Wireless Netw (2007) 13:855–870
DOI 10.1007/s11276-006-9858-y

Context-aware platform for mobile data management

Moira C. Norrie · Beat Signer · Michael Grossniklaus · Rudi Belotti · Corsin Decurtins · Nadir Weibel

Published online: 23 October 2006

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 platforms to support the rapid prototyping of multi-channel, multi-modal, context-aware applications. The Java-based platform presented here is based on an integration of a cross-media link server and an object-oriented framework for advanced content publishing, along with a Client Controller and Context Engine. We also describe how this platform was used to develop a mobile tourist information system for an international arts festival where interaction was based on a combination of interactive paper and speech output.

Keywords Mobile information systems · Interactive paper · Web publishing · Context-awareness · Rapid prototyping

M. C. Norrie (✉) · B. Signer · M. Grossniklaus · R. Belotti · C. Decurtins · N. Weibel
Institute for Information Systems, ETH Zurich, CH-8092 Zurich, Switzerland
e-mail: norrie@inf.ethz.ch

B. Signer
e-mail: signer@inf.ethz.ch

M. Grossniklaus
e-mail: grossniklaus@inf.ethz.ch

R. Belotti
e-mail: belotti@inf.ethz.ch

C. Decurtins
e-mail: decurtins@inf.ethz.ch

N. Weibel
e-mail: weibel@inf.ethz.ch

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.

The tourist domain has been a focus of several research projects in mobile information systems [22] and yet, while many of these projects have successfully demonstrated both the advantages of context-awareness and the potential of new mobile technologies, most of them fail to meet the requirements of performing complex, often collaborative, tasks in mobile environments. Tourism is generally a social activity and part of the enjoyment is planning activities together with family, friends and locals. PDAs are often used in mobile applications but their screens 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 [6]. Some researchers have therefore experimented with tablet PCs to provide better functionality [7], but clearly these further restrict mobility as they are much heavier than PDAs and require the use of both hands. Another problem of mobility is that of power and the problem of electronic devices is that they deliver absolutely no information unless switched on and using power.

For all of the above reasons, we wanted to investigate the potential of alternative technologies for access to both static

and dynamic information in mobile environments. In particular, we were interested in experimenting with paper-based interfaces exploiting new forms of digital pen and paper that could be used both for interaction and data capture. One major advantage of these interfaces is the fact that core, static information is printed and hence accessible even when the digital system is not in use. The digital system provides optional, valued-added services that give access to supplementary information, dynamic data and transactional services. The choice of output channel was some form of visual display such as a PDA or head-mounted display, or voice based on a text-to-speech engine. To support these investigations and the rapid development of applications as well as the experimentation process itself, we developed an experimental platform for mobile information systems [5]. The system was used to develop a mobile guide for visitors to the Edinburgh international arts festival in 2004 where we carried out experiments with two variants of the system—an interactive paper brochure combined with voice input/output and a version based only on voice input/output.

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. The database management systems are object-oriented and manage code as well as data, inclusive of triggers, and therefore can be considered as driving the application rather than being simple repositories of data. Further, the approach that we adopt is based on an integration and extension of concepts from open hypermedia systems and content publishing databases.

Based on our experiences with the 2004 festival system described in [5], we refined the platform and re-designed the application in 2005. A major change in the architecture was to replace the content publishing platform with a newly developed system with integrated support for context-awareness through an extended concept of object variants and versions [4]. With this change, we also moved to a platform that is entirely Java-based. With respect to the application, we focussed on an interface that is based on a set of interactive documents—an event brochure, a map and a bookmark—and with a voice output channel. This interface is offered in parallel to HTML browser interfaces on desktop PCs or PDAs, including a kiosk system that enables users to print personalised interactive daily event schedules.

In this paper, we present the overall architecture and main components of our Java-based platform for the development of multi-channel, multi-modal, context-aware applications. The case study of the Edinburgh festival system for 2005 is used to explain the operation of the various components and their interplay as well as motivating the architectural design choices.

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 festival guide that we developed and the specific architecture of this system in terms 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 content 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 2005 festival. Concluding remarks are given in Section 8.

2 Platform overview

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. Although these are all requirements for experimentation, we note that, increasingly, there are requirements for operational systems to also offer such flexibility in order that enterprises can react quickly to new technologies and customer demands.

One of the major challenges of mobile data management is the issue of how to support interaction for users on the move [10]. Consider the case of tourists visiting a city. 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 the 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. In fact, although tourist guides for PDAs are now commercially available, studies of tourists show that paper maps and guides are still considered the essential tourist accessories. It is therefore important that other forms of innovative interfaces are investigated and it is expected that wearable devices and non-visual channels such as audio can play an important role. One of our major interests is the possibility of

retaining familiar forms of paper documents such as maps, guides and event brochures and using emerging technologies for digital pen and paper to augment these with digital information and services. In this way, there is an easy and natural transition from the paper world to the digital world and both can be supported alongside each other with bridges between them. In our experiments on mobile information systems, we have developed a number of demonstrator applications for tourists that offer a variety of interfaces, including ones based on interactive paper and audio as well as regular HTML web interfaces for a range of fixed and mobile devices.

