Zuerst ersch. in : World Wide Web ; 10 (2007), 4. - S. 529-556
http://dx.doi.org/10.1007/s11280-007-0026-9

Interactive Paper as a Mobile Client for a Multi-channel Web Information System

Beat Signer · Michael Grossniklaus · Moira C. Norrie

Abstract We describe how interactive paper can be used together with a multi-channel web information system to build a platform for experimenting with multi-modal context-aware mobile information services. As an application, we present a tourist guide for visitors to an international festival that was developed to investigate alternative modes of information delivery and interaction in mobile environments. The guide is based around a set of interactive paper documents—an event brochure, map and bookmark. The brochure and map are augmented with digital services by using a digital pen to activate links and a text-to-speech engine for information delivery. The digital pen is also used for data capture of event ratings and reviews. The bookmark provides access to advanced searches and ticket reservations. We describe the architecture and operation of the system, highlighting the challenges of extending a web information system to support both the generation of the paper documents and the interaction from these documents, alongside more traditional access channels. Finally, we discuss the range of context-aware interactions that is supported by our platform.

Keywords interactive paper · multi-channel web information systems · mobile applications · context awareness · multi-modal user interfaces

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

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

M. C. Norrie
e-mail: norrie@inf.ethz.ch

1 Introduction

One of the major challenges of mobile data management is how to support interaction for users on the move. Most research projects focus on small mobile devices such as phones and PDAs, and yet it is accepted that these are less than ideal for information systems of any complexity. Restricted screen size makes it difficult to compare and combine information and to support various forms of collaboration. Current screen technologies also make it difficult to view information in outdoor environments, but this is something that new screen technologies should overcome.

Adaptivity and context-awareness have been recognised as key requirements in supporting mobility [12] by ensuring that the right information is delivered to the right user at the right time. They reduce the information bandwidth and provide a focal data view and this is certainly important when presenting data in a mobile environment, not only due to communication costs and restrictions of client devices, but also the fact that users may be involved in parallel activities or need to react quickly to a given situation. However, these alone are rarely sufficient to overcome the problems of using mobile devices such as PDAs. Some projects have chosen to use devices with larger screens such as Tablet PCs to enable them to present various information streams alongside each other. For example, in a project to support collaboration among tourists visiting a city, Tablet PCs were used to display maps, photos and textual descriptions of objects of interest and also recommender information [6]. While successful studies were undertaken with recruited test users, Tablet PCs are heavy and require the use of both hands and therefore not appropriate for real tourists.

As a consequence, we decided to investigate the possible use of other technologies and modes of interaction for mobile information systems. In addition to speech-only interfaces, we have carried out experiments with interfaces based on new technologies for interactive paper documents coupled with speech output, thereby integrating electronic information resources with the benefits of paper maps as suggested in [23]. These interfaces are offered in parallel to more traditional web browser interfaces running on desktop PCs or PDAs. In general there is a need for platforms enabling the experimentation with interesting new forms of interaction and multi-modal interfaces. In this paper, we present an architecture that combines interactive paper, speech output and a multi-channel web information system into a platform for mobile systems. As an application, we describe a mobile tourist guide for a large international arts festival and discuss the requirements imposed on web information systems by such applications. We then describe how we addressed these challenges using an extended web information system based on an object-oriented database that supports multi-channel access and context-awareness to implement the tourist guide.

We start in Section 2 with a brief motivation for interactive paper and the technologies on which we based our system. The set of interactive documents used in our tourist guide application and the functionality that they offer is described in Section 3 where we also highlight some of the major design issues. An overview of the workflow involved in the generation and operation of such an application and the major system components is provided in Section 4, whereas Section 5 describes the main technical requirements and challenges in terms of the web information and data management system. Section 6 details the support for context-aware interaction.

Section 7 provides an initial evaluation of the platform and also the tourist guide application. Concluding remarks are given in Section 8.

2 Interactive paper

Tourism is a domain with considerable potential for the use of mobile technologies and has been the focus of many research projects investigating everything from mobile transactions to context-aware services and personalisation. Commercial PDA-based guides for tourists are now available and have had some limited success, but still paper maps and guidebooks remain as the essential tourist accessories. Tourists spend a lot of time comparing and combining information. It is therefore important that they can easily switch back and forth between options and display related information such as maps and descriptions of historical sites or events simultaneously. Working with paper documents such as maps and guidebooks, users make extensive use of bookmarks and annotation, even using fingers as temporary bookmarks to keep track of places within a book. Folded maps and tickets are other examples of bookmarks commonly used in guidebooks. It is also important to recognise that tourism is primarily a social activity and the interaction with fellow travellers and locals in navigating around a city and planning activities is a major part of the experience. Studies show interesting ways in which tourists work with multiple documents, often distributing the tasks of finding information and using actions of pointing and positioning of documents to relate information items to each other and also to their environment [5]. Maps are used in many ways and not just for the task of route planning. They also provide users with an overview of the city, emphasising major features and spatial relationships between areas, and helping them to identify possible areas of interest and keep track of places visited.

Paper has many affordances that make it attractive in mobile environments. It is light, cheap, robust and requires no power. It can be easily annotated in various ways. It can be folded and torn, increasing the ease of taking particular items of interest with you. It is easy to read in dim and bright environments and from different angles. Clearly it also has limitations in that it is a static representation of information and the presentation favours only certain forms of searching (for example, alphabetical listings based on names of places or events). The challenge is whether it is possible to merge the paper and digital worlds thereby gaining the best of both. The idea of interactive paper documents is to present certain key static information on paper and then to enable users to activate digital services from paper for access to information that may be supplementary, dynamic or context-dependent. Other forms of digital services may enable them to carry out transactions such as reserving tickets or entering reviews.

There are numerous research projects and commercial products that link paper to digital services based on some means of detecting user actions on paper. A detailed survey of technologies for integrating paper and digital information and a discussion of related research projects can be found in [25]. In the DigitalDesk project at Xerox EuroPARC [27], a video camera mounted above an office desk is used to detect user actions on documents. It is an example where the means of detection is related to the physical environment rather than the paper itself. For mobile environments, it is more appropriate to use a technology where either link or position information

is encoded on paper. Barcodes or special icons could be used to encode unique identifiers that are linked to web pages or digital files to be displayed as is done, for example, in WebStickers [19]. Another approach is to encode position information across pages enabling the pen position to be tracked. The advantage of this approach is that it can be used to capture writing as well as for interaction. Further, since links are based on relative position within a document rather than simple identifiers, it is possible to find all the elements within a document that link to a specific target. Anoto's digital pen and paper functionality is based on such an approach: A special digital pen has a camera situated alongside the writing stylus to capture images of an almost invisible pattern of infrared absorbing black dots printed on arbitrary paper documents, as indicated in Figure 1.

