Experimental Platform for Mobile Information Systems

Rudi Belotti, Corsin Decurtins, Moira C. Norrie, Beat Signer, Ljiljana Vukelja
Institute for Information Systems
ETH Zurich
CH-8092 Zurich, Switzerland
{belotti,decurtins,norrie,signer,lvukelja}@inf.ethz.ch

ABSTRACT

Interaction design is a major issue for mobile information systems in terms of not only the choice of input-output channels and presentation of information, but also the application of context-awareness. To support experimentation with these factors, we have developed a platform that supports the rapid prototyping of multi-channel, multi-modal, context-aware applications. The paper presents the main components of the platform and describes how it was used to develop a tourist information system for an international arts festival where interaction was based on a combination of speech input-output and interactive paper.

Categories and Subject Descriptors

H.4.3 [**Information Systems Applications**]: Communications Applications—*information browsers*; H.5.2 [**Information Interfaces and Presentation**]: User Interfaces—*prototyping*; H.5.4 [**Information Interfaces and Presentation**]: Hypertext/Hypermedia—*architectures, navigation, user issues*

General Terms

Design, Experimentation

Keywords

Mobile information system, interactive paper, web publishing, rapid prototyping, voice interface, tourist guide

1. INTRODUCTION

Mobile information systems require platforms that not only deal with the challenges of data distribution and dynamic networking, but also entirely new forms of interaction and information delivery. Ideally, users should receive the right information at the right time and place, and in a way that restricts neither their mobility nor their interaction with other people and the environment. This means that devices must be either wearable or very portable and easily placed in pockets when not in use.

If we consider the domain of tourism which has been a focus of several research projects in mobile information systems [16], many of these requirements are not met in terms of the devices and forms of interaction provided. Tourism is generally a social activity and part of the enjoyment is planning activities together with family and friends. The screens of PDAs that are often used in mobile applications are small and difficult to read outdoors, especially by more than one person at a time. Further, the small screen size limits the amount of information that can be viewed at one time and does not support the actions of comparing and combining information which is often what tourists want to do [4]. Some researchers have therefore experimented with tablet PCs to provide better functionality [5], but clearly these further restrict mobility as they are much heavier than PDAs and require the use of both hands. Various other options, such as head-mounted displays, digitally augmented paper and audio, have received less attention but clearly are all worthy of consideration and experimentation, either alone or in combination.

Our research goals into mobile computing are twofold: first, to investigate new forms of information access and interaction suited to mobile environments and, second, to develop information system platforms capable of supporting them and the experimentation process. In this paper, we present an experimental platform for the rapid prototyping of multi-channel, multi-modal and context-aware information systems. To motivate the architectural design choices and explain the operation of the various components and their interplay, we explain in some detail how the platform was used to develop a tourist information system for the Edinburgh international arts festivals where interaction was based on a combination of speech input-output and interactive paper.

The necessary generality and flexibility required for rapid prototyping was achieved by adopting a *database approach* that enables all information about the application and its interface, the system and the devices to be stored in one or more databases and be updated dynamically at run-time. Further, the approach that we adopt is based on an integration and extension of concepts from open hypermedia systems and web publishing databases.

We begin in Section 2 with a description of our approach and the main components of the platform. In Section 3, we then present the tourist information system that we developed and the specific architecture of this system in terms

MobiCom'05, August 28–September 2, 2005, Cologne, Germany.

of the functionality, devices and modes of interaction supported. Using this example, we go on to explain the details of the various components of the platform, at the same time showing what is involved in developing specific applications. Section 4 details the client controller, while the web publishing and hypermedia components are presented in Sections 5 and 6, respectively. Section 7 provides some discussion of our experiences with the system during user trials carried out in Edinburgh during the 2004 festivals. To show the generality of the platform, we briefly describe in Section 8 how the same platform was used to develop an installation for a media artist. Concluding remarks are given in Section 9.

2. APPROACH

Rapid application prototyping and experimentation in mobile information systems requires a flexible and extensible information platform for content delivery. Not only must it support the requirements of multi-channel and context-aware access that have come to be expected in state-of-the-art mobile systems, but also the highly-dynamic nature of experimental systems where it may be necessary to integrate and reconfigure new devices, resources, modes of interaction etc. at any time. Further, for purposes of experimentation, it may be desirable to offer alternative interfaces and modes of operation in parallel or to easily be able to switch back and forth between different configurations.

In the case of mobile systems, it is especially important to consider the possibilities of different interface modalities in order to meet the demands of users who are on the move and possibly in an environment where normal desktop-style interfaces are inappropriate. If one considers the example of tourists, they do not want to carry heavy equipment and often want to be hands-free. They move between very quiet environments, such as art galleries and theatres, and very noisy environments, such as main streets and bars. Much of the time is usually spent outdoors sightseeing and wandering. They often travel in pairs or groups and collaborate in discovery and planning of activities. For all of these reasons, a simple adaptation of a visual desktop interface to a small screen device such as a PDA may not be the most appropriate solution. Various forms of innovative wearable devices are under investigation and non-visual channels such as audio can play an important role. Paper is an information medium already used extensively in mobile environments and emerging technologies for interactive paper also present interesting possibilities for augmenting paper maps and guides with digital information and services. In our experiments on mobile information systems, we have developed a demonstrator application for tourists at the annual Edinburgh Festivals that offers a variety of interfaces, including ones based on interactive paper and audio as well as one based purely on audio.

To facilitate the rapid prototyping of user interfaces for multiple devices and modes of interaction, it is clear that the user interfaces should be generated dynamically based on content and presentation templates rather than hard-coded. Such a content-driven approach brings the advantage that only the final visualisation step has to be changed to support a new client device or mode of interaction, while the application logic and content remain the same for all output devices. Assuming an architecture based on XML, this essentially means that new XSLT templates have to be written to adapt content to a specific structure and layout represented in the appropriate format of the output channel. In some cases, additional automatic content repurposing may be necessary to conform to the features of a specific device.

It is also important that the development platform should support experimentation with context-awareness, enabling application developers to easily define and change their notion of context and allowing all aspects of a system to be made context-dependent. This means that not only may the information presented to the user depend on factors such as time and location, but also the mode of interaction. For example, as the user moves from an indoor environment to an outdoor environment, the system may automatically switch the output channel from a visual display to an audio device. Only a *multi-modal* information platform can guarantee that the user or system may choose the appropriate access modality based on the current context. It also ensures that the developers and interaction designers have the necessary support to experiment with flexible combinations of various input and output modalities.

To meet all of these requirements, we have developed the web publishing platform OMSwe [15] and a client controller for input/output handling, that together support not only multi-channel, context-aware information delivery, but also multi-modal interfaces. To achieve maximum flexibility, the web publishing platform is an object-oriented database system OMS Pro [9] into which new web publishing concepts have been integrated. The OMS Pro system was itself developed to support rapid prototyping of, not only database applications, but also new database concepts and all aspects of the system can be changed dynamically at run-time. Key to this is the fact that all information is represented as database objects—including application metadata and system data. In the case of OMSwe, this also means that the structure and presentation of web documents, as well as the content, are defined through objects which can be updated at run-time. Further by introducing a context-awareness mechanism that applies to objects, all aspects of the application and system can be made context-aware. Details of the OMSwe system and its role in the platform for mobile information systems are given in Section 5.