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 view, 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 approaches a kiosk with a display screen, the system may automatically switch the output channel from an audio device to visual output on the screen or a combination of both. 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 content publishing platform XCM 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 content publishing platform has been implemented as an extension of the object-oriented database system OMS [15, 21] which has been augmented with new content publishing concepts. The OMS platform 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 XCM, this also means that as well as content, the view, structure and layout of documents are defined through objects. Further, by introducing a model of context together with the concept of object variants, all aspects of the application and system can be made context-aware. Details of the XCM 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 information management platform, called iServer [25], that allows any form of physical or digital resources 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 the content publishing component (XCM), the cross-media server (iServer and iPaper) and the Client Controller—as shown in Fig. 1, an extremely flexible and powerful platform for experimentation with mobile information systems is achieved. It enables context-aware applications with multi-modal, multi-channel interfaces to be developed quickly

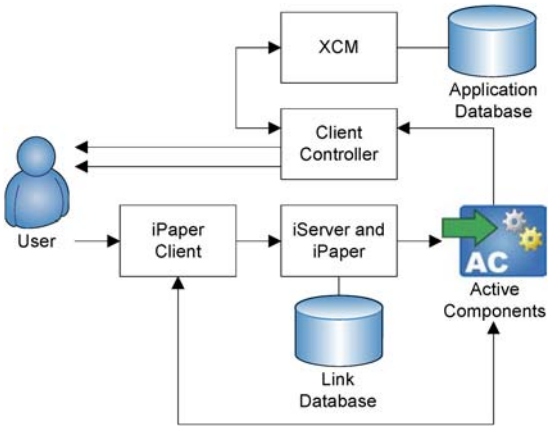

Fig. 1 System components

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 selection of an active component results in the execution of its associated program code on the *iPaper Client* and on *iServer*. An active component that is executed on iServer may directly access information resources that are, for example, stored in external databases. However, some active components do not immediately 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.

The *Client Controller* component handles some of the user interaction. Any request from an active component that is sent to *XCM* is also processed by the Client Controller which is configured as a proxy for the XCM server within iServer. The Client Controller augments any request with contextual information such as time or location before forwarding it to XCM. The information data that is stored in the application database will be transformed to the appropriate format for the current output channels by XCM. The result may contain information for multiple output channels and it is the responsibility of the Client Controller to activate the appropriate output channels.

A simplified version of the interactions involved in accessing information is shown in Fig. 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 dynamically 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.

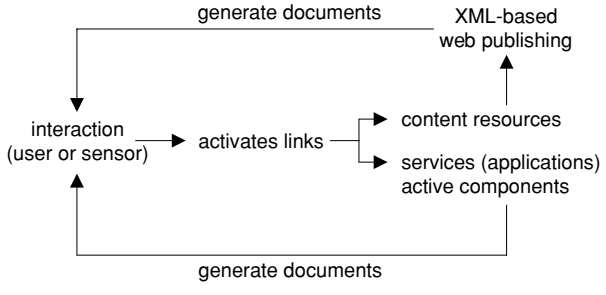

Fig. 2 Interaction process

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 [9] as well as Xerox PARC's electronic guidebook [29]. While commercial guides such as the city guides from Vindigo [27] have had some success, ethnographic studies of tourists such as that of Chalmers and Brown [6] 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 [11] 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 [16, 18, 24]. 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 international arts festival and it celebrated its 59th year in 2005 with around 340 venues, 1800 events and 27000 performances over a four week period. 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 previsit 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

Fig. 3 EdFest interaction components

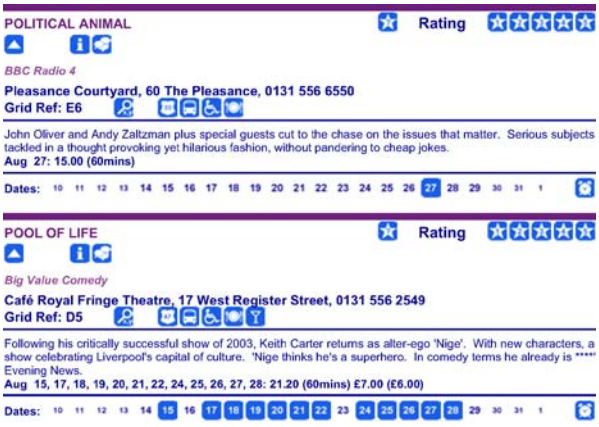

Fig. 4 Part of EdFest booklet page

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 Fig. 3, namely a special interactive paper brochure containing a categorised event list and blank pages for comments, a digitally augmented map showing the positions of all venues, a two-sided bookmark, a digital pen and an earpiece used for voice output.

The interactive paper brochure is implemented using Anoto technologies [2] originally developed for handwriting 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, y) pen positions per second. This information is stored in the pen and can be transmitted to a computer on demand. Logitech, Nokia and Maxell have all 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 continuously 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 event summaries listed alphabetically according to title under the various festival categories such as comedy, dance and theatre. Two event entries in the brochure are shown in Fig. 4.

Pictograms denote active areas which, when activated by touching them with the pen, provide supplementary information from the festival database through a text-to-speech interface. Examples of such information include descriptions of bar and catering facilities on offer at the event venue, warnings about nudity or the use of bad language and information about disabled access.

At the bottom of each event, the user can get information about when the show is performed and whether tickets are still available through pictograms for dates shown within the festival timeline. By pointing to the pictogram depicting an alarm clock to the right of the timeline, a user can set a reminder for a specific event. The system will then remind registered users by sending an SMS message to their mobile phone 30 minutes before the start of the event.

Last but not least, the users can enter their ratings and reviews of events and share them with other users. At the top right of the event entry there is a rating area. By pointing to one of the ratings to the right of the rating label, the user can enter their own rating of an event. Selecting the '?' pictogram to the left of the rating label gives the user access to the average rating of the event. Users can also enter handwritten comments about events using the blank comment pages at the back of the brochure. These comments are captured digitally and using intelligent character recognition (ICR) software, they are converted to text. When the comments are entered they are read back to the user for confirmation. A pictogram in the event entry allows users to hear comments entered by other users.

The second document, the interactive map, shows all of the venues in the city and provides functionality to access information about them. By pointing anywhere on the map, ideally on the number of a venue itself, the user can access a description of the venue that is closest to the position where they pointed. 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 top 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. Similarly, the user can be guided to the location where an event is taking place by selecting the appropriate pictogram in the event brochure. They are then given a map grid reference and again given instructions to help them locate the precise position of the venue on the map.