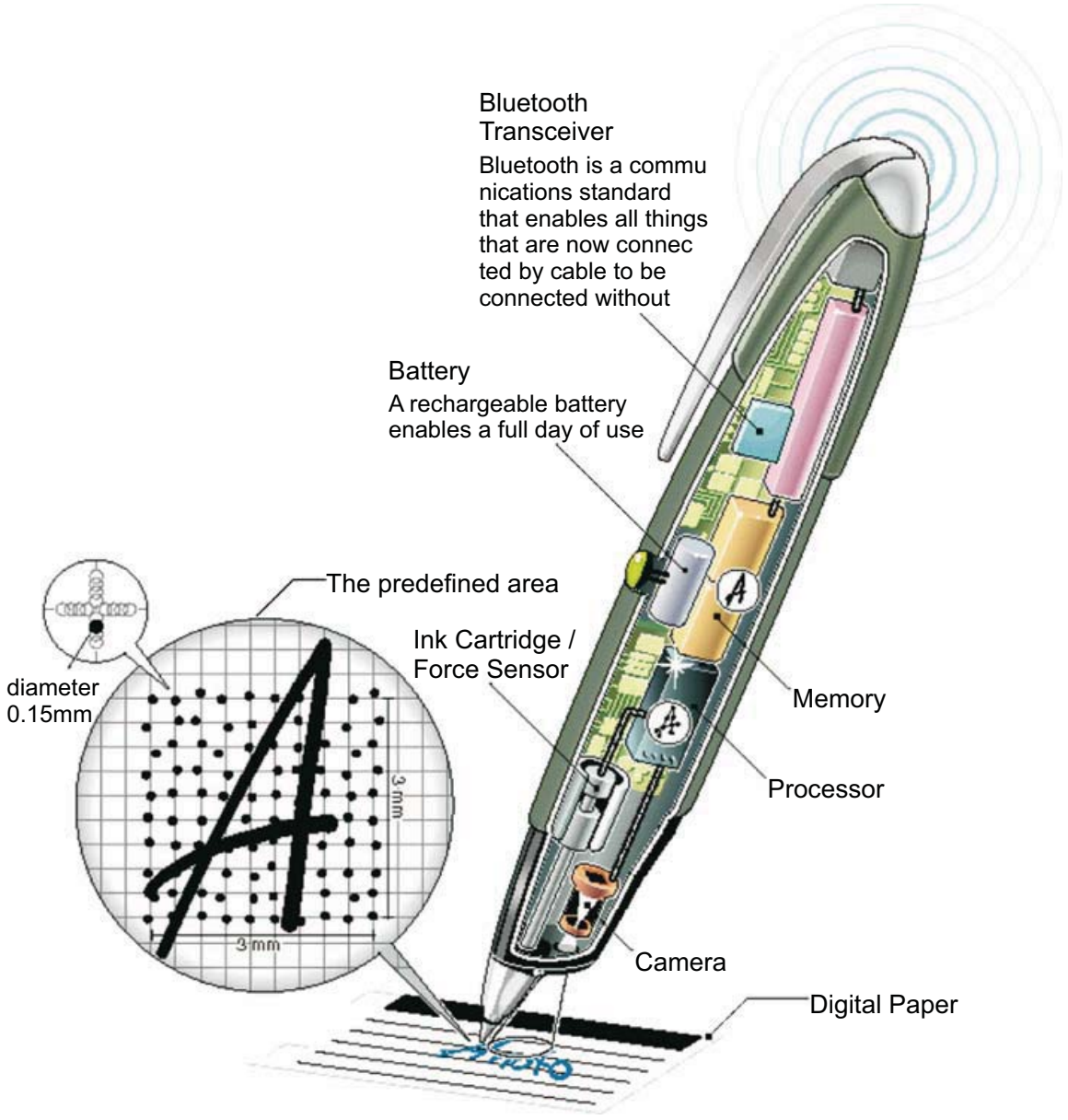


Figure 1 Anoto technology.

The pattern of dots encodes (x,y) positions in a vast virtual document space. Camera images are recorded and processed in real-time giving up to one hundred (x,y) positions per second. The technology was developed for the digital capture of handwriting and several pages of handwriting can be captured and stored within the pen before being transmitted to a PC. While data transmission on demand is sufficient for writing capture, immediate transfer of position data is required if the digital pen is to be used as an interaction device. We have access to a prototype version of the Nokia Digital Pen [21] modified by the Anoto engineers which, in contrast to the commercial version, works in a streaming mode, sending position data immediately and continuously to the computer. This enables us to use the Nokia Digital Pen as an interaction device as well as for writing capture. More recently we have integrated Maxell's Magicomm G303 [20], the first commercially available digital pen that offers streaming functionality based on Anoto technology.

While technologies to track the movement of a pen on paper provide the basis for realising interactive paper documents, a server is required that can interpret the position information and invoke the appropriate digital services. Many of the projects dealing with an integration of paper and digital information tend to have developed simple servers dedicated to a specific application or set of applications. Our goal was to develop a general server platform that could be used to support any type of application and enable both writing capture and interaction. The key to achieving this was to develop a general cross-media link server, called iServer. To ensure full flexibility in terms of resources, a plug-in mechanism is used to support specific media types and links can be defined between resources or selected elements within resources. In Figure 2, we show a simplified version of the iServer link model containing only the main concepts such as links, selectors and resources. The `Selectors` representing elements within a resource in combination with the `Resources`, which represent entire resources, are the central components for the

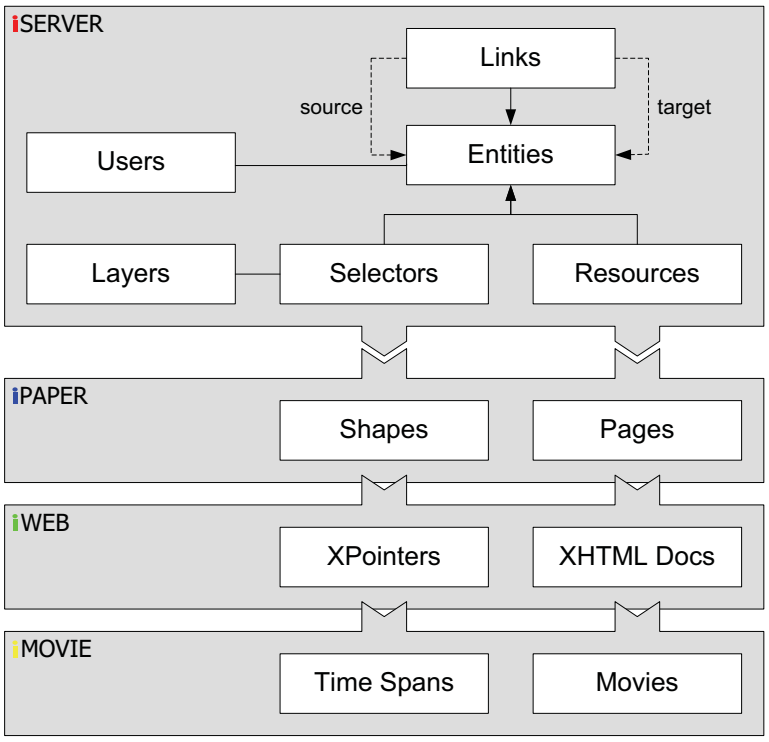


Figure 2 iServer with resource plug-ins.

resource plug-in mechanism. For a particular media type, we can extend the iServer platform by introducing a component that defines selectors and resources for that media type.

To support interactive paper, we developed an iPaper plug-in that uses geometrical shapes within pages as selectors to define active areas in paper documents as shown in Figure 2. Other plug-ins have been developed for integrating movies and XHTML documents. In the case of the movie plug-in (iMovie), a resource is defined by a movie and the corresponding selectors are represented by time spans. The iWeb plug-in deals with web pages where a resource is given by an XHTML document and XPointer expressions [26] are used to select parts of a web page. Furthermore, to create links to dynamic application information stored in a database, plug-ins can be used where the resource is a database and a selector is a database query.

It is out of the scope of this paper to give a detailed description of each resource plug-in. However, it is important to note that a major advantage of the chosen plug-in mechanism is the tight integration over all types of media based on a common core link model rather than isolated applications for specific kinds of media. As soon as a plug-in for a new resource type has been implemented, entities of that medium can be cross-linked with instances of any existing media type and the new input/output channel can be used to interact with existing resources. With each resource plug-in introduced, the iServer platform becomes more powerful and provides a richer cross-media information space.

The iServer link model is a generalisation of link models found in hypermedia systems [2], including that of the XML standard, XLink. This means that we can support concepts such as multi-target and multi-source links, links over links and also link annotations. In addition, we introduced a layer concept to support nested selectors—or nested active areas in the case of the interactive paper plug-in. Layers can be activated, deactivated and re-ordered dynamically enabling layers to be used to provide context-dependent access. For example, we developed an interactive map application where different layers provide different categories of information about a city, such as cultural, entertainment or shopping. Users could then customise their personal interactive map by selectively activating or deactivating specific layers. Furthermore, the iServer model provides a user management component to control link access rights and specify different services to be provided to different categories of users depending on their roles and access rights.