As stated above, interactive paper offers interesting possibilities for users to access digital information and services in mobile environments. It is just one example of linking physical objects in the user's environment to digital artefacts. The web is a hypermedia system that links arbitrary digital resources together and, in effect, what we would like to do is to extend the web to physical spaces (sometimes referred to as physical hypermedia). To support cross-media linking, and specifically interactive paper, we have developed a cross-media server, called iServer [19] that allows any form of resources, physical or digital, to be linked together. A plug-in architecture enables new types of resources to be integrated easily. Further, we can link not only static information, but also active content represented by program code which gets executed at link activation time. In Section 6, we describe how iServer was used in the festival application to support an interface based on interactive paper, including the use of active components to allow writing capture.

By integrating the components introduced above—namely OMSwe, iServer and the client controller—as shown in Figure 1, an extremely flexible and powerful platform for exper-

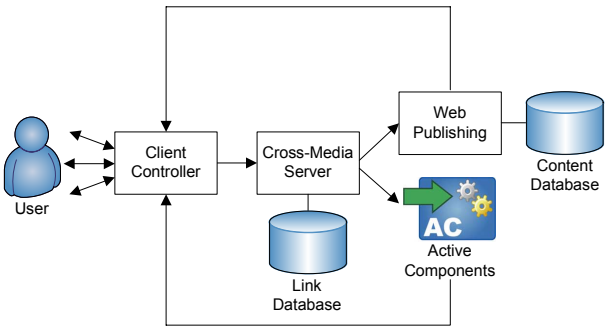

Figure 1: System components

imentation with mobile information systems is achieved. It enables context-aware applications with multi-modal, multi-channel interfaces to be developed quickly and even allows for the simultaneous testing of alternative interfaces and run-time system evolution. Also, these applications may span physical and digital spaces allowing all sorts of physical objects and locations to be digitally augmented.

The *Client Controller* component handles the interaction with the user. Based on a user's interaction and other contextual information such as positional and temporal data, it sends a request to the *Cross-Media Server* component which delivers the linked information. This information can either be data stored in the content database or active content. Information stored in the content database will be transformed to the appropriate format of the output channels by the *Web Publishing* component, which includes a context engine. The result may contain information for multiple output channels and it is the responsibility of the *Client Controller* to activate the appropriate output channels.

The selection of an active component results in the execution of its associated program code on the client and on the server side. An active component may directly return a result to the *Client Controller* by accessing other information resources such as, for example, an external database. However, some active components do not directly return a result to the user but instead process subsequent client requests. This second form of active component is helpful in defining complex interaction patterns which may be allocated to different active components.

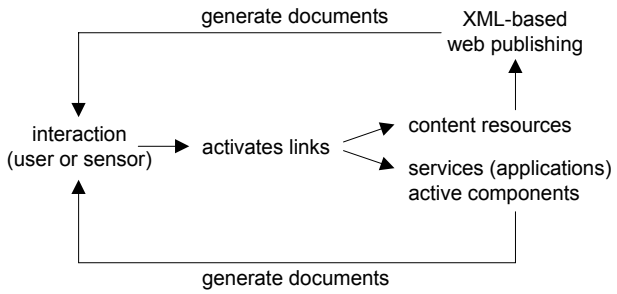

Figure 2: Interaction process

A simplified version of interactions involved in accessing information is shown in Figure 2. Interaction may be explicitly invoked by a user or implicitly triggered by a context-aware object (context sensor) and results in an activation of the resources linked to the triggered event, which can be content resources, services or active components. Finally, a generated document is sent back to the user.

A database approach is used throughout the development of all components. This means that all component metadata are represented as database objects, enabling dynamic updates and system reconfiguration at run-time. In the next section, we present the festival application that is used in later sections to describe the components and their interactions in detail.

3. EDFEST SYSTEM

Tourism has been recognised as a domain with considerable potential for the use of mobile technologies and a number of research projects have developed PDA-based tourist guides, for example, Georgia Tech's Cyberguide [1], the Lancaster GUIDE system [6] as well as Xerox PARC's electronic guidebook [23]. While commercial guides such as the city guides from Vindigo [21] have had some success, ethnographic studies of tourists such as that of Chalmers and Brown [4] report on the fact that it is rare to see tourists on city streets using these guides. Paper maps and guide books continue to be considered the essential tourist accessories. The Campiello project [8] chose to investigate the use of paper as an interface. Paper flyers and special newspapers were distributed around a city and tourists could use them to activate services or input data to a community information system by using scanners at special kiosks.

There are many strong arguments for retaining paper in mobile environments, including the fact that it is light, robust, cheap and easily annotated in a number of different ways [10, 12, 18]. Also, the planning of activities during a city visit often involves combining and comparing information within and across documents such as maps, event brochures and guidebooks and this is easier using paper documents than working with digital mobile devices with small screens. We therefore chose to investigate the use of emerging technologies for digitally augmented paper in mobile tourist environments and, particularly, in the context of a large international arts festival.

The Edinburgh Festival Fringe is the world's largest public arts festival with more than 250 venues and 1700 shows per day. With so many events on offer, visitors often plan which events to visit at short notice and based on contextual factors such as location and time as well as ticket availability. Reviews also play an important role in the selection process and are not only published in newspapers and on-line, but also displayed outside venues and attached to event flyers. Ideally, tourists should be able to access information about the city, the venues, the events and also reviews during the visit and not only during pre-visit planning. The Edinburgh festivals therefore provide an ideal environment for testing technologies for mobile information systems and appropriate means of delivering relevant information in a timely and convenient manner.

We considered various options for the display of information and decided to dispense with any form of visual display such as a PDA, tablet PC or head-mounted display and instead focus on audio output for a first demonstrator. However, since a major goal of the longer-term project is to investigate different interaction modes in mobile environments, a key requirement was to ensure that different access modes could be supported simultaneously and to provide

a flexible platform for development and experimentation. While our main focus for this project was therefore on paper and audio, we also developed basic interfaces suited for tablet PCs, PDAs and head-mounted displays.

The resulting EdFest system was based on the interaction components shown in Figure 3, namely a special interactive paper brochure containing a map and event list, a digital pen and an earpiece used for voice interaction.

Figure 3: EdFest interaction components

The interactive paper brochure is implemented using Anoto technologies [2] originally developed for hand-writing capture. This technology is based on a special pattern of dots that encodes document position information and a digital pen that has a camera alongside the stylus. It can process images in real-time to give up to 100 x and y pen positions per second. This information is stored in the pen and can be transmitted to a computer on demand. Both Logitech and Nokia have developed digital pens based on this technology. We were able to use a prototype of the Nokia pen specially modified by the Anoto engineers to send position data directly and hence enable us to use the pen as an interaction device as well as for writing capture.