Finally, the bookmark offers search facilities on one side and ticket reservations and setting of preferences on the other. The searches can specify a number of criteria such as the date, time, category and location indicated by the user's present position or by pointing to the map. On the side visible in Fig. 3, users can set their preferences using check boxes against a list of categories and also book tickets by selecting an event and a date from the brochure and the number of tickets on the bookmark. A confirmation step ensures that all details are correct before the reservation is completed and the user is then given a reservation number. 'Repeat' icons at the bottom of each page can be used at any time to repeat the last message and this allows the user to have the reservation number repeated if necessary. Last but not least, the bookmark also contains a small map with the main navigation features marked. This map also provides access to all of the functionalities available on the full-sized map such as 'Where Am I?'.

A component of the EdFest application that does not interact with the application through the paper-based mobile interface is the kiosk, a stationary client platform that is connected to the rest of the system through a reliable network connection. Sitting at the kiosk, a user can browse through all events—even the ones that are not contained in the booklet as they were latecomers—and see the reviews that other users have entered for a given show. The users can also compose and print a customised personal event schedule that then interacts with the system as a part of the mobile client in much the same way as the booklet does.

The resulting EdFest architecture is shown in Fig. 5. The system is based on a client-server infrastructure. On the server side, we have XCM, the content publishing server that will be described later in this section. On the client side, the system consists of several components that offer specific functionalities. The iPaper Client is responsible for the communication with the digital pen, while the voice engine, Natural Voices from AT&T [3], is responsible for processing the VoiceXML files included in the response from the server. 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 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 public access points or mobile phone GPRS connections. While developing the interactive festival system, 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 iPaper Client is directly sent to the iServer component. 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, y) coordinates delivered by the digital pen to digital objects represented by active areas that are defined by arbitrarily complex geometrical shapes within pages. As well as mapping to content resources such as images, videos and web documents,

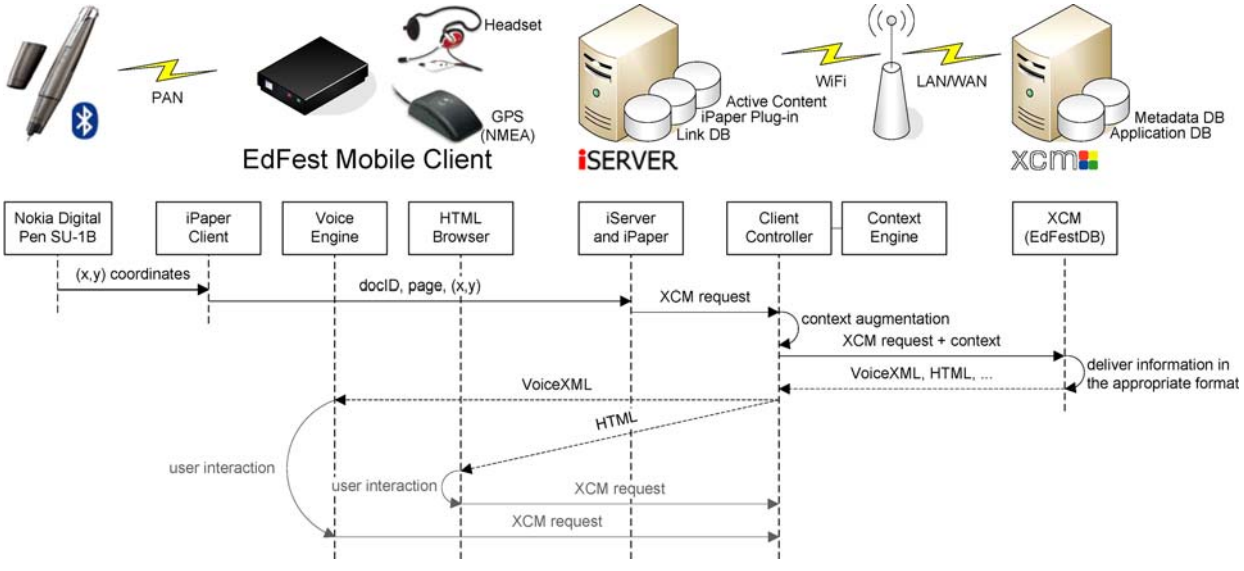

Fig. 5 EdFest architecture

active components can be used to bind areas on paper to arbitrary services as described later in Section 6. When iServer receives the request including the pen's (x, y) coordinates, it resolves the activated link to the appropriate URL-encoded request which is then sent to the XCM server. For example, a specific active area within the booklet might be mapped to `http://edfest.org/xcm?anchor=getRating`.

A special component, the Client Controller, is configured as a transparent proxy on the client-side between iServer and XCM. A core component of the Client Controller is the Context Engine that 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. It gathers all relevant information and updates the corresponding context elements such as the user's location, the protocol in use, language settings and the content type. An additional context factor allowed for is the set of users nearby. If some important changes in context happen, the Client Controller can contact the interested user using the information provided by the Context Engine. This functionality is used, for instance, to remind users about the start of events.

The XCM server manages the application database, which contains information about the events, performances, venues, users, etc. Additionally, in a special metadata database, XCM stores information about the definition of interfaces in terms of document structures and XSLT presentation templates. Document structures, content views and layout information are defined in terms of metadata objects that govern the standard publishing process of XCM. In the case of EdFest, XCM delivers VoiceXML files for the voice engine and different HTML pages for the stationary kiosk, pre- and post-visit desktop web browsing, and also for the possible use of a head-mounted display or PDA during the visit. When XCM receives a request from iServer, it uses the information contained in its databases to return an appropriate document according to the context that has been inserted by the Client Controller. This document is then delivered back 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. More details about the publishing process and our content publishing framework itself are given in Section 5.

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 iPaper 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 iPaper 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 iServer 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 Context Engine.