Details of iServer, including the link metamodel, can be found in [25]. For the scope of this paper, we can consider iServer, and specifically the iPaper plug-in, as providing the basis for integrating paper as an input channel in an application system. In the case of our mobile tourist guide, it is responsible for detecting user selections of active areas on paper and mapping these to requests sent to a general content publishing system which then executes the appropriate services and delivers information on the appropriate output channel (e.g. speech output).

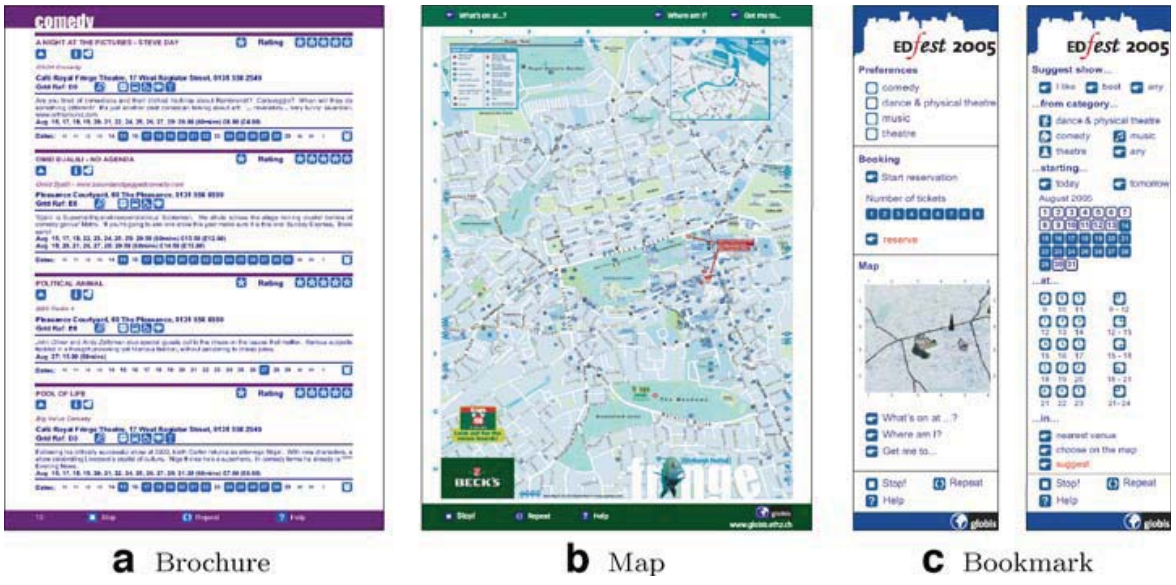
3 Design and functionality of the tourist guide

The Edinburgh Festival Fringe is the world's largest international arts festival with around 260 venues, 1,800 events and 28,000 performances in 2006. With so many

events on offer, tourists often select events during their visit based on contextual factors such as location, time and weather as well as ticket availability and reviews. During the Fringe, there are many sources of printed information for tourists to be found throughout the city of Edinburgh. An official Fringe brochure gives listings of all events organised under categories such as comedy, theatre and dance and an official map shows all venues. Newspapers publish daily listings of events and also reviews. Advertisement materials such as posters and flyers appear all over the city and often copies of reviews will be attached to these. It is therefore an ideal environment in which to experiment with new forms of interfaces based on interactive paper.

We based our system, EdFest, around three interactive paper documents—a brochure, a map and a bookmark as shown in Figure 3. All three were designed so they served some useful purpose without the digital system (even if only to serve as a physical bookmark by placing it between the pages of some document). With the benefits of the digital system, they could each provide some functionality on their own and therefore be used independently, but there are also some services that require more than one of these paper documents.

The design of the interactive brochure was based on that of the official Fringe brochure where the organisers provide a compact summary of information about an event, including the dates and times of performances and the cost of the tickets as shown in Figure 4a. Events are listed alphabetically according to title under sections for the separate categories. In our interactive brochure, which is shown in Figure 4b, we included pictograms as active areas which, when activated by touching them with the pen, provide supplementary information from our festival database through a visual or voice output channel. Examples of such information include descriptions of bar and catering facilities on offer at the event venue, warnings about the use of bad language or nudity, bus information for getting to the venue and information about disabled access.



a Brochure **b** Map **c** Bookmark

Figure 3 Interactive paper documents.

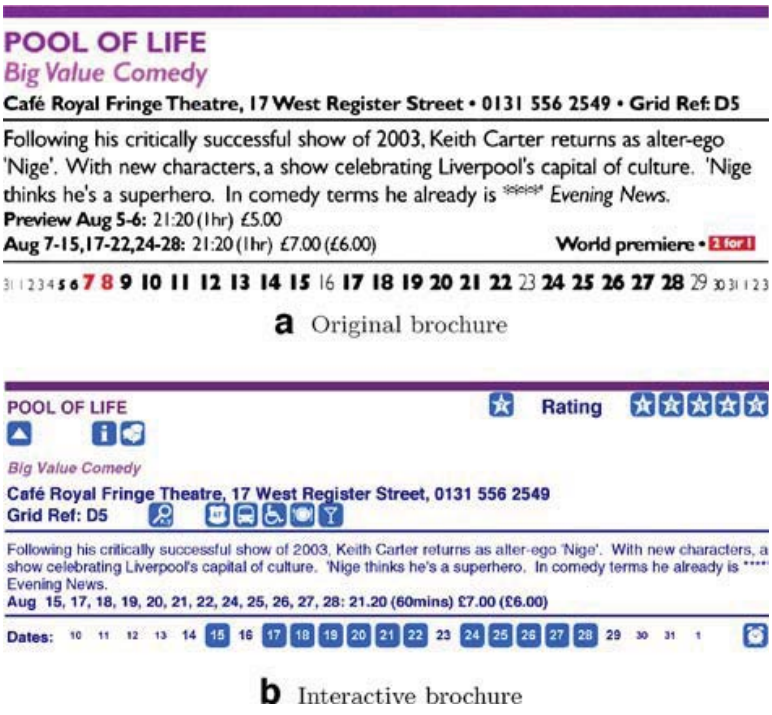


Figure 4 Festival brochure event entry.

At the bottom of each event entry, the official brochure gives a timeline view of the festival showing the dates of event performances in bold typeface. We replaced the bold entries with pictograms which, when selected, give information about ticket availability on these days. To the right of this timeline is an alarm clock pictogram which enables registered users to be sent a reminder by SMS 30 min before the start time of the event. This is an example where the user selects a pictogram and then will be asked in a second step to select a date from the timeline in order to choose a specific performance and complete the transaction. One major design issue was how to handle such cases where the system is waiting for a user to complete a transaction that they may never complete, especially as the functionality of the pictograms can be context-dependent. The reminder example is such a context-dependent transaction. If a user selects a date pictogram without first choosing the alarm clock pictogram, they will get information about the number of tickets still available for the specified date. We simply use timeouts to abort a transaction and return the system to the default interaction mode rather than prompt them with further messages. It is important to note that the design of an interface based on speech output is more challenging than that of visual interfaces as users become much more easily irritated and confused if too much information is given or there is too much repetition. All of the documents have ‘repeat’, ‘stop’ and ‘help’ buttons at the bottom of every page to give users control over the speech output and also to obtain context-dependent help information when required.