A central server has a database with information about venues, events, restaurants and also user reviews. The paper brochure contains a list of events as well as a map which is marked with venues and the user can request information about a venue by simply pointing with the pen at the appropriate location on the map. The system will then initiate a voice dialogue that allows a user to get general information about the venue and events being held there.

The user also has a GPS device, enabling the system to detect their location and support locator and navigation tasks. For example, there is a "Where Am I?" button located at the bottom of the map. The system helps the users locate their position on the map by telling them the general grid position, together with a general guide to placement within the grid e.g. "Grid F5, top right". If the user then points with the pen within that grid, the system will give feedback telling them where to move the pen to arrive at the precise location. Users can also use this functionality to locate events listed in the brochure by pointing to the venue and being told where to find that venue on the map.

The user can access additional information about an event by simply pointing to relevant areas within the brochure event listing, part of which is shown in Figure 4. The user can get information about the artist, a description of the event, names of other events of the same category and also information about ticket prices and availability. In many cases, the choice of exactly what type of information is required is determined through voice dialogue. There is also a rating area where users can input their rating by selecting a star rating between 1 and 5. The average rating is accessed by pointing to the text "Rating".

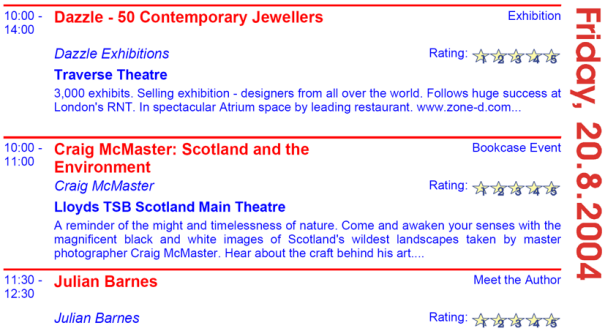

Figure 4: Part of EdFest booklet page

By pointing to the timing information associated with each event, a user can set a reminder for a specific event. The system will then remind the user via an audio message a specified time before the start of the event. Last but not least, the users can enter their reviews either by writing short comments alongside the event listing or by writing longer texts in a separate notebook with the Anoto pattern. These reviews are sent to the central database server and can then be accessed by other users requesting information about the corresponding event.

The resulting EdFest architecture is shown in Figure 5. The system is based on a client-server infrastructure. On the server side, we have the two components: iServer [19], the cross-media link server, and OMSwe [14], the web publishing server. The client side consists of several components described later in this section.

As mentioned in the previous section, iServer uses a plug-in mechanism to support different resource types. In the case of digitally augmented paper, the iServer plug-in manages the link information necessary to map x and y coordinates delivered by the digital pen to digital objects represented by active areas. Thus, in this case, the resources are *pages* and the selectors over resources that define link anchors are arbitrarily complex *geometrical shapes* within pages. As well as mapping to content resources such as images, videos and web documents, active components can be used to bind areas on paper to arbitrary services as described later in Section 6.

The OMSwe server manages the application database, which contains information about the festivals, events, users, venues, etc. and also the definition of interfaces in terms of document structures and XSLT presentation templates. Document structures are defined in terms of components, known as web elements, and the component model allows for both dynamically defined and parameterised web elements. In the case of EdFest, OMSwe delivers VoiceXML

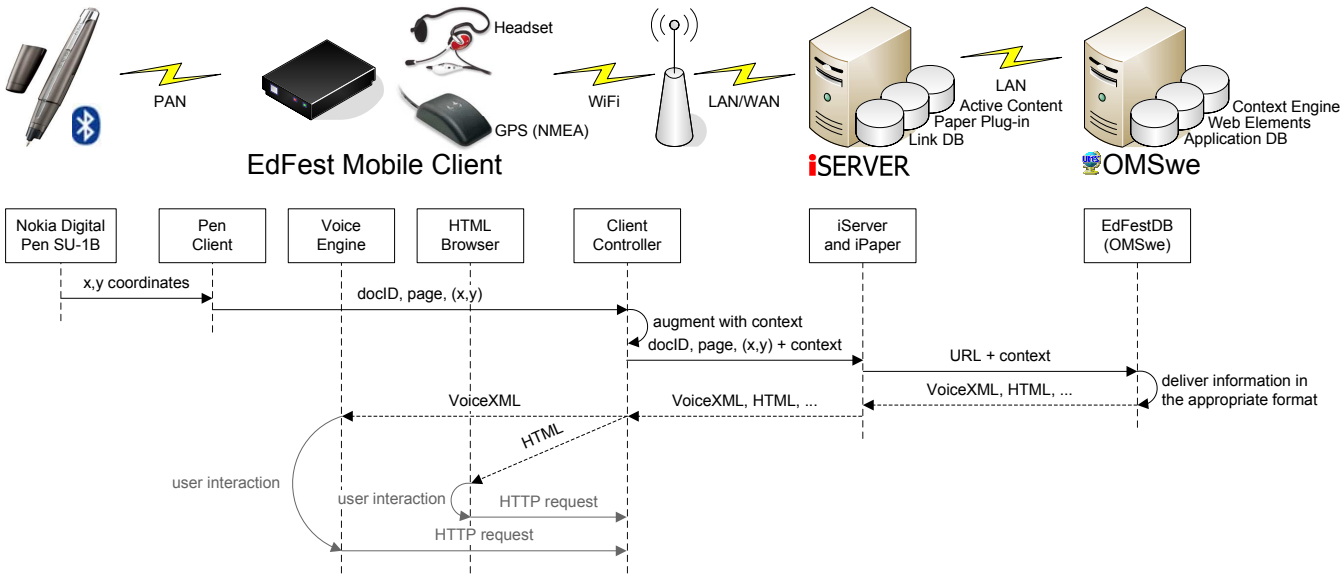

Figure 5: EdFest architecture

files for the voice engine and different HTML pages for pre- and post-visit desktop web browsing, and also for the possible use of a head-mounted display or PDA during the visit. The Context Engine is responsible for managing the context information and, if required, can build high-level semantic context objects from primitive values obtained from hardware or software sensors.

On the client side, the system consists of several components that offer specific functionalities. The Pen Client is responsible for the communication with the digital pen, while the voice engine is responsible for processing the VoiceXML files included in the response from the server and also provides the facility to interact with the system using voice. An HTML browser was also provided on the client side for experimentation with head-mounted displays.

The digital pen is connected to the client computer over Bluetooth. For the audio input and output, we use wired as well as Bluetooth headsets. A GPS sensor is connected to the USB port and provides a serial port emulator, which makes it easier to get the GPS coordinates. The client is connected to the server using an ethernet connection, ad-hoc wireless network, wireless network's public access points or mobile phone GPRS connections. With the first prototype, we did most of the tests in the laboratory using ethernet connections, whereas in the field we used ad-hoc wireless connections.

