address they gave us. In the two weeks after the event, of 85 students who picked up iButtons, 30 returned to the site, logged in and loaded their custom web page. This one-third plus return to the system is a positive indicator. It suggests that students were interested enough in the system or the associated material to go through the trouble of logging in and looking around the page. What we need to investigate further are the attributes responsible for generating a return to the information: the interaction with the iButtons, the notion of a customized page based on those deliberate interactions, interest in a specific demonstration, the convenience of having subjects of interest gathered in one place, and so on.

7.2 System events

Beyond the interaction, we have also been able to test our infrastructure for handling multiple events (some 7000 in 2 hours) and for our Triplestore technology to handle that many assertions into it in real time.

7.3 Registration, Queues, Bottlenecks and Ergonomics

The Preview Day experience was planned as three, one hour slots during which time the visiting students would arrive, register, and engage with poster presenters around the room. The School indicated to us that it would be impossible to know in advance the numbers that would arrive. We were given estimates of approximately 20 to 30 students per session. With little else to go on we ordered 100 standard identity iButtons, 100 plastics fobs for iButtons and printed 100 colour instruction sheets to hand to each student on entry. The primary purpose of the sheet of paper was to carry the URL to the post-experience web page. There was a labour cost in fixing iButton to fob, printing the sheets and sticking each fob to the sheets. The reality was that 45 people arrived for the first session, slightly more for the second and slightly less for the third. We had set up one registration terminal on a normal height table just outside the door to the seminar room. A quick speech to each group by one of the team to the visitors before they entered the room explained what we were doing and that each person should taken an iButton, register and go and explore. It quickly became apparent that the registration process would become a bottleneck.

The time for each person to register was two to three minutes. The majority of students filled in all three fields (name, email and school) even though we explained they were optional. The registration terminal was on a normal height table, so users had to put their things down, reorganise their bags and familiarise themselves with a keyboard and screen, then type. Most were fast typists and nearly all had email addresses. It should be noted that pressing the Return key did not activate the Submit button. The mouse needed to be used or a user had to tab to the Submit button and

then press Return. This was a cause of slight confusion and delay in some cases. The cumulative effect was a queue that blocked the foyer of the School for approximately ten minutes in advance of the first session.

We quickly changed plan and told people to take an iButton, go straight in and register later as and when the terminal became available. Acknowledging a user's interest for a poster/demo required only the iButton to be docked, and due to the nature of the way in which the data was stored in the Triplestore using the iButton ID their registration and visit data automatically tied up when they subsequently came to register. However, the applications requiring registered information, such as the `IntroductionService`, could not be used until that information was registered with the system.

The "register whenever" approach helped reduce congestion but still led to a queue for the terminal. For the second session we set up a second registration terminal using a laptop. This was easy to do as the messages it sent were no different to the other machine and the rest of the infrastructure was unaffected. This was a definite plus point to using asynchronous messaging. Unfortunately the laptop keyboard was a little harder to use than a normal keyboard but it had the desired impact on the queue of people. The recommendation is that laptops should not be used as data entry terminals for the public. A second recommendation is that a public terminal to be used by a standing person should be placed on a bar-height location, not a table. There also needs to be a certain amount of room around each terminal to avoid choke points.

As we used up our entire iButton stock by the end of the second session we packed away the system before the start of the third session while the demonstrations and posters were left up. If the system was deployed for other types of events, such as a conference, there is the advantage of having pre-registration. Then an iButton, or other identifying device such as an RFID tag, can be pre-registered to users and handed out on arrival with the rest of the conference material. There is still an additional workload with this as well as the requisite for technical support to be on hand to support the people running a conference registration desk. There are also going to be a non-trivial number of people who arrive at a conference who are not fully registered, then the registration process must available on the day.