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 selection of an HTML link. Without the Client Controller, we would have a multi-channel interface where the channels are completely independent. For a full multi-modal 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 by pointing with the pen to a particular area in the paper brochure. In this case, the HTTP request is triggered by the iPaper plug-in for iServer 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 iPaper plug-in, the Client Controller sends back a default response for the content type of the iPaper plug-in 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 current 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 this is what the user wants in many cases, 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. Context information such as the GPS coordinates and information about the devices in use originates on the client side. The Context Engine keeps track of the context by gathering this information continuously either by polling all attached sensors or through directed updates that it receives from other components capable of delivering context. The Client Controller is also responsible for collecting this information and propagating it to the content publishing system on the server side. This can be done by augmenting the URL of the HTTP request with additional parameters.

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 XCM 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 delivers to the Context Engine which keeps track of the callback information and delivers it to all registered components by attaching callback information to each request. If a server-side component needs to send a notification to a user, it extracts the corresponding callback URL from the request 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 [8, 13, 14] 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 general content and web publishing integrated into the core model and system.

The resulting content publishing framework is the eXtensible Content Management system (XCM) [12]. As context-dependent content delivery is becoming more and more important in many fields including the field of mobile computing, XCM has been built to be context-aware by design. Two of its basic concepts are of particular importance for this aspect. The first of these concepts is a well-defined model of context and the second is the fact that in XCM all objects that need to be context-aware can have multiple variants, one for each context. Finally, context-dependent information delivery is then controlled by matching the context of a request to the variants of the objects involved.

The model of context that we use is similar to that proposed by others for HTML [28] and also semi-structured data [26]. Context is defined in terms of a set of (name, value) pairs which specify the various context dimensions to be taken into account in deciding on the content, structure, view and layout of a document. These dimensions can include anything from user-related properties such as language preference and location to system-related properties such as the request protocol and client device. The EdFest application allows for several context dimensions such as a user's identity, their current location, the time, the language setting and the protocol used.

As mentioned before, objects which are deemed to be context-sensitive can have multiple variants. Each variant has a set of characteristics also represented using (name, value) pairs that define the context in which it is appropriate. In contrast to context, where exact values for all dimensions have to be supplied, the values for characteristics can also include wildcards, sets and intervals. The syntax that was defined for these cases, as well as their precise interpretation, can be found in Table 1. As an example for such a characteristic value, consider a template object that is used to render a venue object in EdFest. Such an object has a variant for each output format that needs to be generated as represented by the format characteristic, e.g. `format=html`.

At the time of processing a request, XCM first extracts the context information from the request. Context information can be either inferred from the HTTP header or can be explicitly set in the query string of the request. When gathering all objects specified by the request, this context information is matched against the characteristics of all object variants

Table 1 Syntax for values of characteristic

Syntax	Definition
v	Unary value that matches a context value v_{ctx}, iff $v_{ctx} = v$.
$*$	Wildcard value that matches all context values v_{ctx}.
$v_1\{:v_n\}$	Set $S := \{v_1, \ldots, v_n\}$. A context value v_{ctx} matches, iff $v_{ctx} \in S$.
$v_{lower}..v_{upper}$	Interval $I := [v_{lower}, v_{upper}]$. A context value v_{ctx} matches, iff $v_{lower} \leq v_{ctx} \leq v_{upper}$.
$+$	Prefix indicating a *required* match.
$-$	Prefix indicating an *illegal* match.

involved and, for each object, the most appropriate variant is selected. The matching algorithm that selects the variants to be delivered to the client performs a best match rather than an exact match. This avoids having to specify an object variant for each possible combination of context dimensions. Hence, the algorithm computes a ranking value for each object variant by comparing all context dimensions specified in the request to those of each variant. If no variant reaches a ranking above a previously defined threshold, a specially denoted default variant that exists for each object is returned. However, in practice, this straightforward algorithm is not always enough to select an appropriate variant. There are ambiguous situations when more than one variant is ranked with the same maximum value. In this case, the algorithm can no longer determine which variant it should return and selecting one at random can lead to very undesirable results. To limit the number of such problematic situations, our algorithm further supports the notion of required and illegal characteristic values. As can be seen from the lower part of Table 1, a value that is prefixed with a plus sign needs to have been specified among the given context dimensions in order for a particular variant to be eligible. In contrast to this, a value prefixed with a minus sign will lead to a particular variant being discarded, if the same value has been specified for the corresponding context dimension. To achieve a maximum degree of expressiveness, these two prefixes can be combined with any of the syntax patterns described in the upper half of the table. Using such a prefix, we could refine the example of a template variant given above and characterise it with `format=+html` to express that this template is exclusively applicable when the context demands an HTML representation of the content.

Another strength of the matching algorithm is that it can also be used to implement certain application processes. In EdFest, for instance, the booking process allowing users to reserve tickets using the bookmark has been implemented entirely with variants without having to implement any application logic that couples the various phases of this process. As the information that is gathered by the reservation process can be seen as context, we modelled each step of the process as a variant that matches to a particular state within the gathering of information as shown in Fig. 6. Initially, when the process begins, no information has been entered by the user and thus the context is empty. Consequently the system displays the default variant that has been defined to match to the empty context. The default variant is a form that prompts the user to select an event and, as soon as this form is submitted, the context is updated with the additional value representing the identifier of the event. As the context has changed so does the variant that is delivered to the client, each guiding the user one step further along the booking process. Finally, when all required data has been gathered the reservation is stored and the user is informed about the booking number.