A request of the Pen Client goes through the Client Controller proxy to the iServer component. When the iServer receives the request including the pen's x and y coordinates, it resolves the activated link to the appropriate URL. For example, a specific active area within the booklet might be mapped to `http://edfest.org/oms?db_anchor=getRating`. The iServer acts as a proxy for the OMSwe publishing framework, which receives the URL and returns the appropriate VoiceXML document dynamically generated from content stored in the database. At the same time, OMSwe delivers the context information contained in the request to the Context Engine. The publishing platform selects the appropriate information from the festival application database and delivers it in the appropriate format according to the current context.

The Context Engine receives all relevant information from the OMSwe server and updates the corresponding context elements such as the user's location, the protocol in use, language settings, the content type and the current callback address. An additional context factor allowed for is the set of users nearby. If some important changes in context happen, the EdFest system can contact the interested user using the callback information. This functionality is used, for instance, to remind users about the start of events.

OMSwe delivers the information to the Client Controller, which is responsible for dispatching it to the appropriate rendering component. In the case of a VoiceXML document, the Client Controller forwards the information to the Voice Engine, which provides audio output to the user. The user can also interact with the system using the microphone provided on the headset.

4. CLIENT CONTROLLER

As we have seen in the previous section, the client of the EdFest system is composed of several interface components such as the Pen Client and the Voice Engine. These components are all HTTP clients since they are either off-the-shelf components such as the Voice Engine or customised components such as the Pen Client that were developed with other application setups in mind. By default, the components are autonomous and do not know anything about each other. They could in principle communicate directly and independently with the server components through an HTTP connection. However, in the case of a platform for mobile information systems, and the EdFest application in particular, we wanted to intertwine the components to form an integrated multi-modal user interface that can provide more functionality than the sum of all functionalities of the separate components.

The Client Controller is the component that takes care of this integration and is the central component on the client

side. It acts as an HTTP proxy for the Pen Client and the Voice Engine. Instead of communicating directly with the server components, the HTTP connection goes through the Client Controller which is therefore able to alter or even replace both HTTP requests from the interface component and responses from the server. It can also trigger side-effects based on the request or the response and further integrates additional components, such as the GPS context sensor.

One of the most important features of the Client Controller is the dispatching of HTTP responses. A request is usually initiated by the user through one of the interface components, either through a touch with the pen or a voice input in response to a voice dialogue. Without the Client Controller, we would have a multi-channel interface where the channels are completely independent. For a full multimodal interface, we need to be able to trigger a request using one modality but display the response in another modality. In the EdFest application, for example, we want to request a voice output or dialogue by pointing with the pen to a particular area in the paper brochure. In this case, the HTTP request is triggered by the Pen Client and the response from the server is interpreted by the Client Controller. It analyses the content type of the response and dispatches the result to the appropriate component (e.g. the Voice Engine). In order to complete the HTTP request that was initiated by the Pen Client, the Client Controller sends back a default response for the content type of the Pen Client upon successful dispatching of the original HTTP response. The dispatch mechanism is kept very flexible so that the Client Controller can easily be extended with new interface components.

Another issue in multi-modal interfaces is the consistency and synchronisation of the various input and output channels. For example, it may not make sense that the Voice Engine continues processing a voice dialogue with information about an event after the user selects another event. User events should therefore be able to interrupt actions. For this reason, the Client Controller analyses the generated requests to determine whether or not to stop actions such as voice output. In the first implementation of the EdFest system, this analysis was kept very simple and the Client Controller stops any running voice dialogues when new requests are received. While, in many cases, this is what the user wants, there are some cases where the user may want to continue listening to the text while perhaps writing a comment on an event. This is one of the interaction issues that we want to experiment with in the future to develop more sophisticated means of deciding when to interrupt actions based on an analysis by the Client Controller of the request content.

As already mentioned in Section 3, the EdFest application also makes use of context information managed by the Context Engine on the server side. A lot of the context information such as the GPS coordinates and information about the devices in use originates on the client side. The Client Controller is also responsible for collecting this information and propagating it to the Context Engine on the server side by hitchhiking with the HTTP requests from the interface components. This can be done by augmenting the URL of the HTTP request with additional parameters or generating a multi-part request depending on the quantity of the data to be sent. On the server side, the additional context parameters are passed transparently through the iServer component to the OMSwe system, where they are passed to the Context Engine.

If the user does not issue any requests for a certain length of time, the Client Controller can initiate a special context update request autonomously. Such a request is used to update the context information and act as a sign of life signal from the client. This ensures that the context information on the server side, such as the location of all system users, is kept up to date.

The Client Controller provides an additional mechanism for pushing information from the server to the client by acting as an HTTP server and listening for notification requests. These requests can be sent by the OMSwe system or, indeed, any other component of the system. The IP address, port and URL that the Client Controller listens to are part of the context information that it sends to the Context Engine which keeps track of the callback information for all system users. If a server-side component wants to send a notification to a user, it looks up the corresponding callback URL and makes a notification call to this URL. In the current EdFest prototype, we use this feature to inform users of friends nearby as well as for event reminders.

5. PUBLISHING COMPONENT

Nowadays a web publishing framework has to support multi-channel access and, increasingly, context-awareness. Although a large variety of tools and technologies are available to support the publishing of static and dynamic data on the web, many of these lack a well-defined declarative model. Within the research community, this problem has been recognised and a number of model-based approaches have been proposed. In contrast to most of these, our approach is system-based rather than tool-based. By this, we mean that, instead of developing tools on top of existing database technologies, we wanted to develop a database system with support for web publishing integrated into the core model and system.

OMSwe [15] is the resulting web publishing framework. To handle context-dependent access to data, a notion of state had to be integrated into the database system to represent both interaction and context states. Context-dependent information delivery is then controlled by matching the context state of a request to context-specific versions of the objects involved. The integration of a general context engine [3] into the database is a major step towards supporting a rich variety of context-aware applications. Since the context dimensions are stored in a database rather than being hard-coded in the application, it is possible to easily adapt applications by adding new context dimensions.

The model of context that we use is similar to that proposed by others for HTML [22] and also semi-structured data [20]. Context is defined in terms of a set of *characteristics* which specify the various factors to be taken into account in deciding on the content, structure and presentation of a document. These characteristics can include anything from user-related factors such as language preference and location to system-related factors such as the request protocol and client device. The EdFest application allows for several context dimensions such as a user's identity, his current location, the time, the language setting and the protocol used.

Objects which are deemed to be context-sensitive can have multiple variants. Each variant has a set of characteristic-value pairs that define the contexts in which it is appropriate. For example, in EdFest, the template object used to

render a venue object has a variant for each output format that needs to be generated as represented by the format characteristic, e.g. `format=html`. Variants may also have a valid time period associated with them to allow time-dependent components of a document to be specified.

While it is beyond the scope of this paper to describe the context model and mechanisms in detail, it is important to note that, for a given request and context state, variants are selected according to a best match rather than an exact match. This avoids having to specify variants for each possible context state. Characteristics such as language can be specified as ordered lists of preferences and defaults are defined for cases where no match is found. For cases where the characteristic value is critical, it may be specified as mandatory. Details can be found in [15].