7.4 Infrastructure issues

The key challenge to the infrastructure posed by this visit day is to provide a suitable layer of abstraction so that applications and experiences can be built and deployed as efficiently as possible. Whilst the visit day was a temporary installation, we envisage permanent deployment of hardware around the building on which many applications can be run, both permanently and for specific events.

The use of the content-based routing means that any number of applications or services can connect to the messaging system and be instantly aware of any events occurring. However, this assumes a common understanding of what the key-value pairs in the messages mean. We envisage an Elvin Dictionary that describes possible messages and their contents. For the Forget About It application the dictionary would be quite small, but for more complex applications, the dictionary could be quite large. Ontological descriptions of message contents would allow translation to facilitate communication between components.

In addition, descriptions of messages and processes that produce them would be an aid to the developer during building and debugging. Ultimately, the construction of an experience could be as simple as dragging and dropping suitable components together.

The use of the Elvin messaging system provides one possible cut through the infrastructure space. Elvin does not provide any form of state or persistence; instead we relied on specific listening processes to create triples for storage in the Triplestore. Other possible infrastructures we may choose to use/evaluate (perhaps in conjunction with Elvin):

- tuple-spaces with triggers
- web-services with service composition and workflow
- modelling the world (eg a semantic mud)

7.5 Use of the Triplestore

The manner in which the Triplestore was used during the Preview Day experience was somewhat outside of that anticipated as it was designed. As part of the work undertaken in the AKT IRC, the Triplestore was conceived to be a high volume store of relatively static RDF data, which would be updated in large batches from data acquired from over-night runs of content harvesting applications. Quite conversely, the Preview Day experience required the real-time assertion of large numbers of very small RDF files, and subsequently some problems were encountered.

First, there was no external interface to the Triplestore for asserting data. The `RDFAsserter` client was written to provide this functionality, however although it was sufficiently operational, the repeated opening and writing of files and execution of the import utility was far from efficient.

Second, after the students had left, a simple script was used to combine the (several thousand) individual RDF files asserted during the day into two RDF files describing person data and visit information, for the purpose of ease content management and to improve database efficiency.

Subsequent discussions have determined that a native interface should be implemented as part of the Triplestore software to allow for the import of small RDF models in real-time.

8 Future work

In this report, we have outlined the rationalisation for the Signage Project. We have described in detail the first prototype system, ‘Forget About It’. The system integrates the layers of adaptive infrastructure, semantic web resources, hypermedia information representations and interaction design. Through this integration, we have been able to begin to explore where the digital meets the physical for visitors to semi-public spaces. The purpose of the Visit Day deployment was to allow us to carry out a loosely formative study of this research space. By building, deploying and watching the system in use, we have begun to gain an understanding of the questions we will need to address to support visitors.

Future research in the area of adaptive infrastructure for the visit scenario will need to consider, for instance, the description, discovery and configuration of services and devices.

In the semantic web space, we need to consider the engineering aspects of the Triplestore design for real-time data assertion. This is an initially unanticipated requirement that has come from this first study.

In the area of interaction research, the questions are legion, and range from both sides of the interaction with the physical. We are looking at adding features to the tagging interaction to support short annotations on site. We want to better understand how to add appropriate affordances to the resulting web space so that the participant can make better and ongoing use of the resources associated with the artefact they selected as of interest.

We are also interested to see how we might be able to tie such visit experiences to persistent information flows both on site and off. For instance, a visitor may indicate a set of artefacts that have a semantically discoverable relationship. That relationship might be usable for connecting the visitor to associated resources, such as researchers in the area, talks to be given, or other events throughout the year.

While this has only been an initial probe into the visit space, it has revealed the space as a rich one, and a powerful scenario for focusing interdisciplinary research questions for physical-digital systems and interactions.

9 References

[1] Danius T. Michaelides, David E. Millard, Mark J. Weal, David C. De Roure *Auld Leaky: A Contextual Open Hypermedia Link Server* in Proceedings of the 7th Workshop on Open Hypermedia Systems, ACM Hypertext 2001 Conference. Post-Workshop Proceedings to be published by Springer Verlag in the Lecture Notes in Computer Science (LNCS) Series, August, 2001, Arhus, Denmark.