Reviews play an important role in the selection of events to attend and the Fringe festival organisers have tried many ways to encourage visitors to submit either ratings or reviews. While this functionality is offered on their web site, it tends to be a relatively small group of visitors who input information. In 2005, they also provided a system where users could submit ratings by SMS but this was not as popular as they had expected. A key feature of our system was therefore to provide an easy

means for users to input and access ratings and reviews. At the top right of each event entry, there is a rating section where users can access the current average rating by touching the '?' pictogram to the left of the rating label and input their own rating by selecting a '1–5' to the right of the label. They can also submit their own comments on events by writing them in the blank comments pages provided at the back of the brochure and then linking them to an event in the brochure by selecting the 'up-arrow' pictogram below the event heading. Note that a comment is always stored in the form of an XML document that contains a list of strokes. Intelligent Character Recognition (ICR) software is then used to analyse the handwritten text and, if the confidence level is high, the text is read back to the user via the text-to-speech engine. If the comment cannot be converted to text with a high level of confidence, it can still be retrieved in image form and can be viewed only via a visual interface such as the usual web browser interfaces accessible at festival kiosks or at home on desktop PCs. Handwritten comments which are successfully converted to text can be accessed by other users by selecting the 'bubble speech' pictogram under the event heading. Alternatively, users can write their comments on Post-Its with Anoto pattern which has the advantage that they can then easily annotate the event entries by placing the Post-Its in the brochure at the corresponding place.

The map can of course be used independently but also provides users with services to help them locate themselves and find out information about venues. By touching anywhere on the map with the pen, information will be given about the nearest venue. A 'Where am I?' option tells the user where they are located in terms of a map grid reference e.g. "Grid position E4 lower left corner". By placing the pen on the map, the system will direct the user towards the exact position with commands such as "move the pen a little to the right". This map locator functionality can also be used to find the location of events by selecting the locator pictogram given in the brochure event listings [15]. This is an example of an operation that provides a link from one document to another. Other options are to find the next event starting at a venue and to get navigation information. Both of these are available by selecting the corresponding pictogram at the top of the map and then selecting the location on the map.

Given the number of events, the brochure is better suited to the lookup of particular events or detailed study rather than being able to find possible events of interest happening at a particular place or time. This functionality is better supported by the daily event listing in newspapers which are listed by time, but even these are several pages long and it is difficult to find something nearby. We therefore decided to provide advanced search facilities through the design of a general bookmark document. One side of the bookmark allows a user to formulate an event search by specifying parameters such as category, date, time and location as shown in Figure 5. The dates and times can be specified in different ways to simplify the interface. For example, the date can be specified as 'today' or as a specific date from the calendar. The times can be specific hours or periods of the day. The location can either be specified by selecting a position on the map or by the user's current location. The search serves as a recommender system as it returns the first of all events satisfying the specified search query according to a decision criteria specified by the user—the best match to user preferences, the most highly-rated or a random pick.

The other side of the bookmark shown in Figure 3c provides a list of preferences that the user can set, a booking service and a mini-map which provides the same

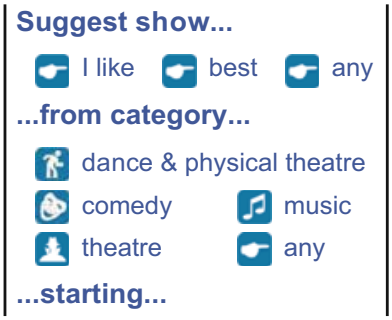


Figure 5 Bookmark search functionality.

functionality as the actual map based on a schematic view of the city showing main streets and landmarks used for orientation and navigation. The ticket booking system is a multi-step transaction started by selecting the 'start reservation' pictogram on the bookmark. The user is then asked to select an event from the brochure and then a performance date for that event. They then select the number of tickets on the bookmark and the system responds with a summary of the booking information. The user completes the transaction by selecting the 'reserve' pictogram and the system gives a confirmation message that includes a reservation number. This can be repeated by selecting the 'repeat' pictogram described previously.

4 Architecture and workflow

A major challenge was how to generate the interactive paper documents from the web information system which managed both the festival data and the publishing data that provided access to information via standard web channels such as web browsers on desktop PCs or PDAs. Thus the web information system should be responsible for both the publishing of information as PDF documents with defined active areas, and the publishing of information in a text-to-speech channel in response to requests generated by activating areas within a printed document using a digital pen.

As can be seen from our description of the functionality of the system, the documents to be produced contain a large number of active areas, each of which should have a unique identifier. The three main issues to be addressed were:

- How to map the dynamic digital information to a static medium such as paper?
- How to connect information in the content management system with the printed material?
- How to link the printed information back to the digital information stored in our system?

The interactive paper publication process shown in Figure 6 outlines our approach. The publishing process is mainly based on three components: the eXtensible Content Management (XCM) system that we developed, iServer which maps interactions on paper to requests to XCM and a third component iPublish is responsible for publishing the paper documents based on information stored in XCM.

The first step is to structure the festival data managed by XCM in a format suitable for publication as a paper document. We defined a document model based on the concept of chapters, sections, sentences, words, tables and images. This enabled us to

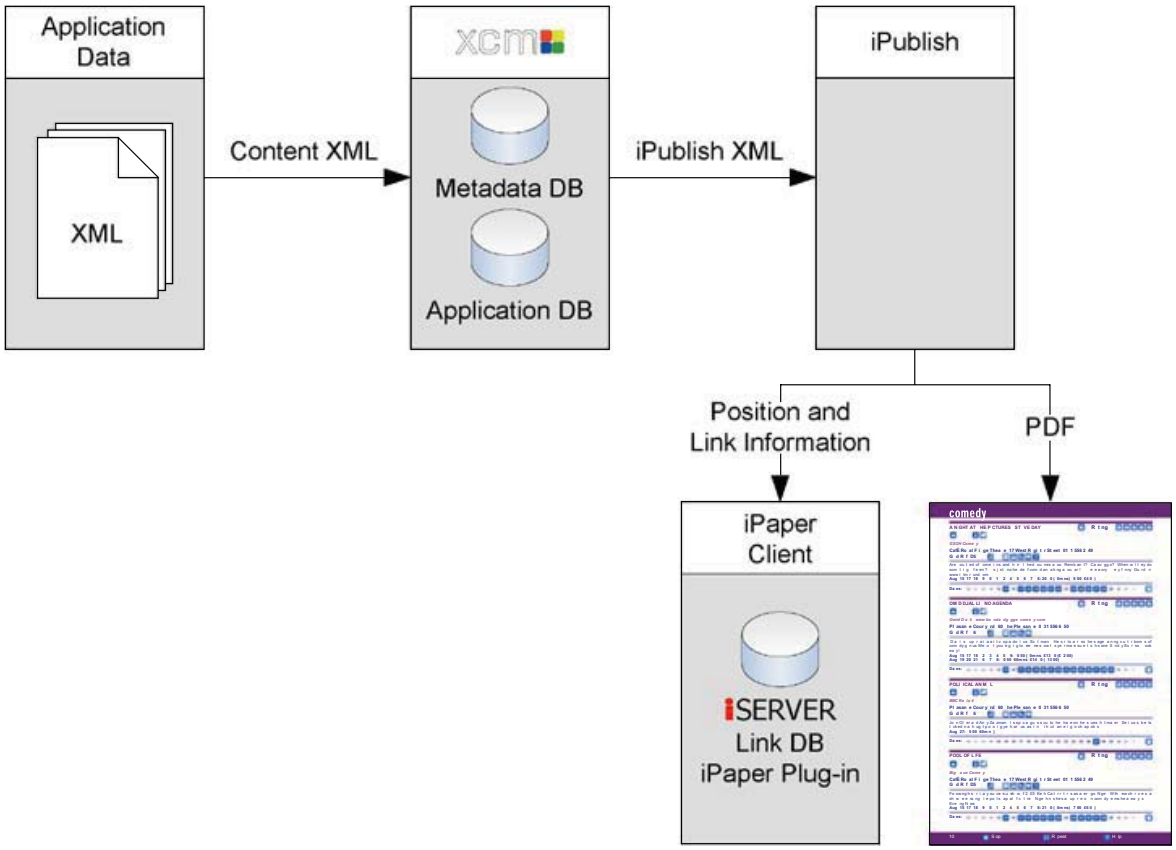


Figure 6 Publishing process.

structure and link the information at different levels of granularity. The document content and structure is represented as an iPublish XML document as shown in Figure 7.