The context-awareness and variant features of OMSwe allow a very elegant implementation of the multi-channel requirements. For the EdFest prototype we have implemented four different channels: HTML, VoiceXML, PDF and a special data export channel. The HTML channel is currently used only for the display of captured notes for users with a head-mounted display. In addition, we have also used it internally for development and technical testing. The aim of the current EdFest system was clearly to support on-site activities during the visit to the festivals. But as we also aim to support pre- and post-visit activities in the future, the HTML channel will be extended for these activities. An important channel, especially for accessing dynamic information, is the *voice channel*. OMSwe generates VoiceXML dialogues from the data objects for the display of information and access to services. The dialogues usually contain multiple forms that the user can navigate by selecting options through voice input. For experimentation, we also developed a version of the EdFest system that has a voice-only interface.

However, the main channel of the EdFest prototype is not the voice interface, but the *paper brochure*. This brochure is also generated by the OMSwe publication framework. The system contains templates that generate an XSL:FO [17] document with the content of the booklet dynamically generated from information stored in the database. This includes the listings of events, the map and other pages. The XSL:FO document is then transformed into a PDF document which can be printed and bound to a booklet. The Anoto pattern that is used by the Nokia pen has to be added manually to the generated PDF document. We are currently working on integrating this step into the publication process, so that the booklets can be printed directly out of the web publishing system.

With the dynamic generation of the paper brochures and booklets, we also need an automated export of the mapping information for the iServer component to define the necessary cross-media links. As the iServer component uses a separate database for the linking information, this database has to be updated upon creation of a new brochure by the web publication system. For this reason, we have implemented a special channel in the content publishing framework that generates an XML description of the linking information for the paper brochure. This XML description can be imported by the corresponding tool of the iServer component.

For the EdFest application, we printed the booklets before going to Edinburgh for the user trials and all users had the same booklet. For this setup, we could also have produced the booklets with other tools, such as word processors or desktop publishing tools, but the authoring of the links from paper would have been difficult to control, especially since the booklet went through many design changes. Another major advantage of using a web publishing framework to generate the printed documents is that we can also dynamically produce customised booklets.

The four channels of the web publishing framework are independent. The integration, if necessary, is done by the Client Controller as already described. From the point of view of the content publication component, it just provides a multi-channel interface to some information and services. This means that the client has to issue separate requests for each document that it wants to access since the web publishing framework can only return one response for every request. For example, it is not possible to return both an HTML document and a voice dialogue as the result of a single request. This is a limitation that is not well suited to a true multi-modal user interface. With the Client Controller, we can currently use a workaround in that we can analyse a request and transform it into multiple requests for the different channels or modalities if required. The drawback however is that the client has to specify the channels for the response of the request. An activation of additional output channels based on context information or data in the EdFest application database would not be possible. For this reason, we are extending the current implementation of OMSwe with the ability to return multi-part responses. Which and how many responses will be generated can be determined dynamically based on application data, publication data or context information. The multi-part responses would be split by the Client Controller and the individual responses dispatched to the corresponding handler components. This would allow, for example, that a single response could return control information for the Pen Client, some voice output and an image to be shown on the head-mounted display. Similar approaches to multi-modal interfaces are currently also being investigated as part of W3C's Multimodal Interaction Activity (MMI) [13]. The XHTML+Voice (X+V) specification, for example, allows the embedding of VoiceXML elements in an XHTML document. There are web browsers that already interpret this format, for example, Opera's Multimodal Browser. The availability of standards and corresponding tools would of course tremendously simplify the development of such multi-modal interfaces.

6. PAPER AS A MOBILE DEVICE

As mentioned earlier, the iServer platform is a cross-media information management framework supporting digital as well a physical artefacts. The iPaper plug-in that we developed for interactive paper and used in EdFest was developed in the context of a European project Paper++. More details of the specific technologies developed within this project and the motivations behind the research can be found in [11, 19].

In this section we introduce *active components*, a new type of resource that was developed for the iServer platform to support the design of complex interaction components as required by the EdFest prototype. While regular links just return a single piece of information such as an HTML page or a movie clip, active components are Java objects that become instantiated based on a configuration stored in the iServer