[2] David E. Millard, Luc Moreau, Hugh C. Davis, Siegfried Reich. *FOHM: A fundamental open hypertext model for investigating interoperability between hypertext domains*. In proceedings of the 2000 ACM Conference on Hypertext, May 30 - June 3, 2000, San Antonio, TX

10 Appendices

Appendix A Elvin notification dictionary

Figure 8 represents the sources and sinks of Elvin notifications within the Preview Day scenario. Examples of each are listed below.

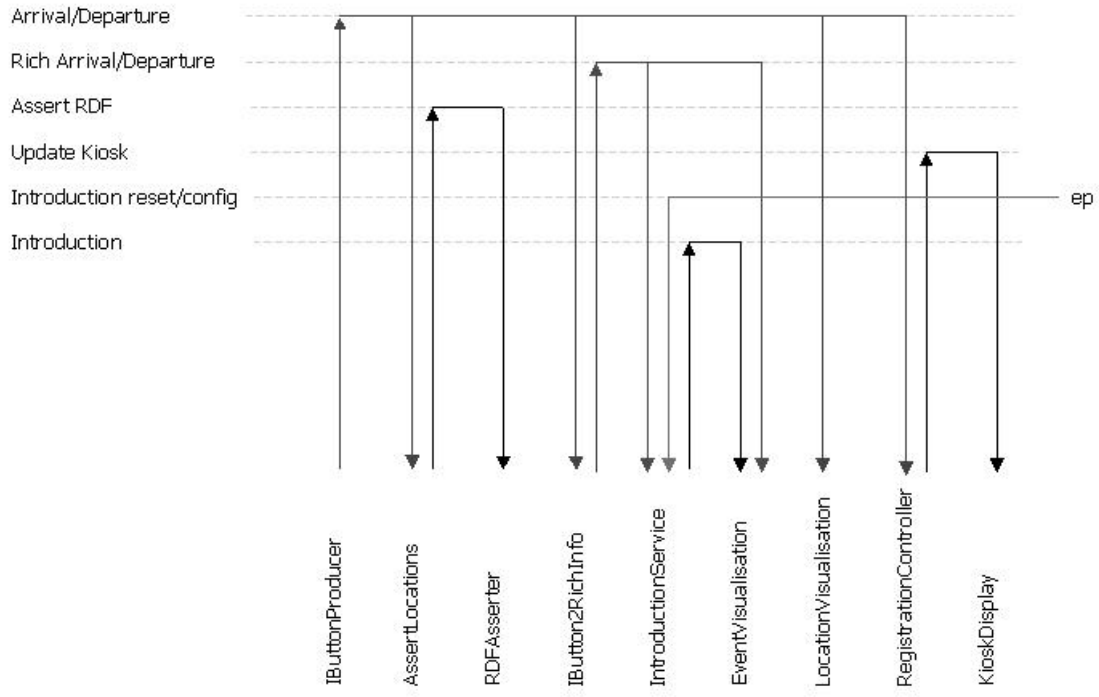

Figure 8 –Producers and Consumers of Elvin noticications

Arrival/Departure

```
--- 2003-06-18T11:57:08.000000+0100 ---
Presence-Location: "http://www.ecs.soton.ac.uk/location/#B59.1257.reg2"
Presence-Identifier: "urn:signage/ibutton#5D0000091A7C2101"
Presence-Action: "Arrived"
```