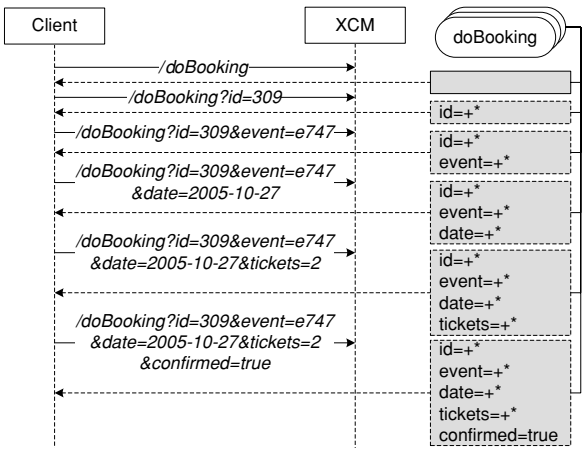

Fig. 6 Using variants to implement a booking process

XCM's model of context and its support for object variants allow multi-channel requirements to be implemented very elegantly. For the EdFest application, we have implemented three different channels: HTML, VoiceXML and PDF. The HTML channel is currently used in the kiosk application and 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 festival. But as we also aim to support pre- and post-visit activities, the HTML channel has been extended for activities such as the browsing of events or reviews and the composition of customised personal schedules. An important channel, especially for accessing dynamic information, is the voice channel that is used in the mobile client. Upon request that the user sends by interacting with pen and paper, XCM generates VoiceXML prompts from the data objects for the display of information. These prompts usually contain a version of the content that has been summarised and processed in a way that is graspable over the voice channel.

However, the main channel of the EdFest prototype is not the voice interface, but the paper brochure. This brochure is also generated by the XCM publishing framework. The system contains templates that generate an XML representation of the content that follows the desired structure of the brochure, i.e. a collection of event categories containing the actual events with all their associated information. This XML document is then processed by a special client component that, based on a set of templates, produces the final PDF document which can be printed and bound to a booklet. The Anoto pattern that is used by the Nokia Digital Pen has to be added manually to the generated PDF document. As we intended the booklet to look as similar as possible to the original brochure distributed by the Edinburgh Fringe Festival, we decided not to use XSL:FO [23] to generate the PDF document as was done previously for the 2004 EdFest system. The precision requirements of the layout for the new

booklet have surpassed the capabilities that XSL:FO has to offer at the moment as objects need to be placed at exact positions. As this publishing process is still rather complex, we are currently working on integrating these tasks into a single step, so that booklets can be printed directly from 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 link information, this database has to be updated upon creation of a new brochure by the content publishing system. For this reason, the client component that generates the PDF document for the brochure also keeps track of the positions of all interactive elements while rendering the content. This information is then used to produce an XML description of the link metadata for the paper brochure. The 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 content publishing framework to generate the printed documents is that we can also dynamically produce customised booklets. As mentioned above, this functionality to generate customised personal programmes has been implemented as a part of the kiosk application which uses the HTML channel of the system.

The three channels of the content 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 publishing 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 content publishing framework can only return one response for each 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 XCM 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) [19]. 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 objects. The iPaper plug-in that we implemented for interactive paper and applied in the EdFest system was developed in the context of a European project called Paper++. More details of the specific technologies developed within this project and the motivations behind the research in interactive paper can be found in [17, 25].

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 patterns 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 dynamically instantiated based on a configuration stored in the iServer metadata database and can be executed on the server as well as on the client side. The iPaper 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 iPaper 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 iServer as well as on the iPaper Client 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, the map and the bookmark while the actual festival data about venues, events, performances, ticket availability etc. are stored in the EdFest application database managed by XCM.

In the first scenario, we introduce the `XCMRequest` active component that runs on iServer and mainly acts as a "proxy" component for information that is accessible through XCM. The `XCMRequest` active component is typically used when the user points with the pen somewhere in the EdFest booklet, on the map or on the bookmark to get additional information in the form of audio output. The iPaper Client sends an HTTP request to iServer and the interactive paper plug-in resolves the positional information to the appropriate target resource as shown in Scenario 1 of Fig. 7. In the case that the information is stored in the EdFest application database, the resolved resource will be an `XCMRequest` active component. Based on the active component's identifier attribute, an object of the corresponding Java class is instantiated and initialised with the active component's supplementary metadata available from the iServer database. In the case of an `XCMRequest` component, this data includes an XCM query encoded as an HTTP request that can be sent to the content management system. The request is sent to the XCM content publishing platform which generates the corresponding response that is sent back to the `XCMRequest` component. Note that for the EdFest system we used a single database for managing information about the festival, but the concept of an active proxy component running on iServer 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 iPaper Client. While the user is browsing an interactive festival document, 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 one of the interactive paper documents which normally should result in a single request. Furthermore, the iPaper Client always checks if there is still an open 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 iPaper Client acoustically informs the user that another request is currently being processed. In this browsing mode, each pen event, after the described filtering process, results in a single request which is sent to the server component as outlined earlier in Fig. 5. However, for certain tasks, it makes sense that the iPaper Client processes multiple pen events before sending a request to iServer. This enhanced application logic of the iPaper Client can be realised by applying client-side active components. In the remainder of this section, we present a client-side active component running on the iPaper Client that was used within the EdFest demonstrator to capture the handwritten comments of users.

The lower part of Fig. 7 shows a second scenario, where a comment is captured based on the user's interaction with the festival booklet, involving a client-side active component. If a user starts to write on one of the empty comment pages at the end of the festival brochure that are defined as capture areas, the iPaper Client first sends a single event to iServer as is normally done in the browsing mode. The iPaper plug-in for iServer performs a lookup for the specific pen position and returns a `Capture` active component. An instance of the `Capture` active component is instantiated on the server side, based on the active component's configuration metadata stored in the iServer database. In the case of the `Capture` component, this information includes an upload address, i.e. a URL where the captured information finally should be uploaded using an HTTP POST request, as well as a timeout parameter which is used to terminate a capturing process without explicit user intervention as described later in this section. The `Capture` active component loaded on iServer sends an XML message, including the identifier attribute of the active component as well as various configuration parameters, back to the iPaper Client.

The iPaper Client receives the XML message and identifies it as an active component response. An instance of the corresponding `Capture` active component stub is instantiated based on the identifier of the active component XML message and the additional information stored within the message. During its initialisation phase, the `Capture` active component has to obtain information about the active region (selector) to which it is actually bound. Therefore, the active component running on the iPaper Client sends a special `getSelector` active component command to the iServer active component which then 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 iPaper Client 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` component is created, it asks the iPaper 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 iPaper Client switches from browsing mode to active mode which simply means that subsequent pen events are delegated to the active component running on the iPaper Client instead of directly being sent to the server.