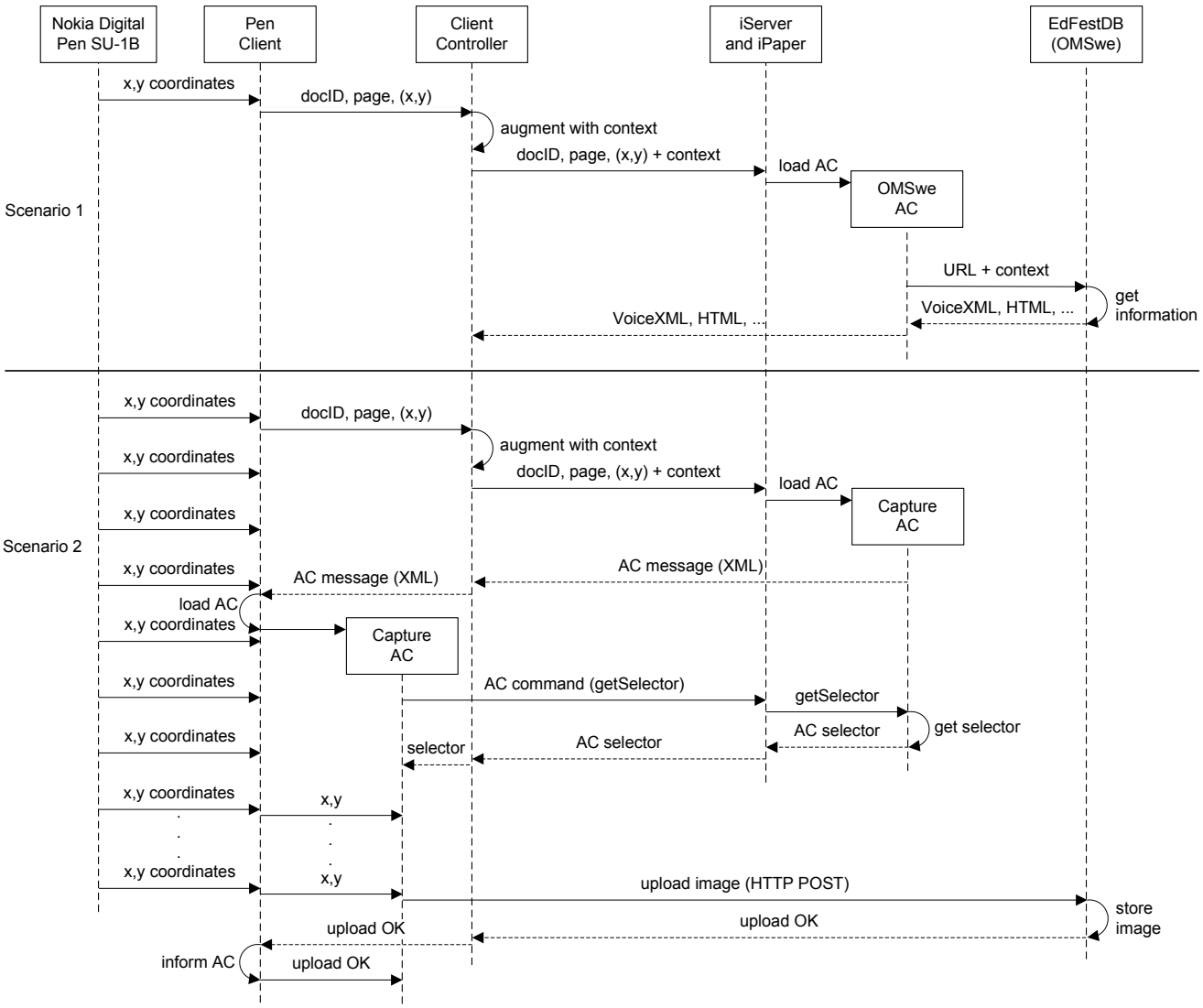

Figure 6: Client- and server-side active components

database and can be executed on the server as well as on the client side. The Pen Client is a component that can run active components on the client side. It distinguishes two working modes: a *browsing mode* where no active component is running on the client side and an *active mode* where an active component has been instantiated and is currently running. On incoming pen events, the Pen Client either sends a regular request to iServer or delegates the pen request to a running active component.

We present two scenarios where active components have been used on the server as well as on the client side to manage complex operations within the EdFest system. As mentioned earlier, iServer with the iPaper plug-in mainly stores metadata about active regions within the festival brochure while the actual festival data about venues, performances etc. are stored in the OMSwe application database.

In the first scenario, the *OMSwe active component* runs on the server side and mainly acts as a proxy for information that is actually stored in the external OMSwe application database. This active component is typically used when the user points somewhere in the EdFest booklet with the pen to get additional information in the form of audio output. The Pen Client sends an HTTP request to iServer and the plug-in for digitally augmented paper resolves the positional information to the appropriate target resource as shown in scenario 1 of Figure 6. In the case that the information is stored in the external EdFest database, the resolved resource will be an OMSwe active component. Based on the active component's identifier (name), an object of the corresponding Java class is instantiated and initialised with the active component's supplementary data stored in the iServer database. In the case of an OMSwe component, this data includes a query encoded as an HTTP request that can be sent to the EdFest application database. Before this request is actually sent to the OMSwe server, it has to be augmented with the contextual information of the incoming iServer request that has been added by the Client Controller. After the query has been sent to the external OMSwe database, the OMSwe active component sends the response directly back to the Client Controller which dispatches it to the

appropriate output channel as described in Section 4. We have used a single database for managing information about the festival, but the concept of an active server-side proxy component could be used to integrate various heterogeneous data sources.

The Nokia Digital Pen that we used for the Edinburgh festival trials, continuously streams data in the form of positional information to the Pen Client. While the user is browsing the brochure, this information is stored in a buffer and, only after a certain amount of time, is it possible to actually send a request to iServer. This filtering of pen events works fine in the case of a user pointing to specific areas of the paper brochure which normally should result in a single request. Furthermore, the Pen Client always checks if there is still an outstanding response for a request that has been sent earlier, in combination with a fixed timeout. If there is a conflict, the new request is rejected and the Pen Client acoustically informs the user that another request is currently being processed. In this browsing mode, each pen event, after the described filtering, results in a single request which is sent to the server component as outlined earlier in Figure 5. However, for certain tasks, it makes sense that the Pen Client processes multiple pen events before sending a request to iServer. This enhanced application logic of the Pen Client can be realised by applying client-side active components. In the remainder of this section, we present a client-side active component that was used within the EdFest demonstrator to capture hand-written user notes.

The lower part of Figure 6 shows a second scenario, where a note is captured based on the user's interaction with the festival booklet, involving a client-side active component. If a user starts to write in an active area that has been defined as a capture area, the Pen Client first sends a single event to iServer as is normally done in the browsing mode. The iPaper plug-in of iServer performs a lookup for the specific pen position and returns a *capture note active component*. An instance of the capture note component is instantiated on the server side, based on the active component's configuration information stored in the iServer database. In the case of the capture note component, this information includes an upload address, i.e. a URL where the captured note finally should be uploaded, as well as a timeout parameter which is used for non-explicit termination of the capturing process as described later. The active capture note component loaded on the server side, sends an XML message including the identifier (name) of the active component as well as various configuration parameters back to the Pen Client. The MIME type of the HTTP response which is sent back is set to `application/paperpp.client` so that the Client Controller knows that it has to be dispatched to the Pen Client.

The Pen Client receives the XML message and identifies it as an active component response. An instance of the appropriate capture note active component stub is instantiated based on the identifier of the active component XML message and the additional information stored within the message. During this initialisation phase the capture note active component has to obtain information about the active region (selector) to which it is actually bound. Therefore, the client-side active component sends a special `getSelector` active component command to the server-side active component which looks up this information and sends back a response containing the requested selector. Note that in the case where this information is only needed once during the initialisation phase of the client-side active component, it could always be directly integrated into the first active component message to reduce the number of requests and therefore improve the system's performance. When the client-side stub for the capture note component is created, it asks the Pen Client for the time when the last request was sent to the server, which is the time when the request for the capture note component itself was initiated. This information is used later to fetch the appropriate information from the buffer. After the active component has been loaded, the pen switches from browsing mode to active mode which simply means that subsequent pen events are delegated to the client-side active component instead of directly being sent to the server. As explained before, the capture note client-side component requested information about the active region that was defined as a capture area.

```
<?xml version="1.0" encoding="UTF-8"?>
<iserver>
<activeComponent id="capture" creator="beat">
  <name>Capture Note Active Component</name>
  <identifier>CAPTURE_NOTE</identifier>
  <properties>
    <parameter>
      <key>org.ximtec.iserver.ac:uri</key>
      <value>http://edfest.org/oms?
        db_anchor=edf_comment_add&amp;
        db_edf_comment_add1=e-Dazzle50ContempJws
      </value>
    </parameter>
    <parameter>
      <key>org.ximtec.iserver.ac:uploadParam</key>
      <value>db_edf_comment_add2</value>
    </parameter>
  </properties>
  <timeout>3000</timeout>
</activeComponent>
</iserver>
```

Figure 7: Active Component

The capture process is completed when either the pen leaves the active capture area or after the predefined timeout, which is also a parameter of the capture note component, elapses. In the meantime, all pen events are stored in the buffer. After the capture process has been terminated by one of the two possibilities just described, the capture note component does a lookup in the buffer to get all positional information that has been acquired during the capture process. This lookup is based on the temporal information that the active component requested in its initialisation phase. Finally, the captured information is sent to the predefined upload URL either as a JPEG image or as scalable vector graphics (SVG). The OMSwe server sends a response confirming that the upload of the image was successful which is delegated to the capture note active component. The active component informs the Pen Client that it has finished its work and it is immediately unloaded by the Pen Client which switches back to the default browsing mode.