A tagging mechanism is used to specify the active elements in a document which will be linked to digital services by iServer via its iPaper plug-in. To be able to export this information to iServer, we introduced a second tag which describes the

Figure 7 Document publishing.

```
<?xml version="1.0" encoding="UTF-8"?>
<document>
<chapter id="c-comedy">
  <body>
   <section id="event.e-51">
    <paragraph>
     <text id="event.title.e-51">
       POOL OF LIFE
     </text>
     <image ilink="true" ac="get_rating?event=e-51"
       id="get_rating.e-51"src="get_rating.png"/>
    </paragraph>
    ...
   </section>
   ...
  </body>
</chapter>
...
</document>
```

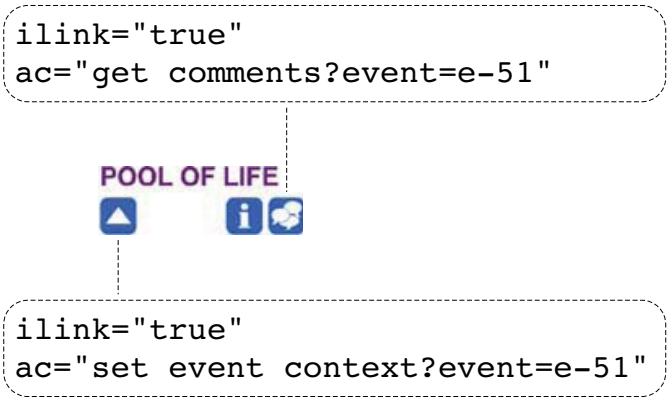


Figure 8 Tagging mechanism.

functionality to be linked and assigns a unique identifier to this particular piece of information. An example of such a tagging mechanism is shown in Figure 8.

Printing is another critical process as we want to be able to map a particular position on paper to the right logical active area. The problem is that the exact position where something is printed can be affected by several factors such as the text editor used to write the document, the printer used and the paper size [16]. To enable us to map positions in a reliable way, we needed to create the link information based on the actual format to be printed rather than the logical format. We do this by first generating the PDF document based on the iPublish XML data received from XCM and then analysing it to determine the precise positions of active areas. The iPublish platform exports all link data in the iServer-specific XML link representation format which allows the specification of documents, pages, layers, various geometrical shapes and links between these shapes and digital objects. This XML data is imported by the iServer platform and made persistent in its link database.

In the next step, the different document pages have to be registered with the Anoto pattern in such a way that a specific global Anoto position can later be assigned to a position on a single document page. To easily cover successive pages with parts of the Anoto address space without having to deal with individual pages, we implemented a document handler for multiple pages.

The idea is to layout successive pages defined in the augmented paper address space within the Anoto encoding space in a grid of a number of columns and a number of rows as shown in Figure 9. Only the upper left corner of the first page (p_{org}) has to be registered with the corresponding global position in the Anoto space. All the pages of a single document have the same width (*pageWidth*) and height (*pageHeight*). Optionally, a gap can be defined between pages in the grid on the horizontal as well as on the vertical axis represented by Δx and Δy, respectively. Note that these gaps in the Anoto address space occur since, for each page, there is a reserved area to define special Anoto functionality such as paper-based buttons. In contrast to our solution, an application based directly on Anoto technology does not always cover a page with a unique pattern. For example, for some paper-based buttons, a section of the specially reserved pattern is pasted over the base pattern covering the page. The Anoto solution not only requires additional pattern space for active areas, but also puts constraints on the minimal distance between adjacent paper-based buttons.

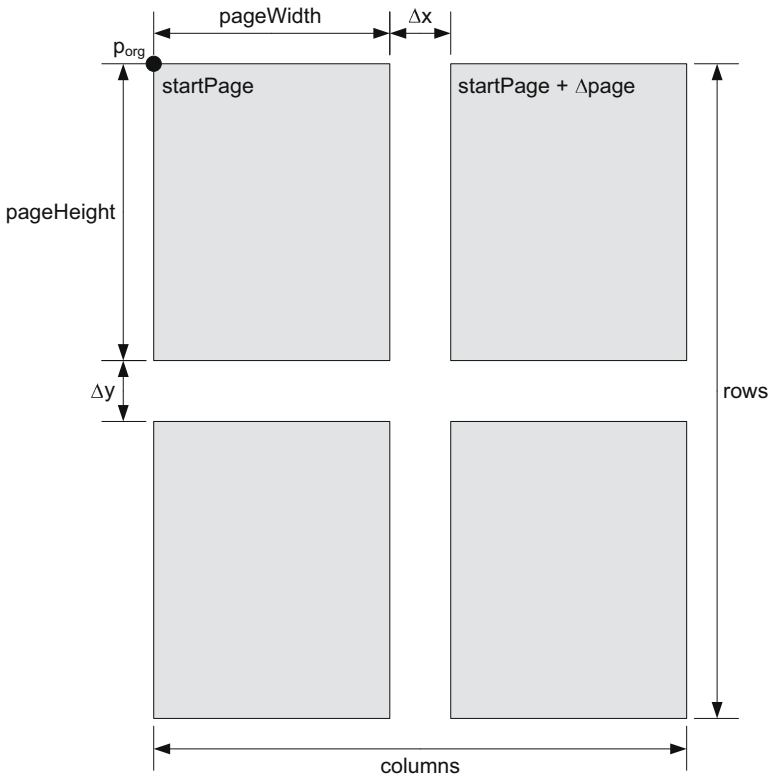


Figure 9 Multiple registered pages.

Any position p_1 lying somewhere within the space covered by the matrix of document pages can be mapped automatically to the corresponding *page* and position p_3 of the interactive paper framework's address space with the following transformation algorithm where each point p_i is defined as (x_i, y_i).

TRANSFORM(p_1)

1 $(x_2, y_2) \leftarrow (x_1, y_1) - (x_{org}, y_{org})$
2 $totalWidth \leftarrow pageWidth + \Delta x$
3 $totalHeight \leftarrow pageHeight + \Delta y$
4 $x_3 \leftarrow x_2 \bmod totalWidth$
5 $y_3 \leftarrow y_2 \bmod totalHeight$
6 $column \leftarrow x_2 \div totalWidth$
7 $row \leftarrow y_2 \div totalHeight$
8 $page \leftarrow (row \times |columns|) + column + 1$
9 $page \leftarrow startPage + (page \times \Delta page) - \Delta page$
10 **return** *page* and p_3

First, we have to translate the (x,y) coordinates of the input p_1 to get a new position p_2 which is relative to the upper left corner of the page matrix by subtracting the origin p_{org} from the position detected by the pen. The final position p_3 relative to a single page can then be computed by applying a modulo operator with the new position p_2 and the page width and height of the page as arguments. Finally, we get the page number by computing the corresponding row and column of the page matrix. The page and positional information is then returned together with the page identifier in the form of a location object which can be further processed by the interactive paper framework. Note that the new Magicomm G303 Digital Pen

Figure 10 Document registration.

```
<?xml version="1.0" encoding="UTF-8"?>
<documents>
<document>
  <identifier>edfest_brochure</identifier>
  <connectedPages>
   <startPage>1</startPage>
   <pageOffset>1</pageOffset>
   <origin>
    <point><x>89551936</x><y>27288</y></point>
   </origin>
   <size>
    <width>148</width><height>210</height>
   </size>
   <columns>53</columns><rows>1</rows>
   <spacingX>251</spacingX><spacingY>0</spacingY>
  </connectedPages>
</document>
...
</documents>
```

no longer delivers global Anoto coordinates but rather positions relative to specific Anoto pages. Therefore, the mapping of Anoto coordinates to positions relative to iPaper pages had to be slightly adapted to integrate the Magicomm G303 pen.

Based on this document handler, we can register the different documents as shown in Figure 10. In this example, the festival brochure (edfest_brochure) is registered with the Anoto position (89551936, 27288) as its origin. We further define the document size (148 × 210 mm) and the number of pages (53) that the document contains. Note that the map and the bookmark have to be registered in the same way.