As explained before, the `Capture` active component stub running on the iPaper Client requested information about the active region that was defined as an active capture area. 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` active 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` component

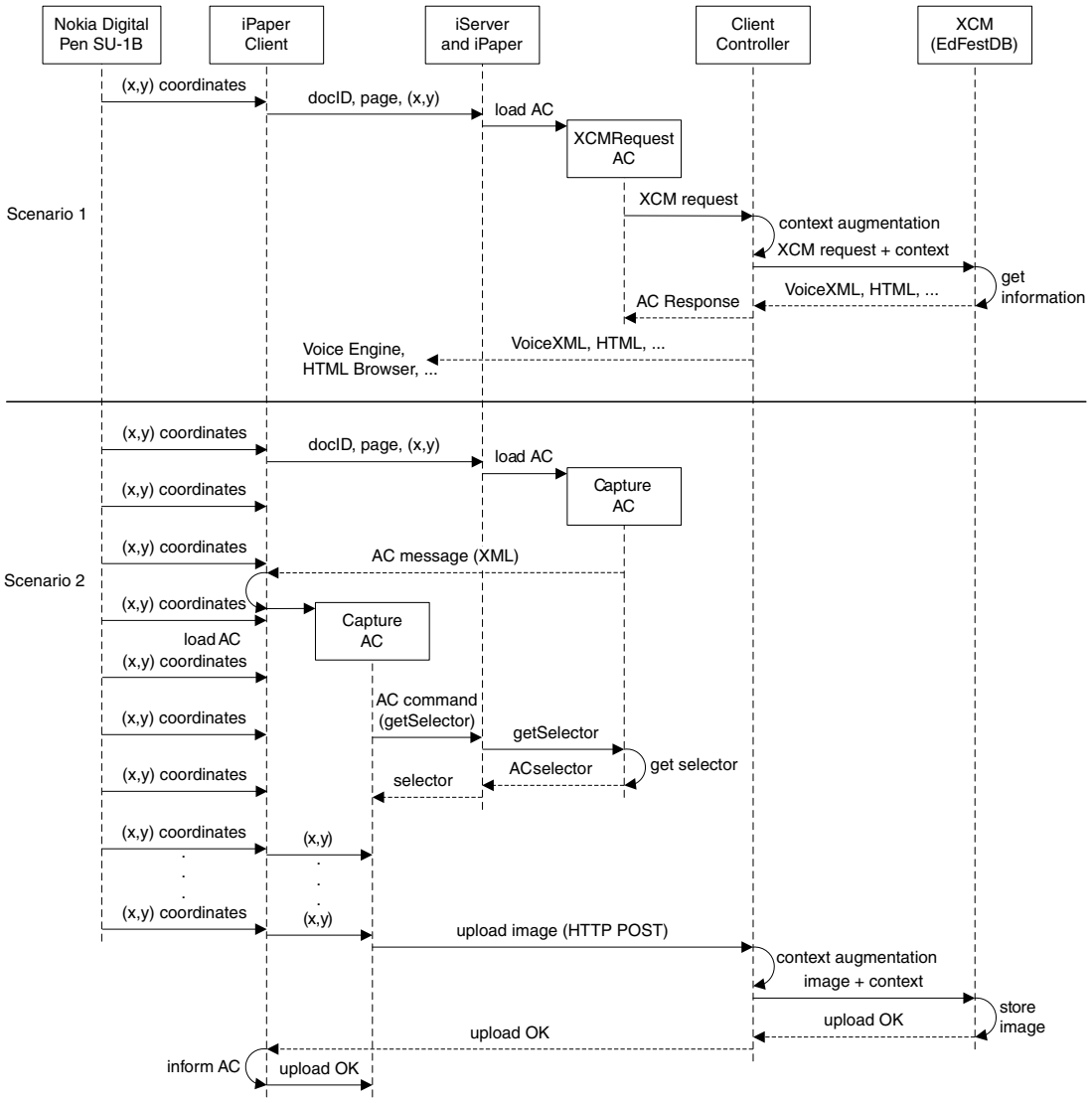

Fig. 7 iPaper Client and iServer active components

performs a lookup in the pen buffer to get all positional information 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 for the XCM server in a special XML format for representing notes consisting of one or more strokes defined by multiple points as well as additional metadata such as timing information or the pressure that was applied to the pen nib at a specific position. The XCM server stores the captured information in this neutral XML format which can easily be adapted to different output channels. To provide direct user feedback after a comment has been stored, XCM does some further processing on the captured data. In a first step, ICR is applied to the stroke information stored in the XML note. To achieve good handwriting recognition results without special training sessions, we used the MyScript Java development environment offered by Vision Objects [20]. The recognised text is embedded in a VoiceXML document and returned to the `Capture` active component running on the iPaper Client. Through this process, the user gets immediate feedback about the interpretation of his comment by the system. In addition, the Client Controller sends a response to the `Capture` active component confirming the successful upload of the comment. The active component informs the iPaper Client that it has finished its work and it is immediately unloaded by the iPaper Client which switches back to the default browsing mode. In the case that the system could not perform good handwriting recognition on a comment, the voice output informs the user that the comment has been stored by XCM but could not be transformed to a textual representation. However, such a comment may still be accessible from another output channel such as the HTML interface offered at the kiosks, where comments are rendered as JPEG images.

```
<?xml version="1.0" encoding="UTF-8"?>
<iserver>
<activeComponent id="capture_ac" creator="beat">
  <name>Capture a Comment</name>
  <identifier>CAPTURE_BOOKLET</identifier>
  <properties>
   <parameter>
    <key>org.iserver.ac:request</key>
    <value>mode=db&amp;
      anchor=setEventComment&amp;
      document=booklet&amp;
      require=event+user+format
    </value>
   </parameter>
   <parameter>
    <key>org.iserver.ac:timeout</key>
    <value>5000</value>
   </parameter>
   ...
  </properties>
</activeComponent>
</iserver>
```