Rich Arrival/Departure

```
--- 2003-06-18T11:57:08.000000+0100 ---
Presence-Location: "http://www.ecs.soton.ac.uk/location/#B59.1257.reg2"
Person-iButtonID: "5D0000091A7C2101"
Presence-Action: "Arrived"
Person-PrettyName: "Joe Bloggs"
Person-SchoolName: "Southampton High School"
Person-EmailAddr: "joe@bloggs.com"
Person-ResourceID: "http://www.aktors.org/ontology/signage#person-5D0000091A7C2101"
Location-PrettyName: "Registration Terminal 2"
X-Mogrified-By: "IButton2PersonInfo (triplestore.aktors.org/152.78.64.96)"
```

Assert RDF

```
--- 2003-06-18T11:57:08.000000+0100 ---
Assert-RDF-In-File:  "location-of-person-5D0000091A7C2101.rdf"
Assert-RDF-Content:  "<?xml version='1.0' encoding='UTF-8'?>
                      <!DOCTYPE rdf:RDF [
                        <!ENTITY rdf 'http://www.w3.org/1999/02/22-rdf-syntax-ns#'>
                        <!ENTITY akt 'http://www.aktors.org/ontology/portal#'>
                        <!ENTITY ecsloc 'http://www.ecs.soton.ac.uk/location/#'>
                        <!ENTITY signage 'http://www.aktors.org/ontology/signage#'>
                      ]>

                      <rdf:RDF
                        xmlns:akt='&akt;'
                        xmlns:rdf='&rdf;'>

                        <akt:Person rdf:about='&signage;person-5D0000091A7C2101'>
                          <akt:is-at-location rdf:resource='&ecsloc;B59.1257.reg2' />
                        </akt:Person>
                      </rdf:RDF>"
```

Update Kiosk

```
--- 2003-06-18T11:57:08.000000+0100 ---
Kiosk-Identifier: "RegistrationTerminal2"
Update-URL: "http://signage.ecs.soton.ac.uk/get-details.php?iButton=5D0000091A7C2101"
```

Introduction reset/config

```
--- 2003-06-18T09:45:12.000000+0100 ---
iButtonIntro: "reset"
iButtonName: "http://www.ecs.soton.ac.uk/location/#B59.1257.introduction"
```

Introduction

```
--- 2003-06-18T11:23:30.000000+0100 ---
iButtonIntroduction: "introduced"
Person1: "Nick Jennings"
Person1Text: "Nick Jennings can't pronounce the word flange properly."
Person2: "Nigel Shadbolt"
Person2Text: "Nigel Shadbolt only eats turnips on Tuesdays."
IntroductionText: "Nick Jennings has just been introduced to Nigel Shadbolt"
```

Appendix B IntroductionService configuration file

```
<?xml version="1.0" encoding="UTF-8"?>
<script>
  <introducing>allow me to introduce</introducing>
  <introducing>let me introduce</introducing>
  <introducing>I'm pleased to introduce</introducing>
  <introducing>please meet</introducing>
  <anecdote>was born on the fourth of July.</anecdote>
  <anecdote>only eats turnips on Tuesdays.</anecdote>
  <anecdote>once played mahjong with the Sultan of Brunei.</anecdote>
  <anecdote>lost a pet hamster to Barbara Windsor in a poker game.</anecdote>
  <anecdote>hates being introduced to people.</anecdote>
  <anecdote>appeared on Question Time.</anecdote>
  <anecdote>is allergic to sunglasses.</anecdote>
  <anecdote>lives in a four story one bedroom flat.</anecdote>
  <anecdote>can't pronounce the word flange properly.</anecdote>
  <anecdote>likes to dress up like the Queen.</anecdote>
  <anecdote>once had J.K.Rowling as a pen pal.</anecdote>
  <anecdote>used to be addicted to jelly babies.</anecdote>
  <anecdote>learnt to read at the age of 13.</anecdote>
  <anecdote>can only use digital clocks.</anecdote>
  <anecdote>likes Barry Manilow.</anecdote>
  <anecdote>plays the tuba in a garage band.</anecdote>
  <anecdote>has a pet woodworm called gnasher.</anecdote>
</script>
```