5 Web information system requirements

As shown in previous sections, the domain of mobile applications poses entirely new requirements in terms of content publishing. Additionally, the choice of paper as an input channel has also led to challenges that have not yet been faced by web information systems. In this section, we will present these new requirements and describe how they were met by XCM, our content management system that was already briefly introduced in Section 4.

Mobile applications such as our paper-based tourist guide extend the traditional notion of the request-response pattern that is the basis of most existing web information systems. Due to the importance of context in such applications, data delivery might be triggered without an explicit interaction of the user with the system. An example of such a content delivery is the reminder functionality available in the brochure for each event. In order to allow this functionality to be handled by the content management system, it has to be able to contact the user agent without it having sent a prior request. As this effectively presents a shift from synchronous content delivery to asynchronous communication, we had to extend both the client browser as well as the server of the publishing platform to allow such behaviour.

In our application, the content management system does not simply communicate with a desktop browser capable of handling only one document format, but rather with a client that is able to process several different formats and therefore content delivery becomes even more challenging. Our publishing platform has to be able to cope with voice output, HTML pages, handwriting recognition and the generation

of images representing the drawings captured by the client and sent to the content management system. Further, communication is not always limited to only one channel at a time. It is also imaginable that the application requests the text of a handwritten note as both spoken text and an image at the same time. To handle these multi-channel requirements, we have broadened our notion of context to also include information that characterises the client device and the output formats it supports and prefers. For example, in the case of the interactive paper client, the information specifying which document was used at the time a request was generated is considered to be context and is used by the server to handle such requests accordingly. Hence, an important requirement for our publishing platform was that is does not make any assumption about what information is to be considered context. Rather than predefining context to be a set of fixed dimensions such as location, user preferences etc., our content management system has to be able to handle context as defined by the application.

As many of the functionalities offered by the mobile tourist guide are quite complex and all application data is managed by an application database, it is not possible to approach data management and delivery using a traditional content management system that represents its units of content delivery as a collection of texts, images and links. Another requirement for a web information system therefore is to rely on a platform that can deal with application concepts such as events, performances and venues. Otherwise, the examples given above would not be manageable by both the application developer and the data management system as the natural object semantics are lost in favour of presentation aspects. Therefore, our system is able to manage an arbitrary application database independent of its own database containing the metadata required for the context-aware publishing of application data.

To fulfil the requirements described above, the eXtensible Content Management (XCM) system has several built-in features that allow highly context-dependent, multi-channel delivery while, at the same time, allowing developers to work with concepts that are semantically important to their application [4]. In contrast to most existing content management systems, XCM has no built-in assumption about which information is regarded to be context in a given application domain. Hence, a possible definition for context as used in XCM is all information and parameters that influence the response to a request at the time it is issued to the content management system. The only restriction therefore is how context information has to be represented and communicated to the server in order for it to adapt the generation of its response. For this purpose, we define a context state C to be a set of $\langle name, value \rangle$ pairs representing individual context values.

$$C := \{\langle k_1, v_1 \rangle, \langle k_2, v_2 \rangle, \ldots, \langle k_n, v_n \rangle\}$$

The set of keys $K_{ctx} := \{k_1, k_2, \ldots, k_n\}$ denotes the names of the context dimensions such as *format*, *position*, *document* etc. that are specified by the context state, whereas the set of values $V_{ctx} := \{v_1, v_2, \ldots, v_n\}$ contains corresponding values for the specified dimensions. Note that our approach does not restrict which keys can or should be used and there are no names that are treated in a special way by the system.

Context-aware access to data and metadata is then achieved through the use of multi-variant objects. A multi-variant object can have multiple representations that

are annotated by a set of properties describing the purpose of each variant. Similar to context values, properties are also represented as ⟨*name*, *value*⟩ tuples. For example, in the domain of web engineering, an image object could have variants for different image formats such as JPEG, GIF or WBMP. To adapt a content response to the current context state C, a matching algorithm compares the context values to the variant property values and retrieves suitable variants of all requested objects at runtime. Rather than performing an exact match, the algorithm computes the variant of an object that best matches the current context. Doing so, the algorithm can cope with the fact that not every combination of context values has a dedicated object variant and thus ensures scalability by preventing an exponential growth of required variants. Further, to give complete control over the matching process to the developer, we have introduced a special syntax for property values shown in Table 1.

The upper part of the table describes the different syntax patterns that our matching algorithm recognises for property values. A single value can be specified to express that the property value matches the corresponding context value if they are exactly the same. An application of this would be the property `format=html` to describe the output channel of an XSL template. A wildcard value, for example `login=*` can also be specified. While the advantage of this may seem cumbersome at first, it simply expresses that a value for the given property should be present in order for the corresponding variant to be valid. Sets of values can be given by separating the individual values with colons. For example, the property `lang=en-uk:en-us:en` could be used to express that a variant contains English content. Finally, for value domains that have a mathematical order relation, it is possible to declare ranges by giving the lower and the upper values. A variant that is valid for a range of browser versions could be annotated with `version=5.5..7.0`. The lower part of the table contains the modifiers that can be used as a prefix in combination with any of the values from the upper part. The two modifiers that are recognised by our matching algorithm can be used to change the semantics of a given property value. On the one hand, if a property value of a variant is prefixed with the plus sign '+' the algorithm will treat this property as a required property, meaning that if the current context specifies a different or no value, the variant is not considered to be valid. On the other hand, if a property value is prefixed with a minus sign '-', the matching algorithm will treat any object variants that specify the same value as the current context as invalid. An example of the former would be the property `format=+html` to express that a given variant of a template object can only be used for the output channel for which it is intended. Similarly, annotating a variant with `login=-fbloggs` could be used to prevent the user `fbloggs` from ever accessing this variant.

Table 1 Syntax for variant property values.

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.

Instead of providing this context-aware access to data by means of a framework as proposed in [10], we believe it is important to place such functionality at the level of a database management system. When maintaining a web information system, reliability and evolution are key requirements. Both issues have been addressed by traditional database systems with recovery technologies, versioning and schema evolution capabilities. Therefore, we decided to integrate our concept of multi-variant objects with a versioning mechanism capable of historisation along the time axis. This new versioning model and the object representation that was derived from it form the core of an object-oriented database system which we have developed [22].

Based on this database system, we have implemented the XCM publishing process shown in Figure 11 that governs all content delivery done by XCM. The process is built around four concepts—content, structure, view and layout [14]—that are all represented as multi-variant objects in a database. The figure shows the actions that will happen when a user selects the 'i' pictogram shown in Figure 4b to request information about an event. Generally, such a request to the system will point to either a content or a structure element. This particular request is an example of the first case and thus the system will retrieve the content object from the application database representing the requested event. After selecting the appropriate variant of this object, it is post-processed by a pre-defined view component that determines which attributes should be displayed or which additional information should be linked to the content. Finally, the content object is rendered for presentation on the client device by applying layout templates to transform it for the corresponding output channel. For our example request, the response will be output to the user using text-to-speech and the information is therefore rendered as a prompt in VoiceXML. In the case that the request is to a structure element, a hierarchical structure is generated before rendering all content elements included in this tree in the same way as before.

The expedience of XCM for highly context-dependent content delivery stems from the fact that XCM does not limit context adaptation to content only. By representing both data and metadata in the system using multi-variant objects, any aspect of the publishing process discussed above can be made context-dependent. Thus, it is possible to handle the need for different structures for different output channels, the internationalisation of content or the personalisation of the application for different users with the same concept.