Fig. 8 Active component specification

The active components can be defined directly in the database or they can be imported from an XML document. Figure 8 shows an XML specification of an active component with its main attributes. The `name` is used to find a specific active component whereas the `identifier` is applied to bind the definition of an active component to its related Java class. The example shows the definition of a capture active component where the first parameter with the `request` key defines the request parameters that have to be sent to the XCM server for uploading of the captured information. The timeout of the capture active component (`timeout` key) has been set to 5000 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-side active components (running on the iPaper Client) and server-side active components (running on iServer) which can communicate by sending special active component messages has proven 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 in this section 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. Based on the outcome of these user trials, the system was improved in various ways and a second user trial took place in August 2005. The usability trials in 2005 were carried out during a five-day period in various locations in the city, including public places as well as locations in and around festival venues including a fixed kiosk installed in the Fringe e-ticket tent. Our activities during the week involved general system testing of the EdFest prototype, a mix of semi-structured interviews and naturalistic observations of tourists using the system and also surveys based on user questionnaires. While it is beyond the scope of this paper to describe the user studies in detail, we include some general remarks on the outcomes in this section.

Test users mainly included visitors to festival venues as well as general tourists interviewed in public places. More than twenty sessions, each lasting about 30 minutes, were carried out with single users and pairs of festival visitors. 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. Figure 9 shows different users working with the interactive EdFest brochure, the map and the bookmark to fulfil a number of tasks.

(a) Collaboration (b) Browsing (c) Writing

Fig. 9 User trials at the Edinburgh festivals

In the interactive festival brochure that was used in the 2004 user trials, all texts were interactive and a user could access additional information by pointing to any printed part of the booklet. However, since paper-based access to digital information was new to most users, the user interface based on active parts of printed text was not clear to many of them. Therefore, for the 2005 version of the interactive festival system, we decided that special pictograms, as shown earlier in Fig. 4, should be used to link any additional digital information. All active pictograms have the same shape and colour making it very clear to a user where they should point with the pen. A first analysis of the 2005 EdFest user trials reveals that the pictogram-based user interface was much easier for users to learn and understand.

Another major change from the system used in 2004 to the current interactive festival guide in terms of the interaction design was the removal of the voice input channel. For many of the users in the 2004 trials it was unclear that they could also talk to the system in order to navigate through the voice dialogues. Even after learning that they had this possibility, some users went on using the pen for input in parallel to the voice input. In addition, some users expressed a reluctance about having to talk to the system in public places in order to get information or perform some task. As a result of the negative feedback concerning the voice input channel, we decided to remove this functionality from the 2005 prototype and use voice only as an output channel and design the interface so that all interaction could be controlled from the paper documents.

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 a writing device. Ideally, the pen itself should have a mechanism to switch between modes, for example by having a retractable writing stylus to indicate the switch from writing to interaction 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, the map and the bookmark 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. Furthermore, we got a lot of positive feedback for the new ticket booking functionality accessible from the interactive bookmark. Despite the very positive feedback from the user studies there is still great potential for experimenting with alternative interactive paper document designs for future versions of the system.

8 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 systems developed in 2004 and 2005 served as both a driving force for the design and development of the platform and major demonstrations 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. After two successive years of user trials at the festival, we now have a system that not only provides a lot of functionality, but also rates highly in terms of both usability and performance. The progress that has been made in a relatively short period of time is mainly due to the flexibility of the underlying platform and its support for rapid prototyping and experimentation. 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.

Acknowledgments We thank the other members of the EdFest team for their contributions to the project, including their help with the user trials in Edinburgh: Barbara Aeppli, Philipp Bolliger, Sandra Brockmann, Marco Dubacher, Tina Körner, Slavisa Maslic, Alexios Palinginis, Jan Rellermeyer, Christoph Schwank, Marco Steybe and Ljiljana Vukelja. We also thank the organisers of the Edinburgh Fringe Festival for their support.

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 (1997) 421–433.
2. Anoto AB, http://www.anoto.com