The active components can be defined directly in the database or they can be imported from an XML document. Figure 7 shows an active component with its main attributes. The `name` is used to find a specific active component whereas

(a) Browsing (b) Voice interaction (c) Writing

Figure 8: User trials at the Edinburgh festivals

the `identifier` is applied to bind the definition of an active component to its related Java class. The example shows an active component for information capture and therefore the first parameter with the `uri` key defines the web address for uploading the captured information. The second parameter contains an upload parameter which will be added to the server request by the client-side active component. Finally, the active component has a `timeout` of 3000 milliseconds which means that if the idle time is greater than this threshold, the capture process will be terminated automatically.

The concept of having client- and server-side active components which can communicate by sending special active component messages has proved to be useful if the client-side component has to get additional information stored in the iServer database. Note that the two active components presented here are just two possible implementations of the very flexible and powerful active component concept which enormously simplifies the implementation of complex interaction components. Various other active components have been implemented as part of the EdFest prototype to support different interaction tasks such as the rating of an event or the localisation of a venue.

7. EDFEST AT THE FESTIVAL

Initial tests and user trials of the EdFest system took place in Edinburgh during August 2004. Usability trials were carried out during a three-day period on various locations in the city, including public places and locations in and around festival venues. They involved the testing of the EdFest prototype, a mix of semi-structured interviews, observations based on video and audio recordings and also user questionnaires. While it is beyond the scope of this paper to describe the user studies in detail, we include in this section some general remarks on the outcomes.

Test users included experts from the HCI and CSCW fields who work in Edinburgh or were visitors to the festival as well as tourists interviewed in public places. A total of 3 expert users and 8 non-expert users completed detailed trials lasting around 45-60 minutes with full video recordings. In addition, we carried out a number of smaller trials as well as observing and videoing tourists in various venues and public spaces such as streets, bars and cafes. A user working with the paper brochure is shown in Figure 8.

The interaction with the system is usually initiated by pointing to a specific area of the booklet with the pen. For some of the users it was unclear that they could also talk to the system in order to navigate through the voice dialogues. Even after learning that they have this possibility, some users went on using the pen for input, in parallel to the voice input. This demonstrates the necessity of a multi-modal interface were people can use their capability of performing redundant and continuous interaction to achieve their goals.

A few users expressed a reluctance about having to talk to the system in public places in order to get information or perform some task. One option would be to remove voice as an input channel and instead use only paper for input and audio for output only. The removal of voice input would certainly resolve the ambiguity of having to use a particular modality for a particular interaction, but of course supporting both modalities would be a richer and more natural interface.

One problem of using the pen as a pointing device was the fact that some users were concerned that they would mark the brochure. We have found this to be a general problem associated with the dual mode of the modified pen which can act both as a selection and writing device. Ideally, the pen itself should have a mechanism to switch between modes, for example by clicking on the end it could retract the writing stylus and switch from writing to selection mode. We consider such amendments to the design of digital pens essential if they are to become devices with this dual functionality.

Generally, the response to the interactive brochure was positive and users found the map-based interaction, including the locator functionality, to be particularly intuitive and very useful. There was positive feedback concerning the means of inputting and getting event ratings as well as for setting the reminders. More problems were experienced with interaction through the printed event lists as here it was less clear to users what response to expect from pointing to various positions on the page. In the case of event lists, there are many possible design layouts that can be chosen and, given the novelty of the technologies, a lack of design guidelines.

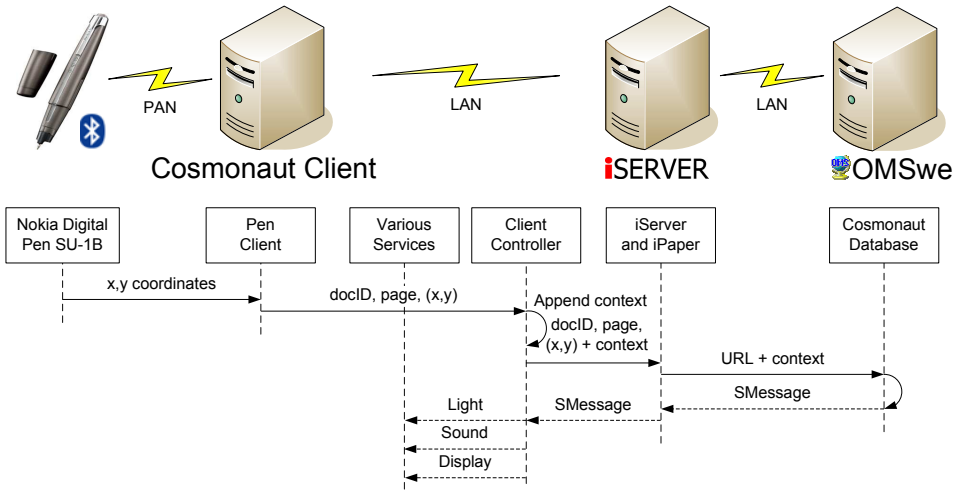

Figure 9: Lost Cosmonaut architecture

Here, the feedback from the user studies provided valuable input towards experimenting with alternative designs in the future.

8. INTERACTIVE ART INSTALLATION

The *Lost Cosmonaut* [7] is an interactive art installation which was realised in collaboration with an artist in residence based on the same experimental mobile platform as EdFest. The goal was to investigate the use of technologies for interactive paper in interactive narratives and story writing.

The general setup of the Lost Cosmonaut installation is shown in Figure 10. A user sits in a dark room in front of a semi-circular desk. The wall in front of the user contains a large round hole which is used as a screen for projecting digital information such as images, videos and animations. On the desk there is a Nokia Digital Pen and three documents—a star map, a book, and a collection of love letters—forming part of an interactive narrative about Soviet cosmonauts lost in space. Movement of the documents determines which one is the current focus of interest.

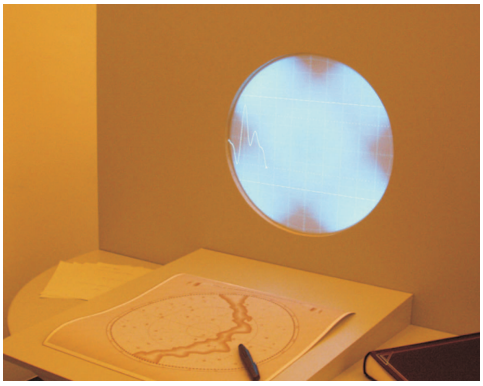

Figure 10: Lost Cosmonaut setup

When a visitor interacts with the documents, the content presented on the round screen as well as the ambient sound and the lighting in the room change according to the user actions. The documents have some pre-authored content, but visitors are encouraged to add texts and drawings to the artefacts themselves. While the linearity of the story is already broken by giving a user the freedom to select arbitrary information in the three documents, each user further becomes an author of the story by adding his own content. Thereby, the interactive narrative collaboratively written by different users evolves over time.