It is important to note that XCM does not entail any models or CASE tools to design web information systems as brought forward by approaches such as

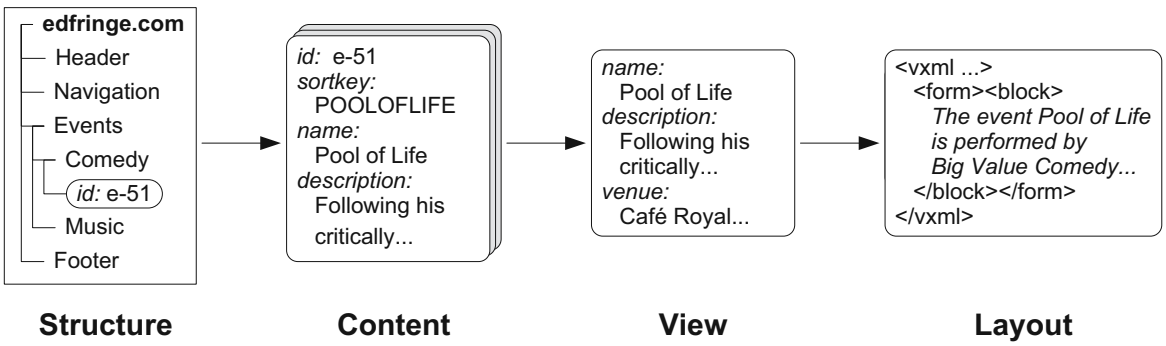


Figure 11 XCM publishing process.

WebML [8] and Hera [17]. Rather, it is an implementation and execution platform that could support the extensions for context adaptation that have been proposed by WebML and Hera in [7] and [13], respectively. The inclusion of XCM as a data management and content publishing component in our mobile tourist guide has highlighted the importance of a clean separation between an application database and the publishing metadata as well as the need for multiple object variants for different purposes.

6 Context-aware interactions

Having presented interactive paper as a client device and the capabilities for context-awareness of our content management system, we now go on to show how these two systems have been integrated into the overall architecture. When designing our architecture we were keen to obtain a platform that is general enough to support context-aware interaction over multiple delivery channels for a range of different clients. An overview of the resulting architecture is provided in Figure 12. It is important to note that the core of this architecture is essentially the same as that of existing platforms that have been proposed in the domain of web engineering. However, due to several extensions on both the client and the server side, we were able to increase the flexibility of our architecture compared to existing solutions.

To obtain context-aware interactions, a client in our architecture needs to acquire and manage its current context. Therefore we extended our client platform with a *Client Controller*, a component that polls all available sensors and represents their

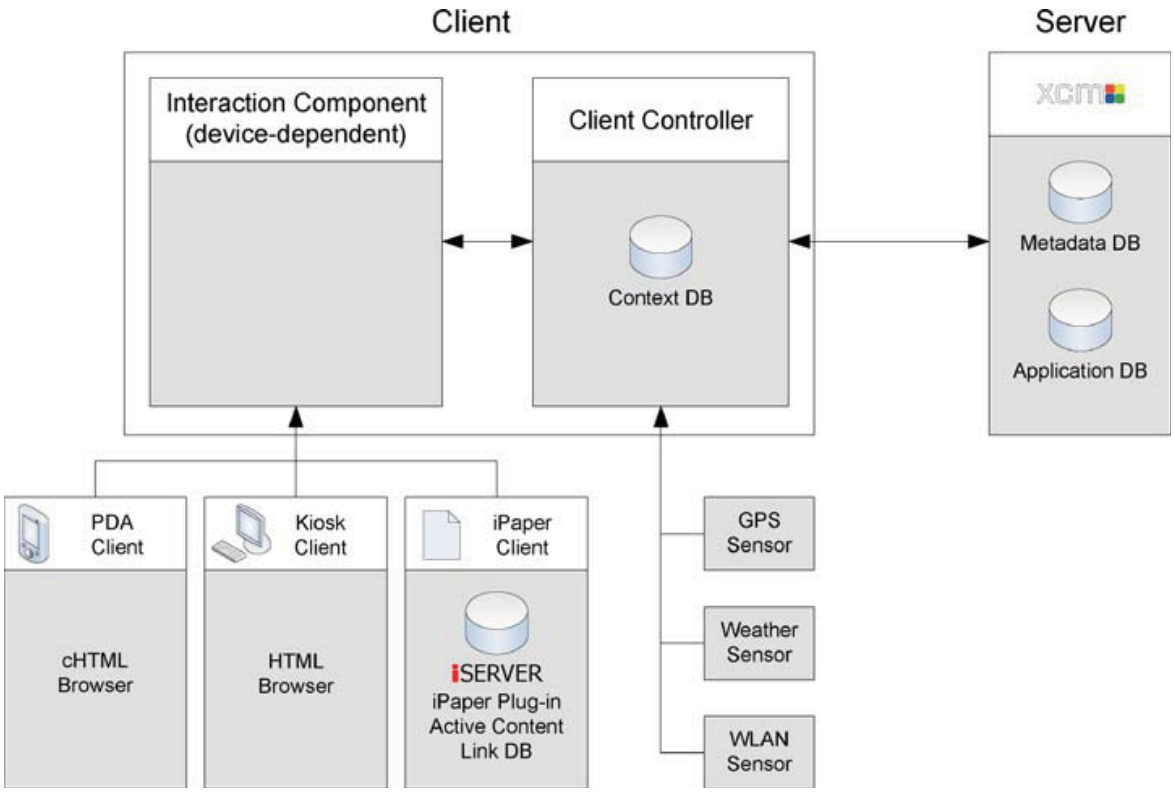


Figure 12 Architecture overview.

data using a conceptual model of context [3]. In our mobile tourist guide application, the client controller manages three different sensors. A GPS device records the position of the user and thus enables location-aware services, such as finding events nearby or navigating the user through the city. The weather sensor is responsible for providing information about the current meteorological conditions to the system and was implemented by downloading the corresponding information from a public weather web site on the Internet. This data could be used to decide whether navigation information should recommend public transport or walking routes. The third sensor is a connectivity sensor that accesses the networking subsystem of the platform where the client is running and queries the current signal strength of the IEEE 802.xx wireless connection. The information delivered by this sensor is used for two different purposes. First, it tells the client system whether there is a network available to be used for certain functionality. Second, and maybe more importantly, by switching to ad-hoc mode, the wireless network adapter can be used to detect other mobile tourist guides nearby. At the moment, if another user is detected in this way, the system simply informs both users of the presence of another user. However, in on-going developments of the system for user-based collaborative filtering [9], we are using this functionality to initiate a peer-to-peer exchange of rating and comment information between the two guide systems.

The Client Controller acts as a proxy component running on the client which filters all requests issued by any device-dependent *Interaction Component*. While the Interaction Component for the *Kiosk Client* is a regular HTML browser, we use compact HTML (cHTML) for PDAs. For the Interactive Paper Client (*iPaper Client*), iServer is installed on some form of mobile client device. Before sending a request from the client to the server, the Client Controller augments the request with the necessary context information. For example, the following request to retrieve an event rating from the XCM server

```
http://../xcm?anchor=getRating&event=e-51
```

would be translated into

```
http://../xcm?anchor=getRating&event=e-51&format=vxml&user=guest
```