3. AT&T Natural Voices, AT&T Shannon Labs, New Jersey, USA, http://www.naturalvoices.att.com/
4. R. Belotti, C. Decurtins, M. Grossniklaus, M.C. Norrie and A. Palinginis, Interplay of content and context, Journal of Web Engineering 4(1) (2005) 57–78.
5. R. Belotti, C. Decurtins, M.C. Norrie, B. Signer and L. Vukelja, Experimental platform for mobile information systems, in: *Proceedings of MobiCom 2005, 11th Annual International Conference on Mobile Computing and Networking*, Cologne, Germany (August 2005) pp. 258–269.
6. B. Brown and M. Chalmers, Tourism and mobile technology, in: *Proceedings of ECSCW 2003, 8th European Conference on Computer Supported Cooperative Work*, Helsinki, Finland (September 2003) pp. 335–355.
7. B. Brown and E. Laurier, Designing electronic maps: An ethnographic approach, in: *Map Design for Mobile Applications* eds L. Meng, A. Zipf and T. Reichenberger. Springer Verlag (2004).
8. S. Ceri, P. Fraternali, A. Bongio, M. Brambilla, S. Comai and M. Matera, *Designing Data-Intensive Web Applications*, The Morgan Kaufmann Series in Data Management Systems. (Morgan Kaufmann, 2002).
9. K. Cheverst, N. Davies, K. Mitchell, A. Friday and C. Efstratiou, Developing a context-aware electronic tourist guide: Some issues and experiences, in: *Proceedings of CHI 2000, ACM Conference on Human Factors in Computing Systems*, The Hague, The Netherlands (April 2000) pp. 17–24.
10. M.J. Franklin, Challenges in ubiquitous data management, in: *Lecture Notes In Computer Science: Informatics - 10 Years Back. 10 Years Ahead*, Springer-Verlag, vol. 2000 (2001) pp. 24–33.
11. A. Grasso, A. Karsenty and M. Susani, Augmenting paper to enhance community information sharing, in: *Proceedings of DARE 2000, Conference on Designing Augmented Reality Environments*, Elsinore, Denmark (April 2000) pp. 51–62.
12. M. Grossniklaus and M.C. Norrie, Information concepts for content management, in: *Proceedings of DASWIS 2002, International Workshop on Data Semantics in Web Information Systems*, Singapore, Republic of Singapore (December 2002) pp. 150–159.
13. G.-J. Houben, P. Barna, F. Frasincar and R. Vdovjak, Hera: Development of semantic web information systems, in: *Proceedings of ICWE '03, 3rd International Conference on Web Engineering* (July 2003) pp. 529–538.
14. G. Kappel, W. Retschitzegger and W. Schwinger, Modeling customizable web applications—a requirement's perspective, in: *Kyoto International Conference on Digital Libraries*, Kyoto, Japan (November 2000) pp. 168–179.
15. A. Kobler, M.C. Norrie and A. Würgler, OMS Approach to database development through rapid prototyping, in: *Proceedings of WITS '98, 8th Workshop on Information Technologies and Systems*, Helsinki, Finland (December 1998).
16. D.M. Levy, *Scrolling Forward: Making Sense of Documents in the Digital Age*, (Arcade Publishing, October 2001).
17. P. Luff, C. Heath, M.C. Norrie, B. Signer and P. Herdman, Only touching the surface: Creating affinities between digital content and paper, in: *Proceedings of CSCW 2004*, Chicago, USA (November 2004) pp. 523–532.
18. C.C. Marshall, Annotation: From paper books to digital library, in: *Proceedings of DL '97, 2nd ACM International Conference on Digital Libraries*, Philadelphia, USA (July 1997) pp. 131–140.
19. W3C Interaction Domain: Multimodal Interaction Activity, http://www.w3.org/2002/mmi/
20. MyScript Handwriting Recognition Software, Vision Objects, http://www.vision-objects.com/
21. M.C. Norrie, An extended entity-relationship approach to data management in object-oriented systems, in: *Proceedings of ER '93, 12th International Conference on the Entity-Relationship Approach*, Arlington, USA (December 1993) pp. 390–401.
22. A. Pashtan, R. Blattler, A. Heusser and P. Scheuermann, CATIS: A context-aware tourist information system, in: *Proceedings of IMC 2003, 4th International Workshop of Mobile Computing*, Rostock, Germany (June 2003).
23. D. Pawson. *XSL-FO: Making XML Look Good in Print*, O'Reilly & Associates (August 2002).
24. A.J. Sellen and R. Harper, *The Myth of the Paperless Office*, MIT Press (November 2001).
25. B. Signer and M.C. Norrie, A framework for cross-media information management, in: *Proceedings of EuroIMSA 2005, International Conference on Internet and Multimedia Systems and Applications*, Grindelwald, Switzerland (February 2005) pp. 318–323.
26. Y. Stavrakas and M. Gergatsoulis, Multidimensional semistructured data: representing context-dependent information on the web, in: *Proceedings of CAiSE 2002, 14th Conference on Advanced Information Systems Engineering*, Toronto, Canada (June 2002) pp. 183–199.
27. Vindigo City Guide, http://www.vindigo.com/
28. W.W. Wadge, G. Brown, M.C. Schraefel and T. Yildrim, Intensional HTML, in: *Proceedings of PODDP '98, 4th International Workshop on Principles of Digital Document Processing*, Saint Malo, France (March 1998) pp. 128–139.
29. A. Woodruff, P. Aoki, A. Hurst and M. Szymanski, Electronic guidebooks and visitor attention, in: *Proceedings of ICHIM 2001, 6th International Cultural Heritage Informatics Meeting*, Milan, Italy (September 2001) pp. 437–454.

Moira C. Norrie is a Professor at ETH Zurich where she is head of the Institute for Information Systems and leads the Global Information Systems research group. Her research interests include object-oriented models and systems for data management, web engineering, mobile and personal information systems and interactive paper as a medium for integrating printed and digital information.

Beat Signer is a Post-Doctoral researcher in the Global Information Systems research group at ETH Zurich. He received a Ph.D. from ETH Zurich in 2005 for his work investigating fundamental concepts for interactive paper and cross-media information management. His research interests include interactive paper, cross-media information management, object-oriented technologies and software engineering.

Michael Grossniklaus is a research assistant in the Global Information Systems research group at ETH Zurich. He received a Diploma (M.Sc.) in Computer Science from ETH Zurich in 2001 and is currently completing his Ph.D. His main research interest is empowering information systems for context-aware data management and delivery in the domain of web engineering and mobile computing.

Corsin Decurtins is a research assistant in the Global Information Systems research group at ETH Zurich. He received a Diploma (M.Sc.) in Computer Science from ETH Zurich in 2002. His research focusses on model-based approaches and infrastructure for ubiquitous and mobile information environments. In addition to his Ph.D. Corsin also works part-time as a senior software engineer at the software company Netcetera.

Rudi Belotti was a research assistant in the Global Information Systems research group at ETH Zurich from 2004–2006. He received a Diploma (M.Sc.) in Computer Science from ETH Zurich in 2004. In his research, he developed a general model and engine for the management of context information in mobile information systems. He is currently working for an e-business services company in Ticino, Switzerland.

Nadir Weibel is a research assistant in the Global Information Systems research group at ETH Zurich. He received a Diploma (M.Sc.) in Computer Science from ETH Zurich in 2003 and is currently working on his Ph.D. His research is in the area of interactive paper, particularly on the authoring and publishing infrastructure for interactive documents as well as issues of human computer interaction and mobile environments.