For the Lost Cosmonaut installation, it was important to have an architecture that is flexible in managing digital information and delivering information on a variety of output channels. The overall Lost Cosmonaut architecture, shown in Figure 9, is similar to the one used in the EdFest demonstrator with the Pen Client, the Client Controller, iServer and the OMSwe publishing component. A new Service Message (SMessage) format was introduced for controlling various output channels such as light control, ambient sound and video. These mood changes are influenced by the current document that the user is working on. All documents are tagged with RFID identifiers which are detected by an antenna placed on the back side of the table and handled by an iServer RFID plug-in.

The requirements of the installation changed as the artist developed his ideas and it was essential to have an extensible information platform such as the one presented in this paper. The platform not only supported the rapid prototyping of the application in terms of content and services, but also enabled easy integration of new input and output channels.

9. CONCLUSIONS

We have presented a platform that supports rapid prototyping of mobile information systems. While we generally advocate the use of rapid prototyping in system development, we feel that it is even more crucial in the relatively new area of mobile applications intended to provide context-aware information services based on emerging technologies. Interaction with these services becomes a major issue and there are many innovations in the features offered by new devices. It is therefore important to experiment, not only with alternative modes of interaction, but also multi-modal interfaces.

We have shown that by combining general platforms for context-aware web publishing and cross-media services, we achieved a very flexible platform for mobile information systems. This platform supports multi-channel, context-aware applications that may even span physical and digital spaces, thereby enabling digital information and services to be linked to places and objects in a user's environment.

The EdFest project served as both a driving force for the design and development of the platform and a first major demonstration of its use. The project integrates many different aspects of mobile systems addressed individually in other research projects and, hence, presented us with many challenges. The system described in this paper represents the first phase of a three year project and trials will be carried out at the festival each August during this period. Three major versions of the EdFest system will be developed and, for each of these, there will be a great deal of smaller tests with alternative designs. It is therefore vital for us that we have a platform that, not only supports the functionality we need, but also enables all aspects of the system to be changed easily and quickly. This is achieved by using data-driven approaches, where all information about the application and its configuration is represented in one or more databases in terms of objects that are subject to semantic consistency constraints. This means that they can be updated dynamically at run-time, but that there are guarantees that this is done in a controlled way.

10. ACKNOWLEDGEMENTS

We thank the other members of the EdFest team: Barbara Aeppli, Philipp Bolliger, Marco Dubacher, Michael Grossniklaus, Slavisa Maslic and Alexios Palinginis. We also thank Rob Procter and his team at the University of Edinburgh for their help with the user trials.

11. REFERENCES

[1] G. Abowd, C. Atkeson, J. Hong, S. Long, R. Kooper, and M. Pinkerton. Cyberguide: A Mobile Context-Aware Tour Guide. *Wireless Networks*, 3:421–433, 1997.

[2] Anoto AB, http://www.anoto.com.

[3] R. Belotti, C. Decurtins, M. Grossniklaus, M. C. Norrie, and A. Palinginis. Interplay of Content and Context. In *Proc. of ICWE 2004, 4th Intl. Conference on Web Engineering*, Munich, Germany, July 2004.

[4] B. Brown and M. Chalmers. Tourism and Mobile Technology. In *Proc. of ECSCW 2003, 8th European Conference on Computer Supported Cooperative Work*, Helsinki, Finland, September 2003.

[5] B. Brown and E. Laurier. Designing Electronic Maps: an Ethnographic Approach. In L. Meng, A. Zipf, and T. Reichenberger, editors, *Map Design for Mobile Applications*. Springer Verlag, 2004.

[6] K. Cheverst, N. Davies, K. Mitchell, A. Friday, and C. Efstratiou. Developing a Context-Aware Electronic Tourist Guide: Some Issues and Experiences. In *Proc. of CHI 2000*, The Hague, The Netherlands, September 2000.

[7] The Lost Cosmonaut, Interactive Art Installation, http://www.lostcosmonauts.ethz.ch/.

[8] A. Grasso, A. Karsenty, and M. Susani. Augmenting Paper to Enhance Community Information Sharing. In *Proc. of DARE'2000, Designing Augmented Reality Environments*, Elsinore, Denmark, April 2000.

[9] A. Kobler, M. C. Norrie, and A. Würgler. OMS Approach to Database Development through Rapid Prototyping. In *Proc. of WITS'98, 8th Workshop on Information Technologies and Systems*, Helsinki, Finland, December 1998.

[10] D. M. Levy. *Scrolling Forward: Making Sense of Documents in the Digital Age*. Arcade Publishing, October 2001.

[11] P. Luff, C. Heath, M. C. Norrie, B. Signer, and P. Herdman. Only Touching the Surface: Creating Affinities Between Digital Content and Paper. In *Proc. of CSCW 2004, ACM Conference on Computer Supported Cooperative Work*, Chicago, USA, November 2004.

[12] C. C. Marshall. Annotation: From Paper Books to Digital Library. In *Proc. of DL'97, 2nd ACM Intl. Conference on Digital Libraries*, Philadelphia, USA, July 1997.

[13] W3C Multimodal Interaction Activity, http://www.w3.org/2002/mmi/.

[14] M. C. Norrie and A. Palinginis. Empowering Databases for Context-Dependent Information Delivery. In *Proc. of UMICS 2003, Workshop on Data Ubiquitous Mobile Information and Collaboration Systems*, Klagenfurt/Velden, Austria, June 2003.

[15] M. C. Norrie and A. Palinginis. Versions for Context Dependent Information Services. In *Proc. of COOPIS 2003, 11th Intl. Conference on Cooperative Information Systems*, Catania, Italy, November 2003.

[16] A. Pashtan, R. Blattler, A. Heusser, and P. Scheuermann. CATIS: A Context-Aware Tourist Information System. In *Proc. of IMC 2003, 4th Intl. Workshop of Mobile Computing*, Rostock, Germany, June 2003.

[17] D. Pawson. *XSL-FO: Making XML Look Good in Print*. O'Reilly & Associates, August 2002.

[18] A. J. Sellen and R. Harper. *The Myth of the Paperless Office*. MIT Press, November 2001.

[19] B. Signer and M. C. Norrie. A Framework for Cross-media Information Management. In *Proc. of EuroIMSA 2005, Intl. Conference on Internet and Multimedia Systems and Applications*, Grindelwald, Switzerland, February 2005.

[20] Y. Stavrakas and M. Gergatsoulis. Multidimensional Semistructured Data: Representing Context-Dependent Information on the Web. In *Proc. of CAiSE 2002, 14th Conference on Advanced Information Systems Engineering*, Toronto, Canada, June 2002.

[21] Vindigo City Guide, http://www.vindigo.com.

[22] W. Wadge, G. Brown, M. Schraefel, and T. Yildrim. Intensional HTML. In *Proc. of PODDP 98, 4th Intl. Workshop on Principles of Digital Document Processing*, Saint Malo, France, March 1998.

[23] A. Woodruff, P. Aoki, A. Hurst, and M. Szymanski. Electronic Guidebooks and Visitor Attention. In *Proc. of ICHIM 2001, 6th. Intl. Cultural Heritage Informatics Meeting*, Milan, Italy, 2001.