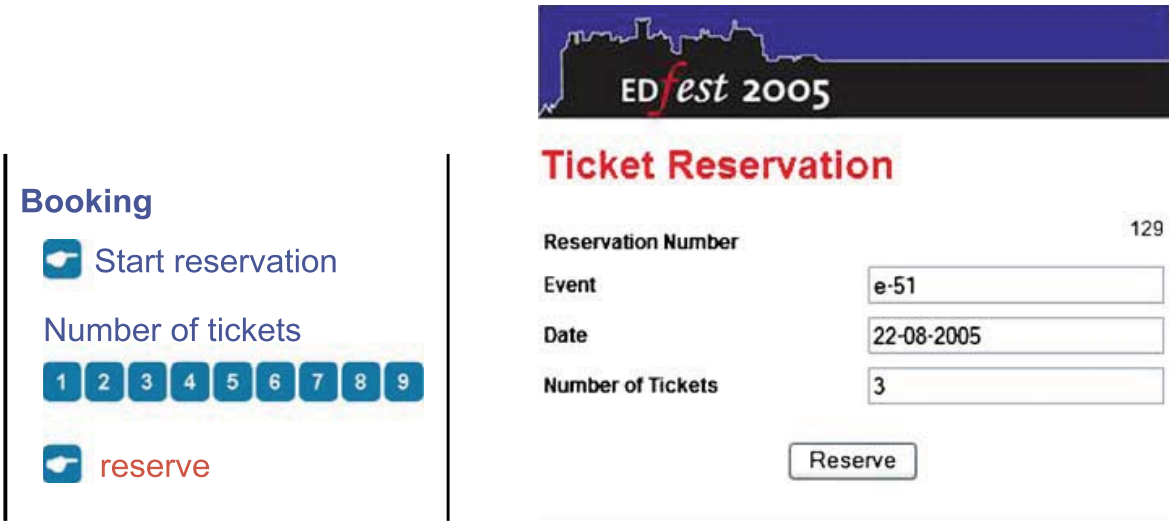
By inserting these additional parameters into the query string of the request, the client provides information about the format it is capable of handling and the user that is currently logged in. The server can then extract this context and select the appropriate variant of the content as described in Section 5. To function properly, all output channels available on a specific client device have to be registered with the Client Controller and be defined in terms of capability profiles. For some output channels, part of the Client Controller's functionality could also be implemented using other technologies such as sessions. In our architecture, however, we have put great emphasis on imposing no assumptions or requirements on client devices and thus have decided to manage the state of the client for all devices in the same way.

The Client Controller does not interface with a client device's Interaction Component only. It also provides call-back services to the content management server to cater for asynchronous communication. An example of such a service is the reminder functionality offered by the festival guide system. If a user wishes to be notified before a particular event begins, a request is sent to the content management system containing the id of the chosen event. The server sends back an acknowledgement to

the client but, as a side effect, starts a timer task that will execute in the appropriate moment. Note that the time offset that determines how far in advance the user is notified could be made dependent on the user's location that is also transmitted to the server as a result of the context augmentation of requests described above. In our current implementation, however, we have simply assumed a fixed offset of 30 min. When the timer task finally executes, it sends a request back to the Client Controller. All services offered by the Client Controller are implemented as Java Servlets, hence sending a request is equivalent to invoking a URL pointing to the desired service on the Client Controller. Upon receiving this request, the Client Controller decides to which channels it should dispatch the message based on the available devices and the current context state. For example, if there is a network connection, it could send an SMS to the user's mobile phone. If not, it could forward the reminder to the voice engine.

A challenge that had to be met on the server side was to cope with the different styles of interaction that stem from the different capabilities of the output devices. Whereas graphical interfaces such as HTML or cHTML allow a lot of parameters to be transmitted with one request by using forms, our pen client and also the voice interface communicate with the content management server at a finer granularity as the user needs feedback for every action that they carry out. An example of such an interaction scenario is the functionality to book tickets for an event. Using the paper interface shown in Figure 13a, a user starts the reservation process by pointing with the pen to 'Start reservation' pictogram. In the brochure, users select the desired event and date by clicking with the pen on the 'up-arrow' beneath the title of the show and the date at the bottom of an entry as shown in Figure 4b. After selecting the number of tickets that they want to book from the bookmark, the reservation process is completed by touching the icon labelled 'Reserve' on the bookmark. Each time a user points with the pen to a pictogram on the paper, a request is sent to the server which delivers a response indicating to the user that their selection has been recorded, informing them about possible errors or what further steps are required to complete the process. In the case of our prototype HTML interface displayed in Figure 13b, the interaction between client and server takes a completely different shape. When a user wishes to reserve tickets, they are presented with a form that has input fields for all required parameters. After they have filled in the desired values and submitted the form, another page will inform them whether their reservation has been processed or if there were errors. Note that, at the moment, the HTML interface is very basic and not very user-friendly as it was not the main focus of our user studies. To book tickets using this channel, users have to adhere to strict conventions about how to specify values. The event, for example, can only be specified in terms of the id of the corresponding object in the application database. While this leads to poor usability, it has no effect on the problems that have to be addressed when offering different styles of interaction for different devices. We note that a more extensive and user friendly HTML interface is currently under development as the basis for investigations on the introduction of kiosk-based systems that allow users to print out personalised daily event schedules as interactive documents.

To implement the scenario described above, we have once again profited from variants and context matching in XCM. While there is only one variant for the code component that is responsible for creating and updating the reservation in the



a Paper interface **b** HTML interface

Figure 13 Different modes of booking.

application database, there is a variety of template components that are tailored to certain context states in a way that enables them to provide the presented interaction styles. The idea behind our solution is that, depending on the interaction style, the requests that are exchanged between the client and the server exhibit very distinct patterns in terms of which parameters are included. Therefore the pattern of a request is also treated as context information by XCM and thus it is possible to also specify variants that adapt to these patterns. In the reservation example, the interactive booking process on the paper interface is characterised by a step-by-step accumulation of data where each subsequent request transmits a new value. When all values have been transmitted, the booking can be processed and stored in the database. The same functionality accessed through the HTML interface is characterised through a single request transmitting all parameters at once which directly leads to the final step of the booking. Employing this additional context, variants can be provided for each step of the process that are then automatically selected by the matcher of XCM whenever they are needed.

As mentioned earlier in Section 3, reviews play an important role in the selection of festival events to attend. While numerous reviews are provided in newspapers, our goal was to provide a tool that enables visitors of the festival to submit and share their comments in a natural and easy way without having to sit in front of a computer or write reviews using the small keyboards of their mobile phones. We therefore provide some empty pages at the end of our interactive festival brochure where users can write down their comments with their digital pen.

After a user has finished writing their comment, as indicated either by them selecting a special pictogram printed at the bottom of each comment page or by not writing for a certain time and thereby triggering a timeout mechanism, the comment is sent to XCM in XML format. However, before the data is sent to XCM, some basic processing of the raw pen data has to be done on the client side. If the pen is in contact with the paper, the pen continuously transmits positional information at a more or less constant rate. Since a user normally does not write a text or draw an

image in a single stroke, we have to apply a trace detection algorithm to the list of points transmitted by the pen. We determine traces heuristically by analysing the time elapsed between two successive positions transmitted by the pen. If the time difference exceeds a defined threshold, we assume that the pen has been lifted between the two points and therefore start with a new trace. The resulting iPaper note consisting of a number of traces is shown in Figure 14a. Note that, in addition to the (x,y) coordinates of each point, we also store a timestamp (ts) and the pressure (force) that was applied on the pen nib.

The resulting XML note is sent to XCM which stores the XML document and associates it with a specific event. When a user requests comments for a specific show, the stored XML messages will be rendered for a specific output channel. If the request comes from an iPaper Client providing voice output only, the XML message is transformed to an internal iPaper note representation which is then handed over to an Intelligent Character Recognition (ICR) engine. The text returned by the character recognition process is embedded in a VoiceXML message as shown in Figure 14b which is returned to the Client Controller that dispatches the message to a TTS engine. The VoiceXML message containing a user's comment is only returned to the client if the character recognition engine returns a high confidence value for the recognised text. However, if a handwritten note cannot be transformed to its textual representation, we can still render the comments managed by XCM for a visual output channel which can then, for example, be accessed from a browser running on a PDA or on the kiosk. Therefore, XCM makes use of some iPaper tools to apply a cubic spline interpolation to the set of points, scale the note to the desired size and finally create a JPEG image that can be embedded in an HTML document and returned to the client. An example of a JPEG image resulting from this process is shown in Figure 14c.

The storage of the messages as XML documents has the advantage that they can be transformed easily to different output formats if requirements of a client device change or new output devices have to be integrated. For example, in addition to the note transformation to a JPEG file, we already support Scalable Vector Graphics (SVG) [24], an XML format for vector graphics, as well as the emerging Ink Markup Language (InkML) [18] which has been especially designed for pen-based interfaces. Furthermore, we have implemented a Java visualisation component for captured pen

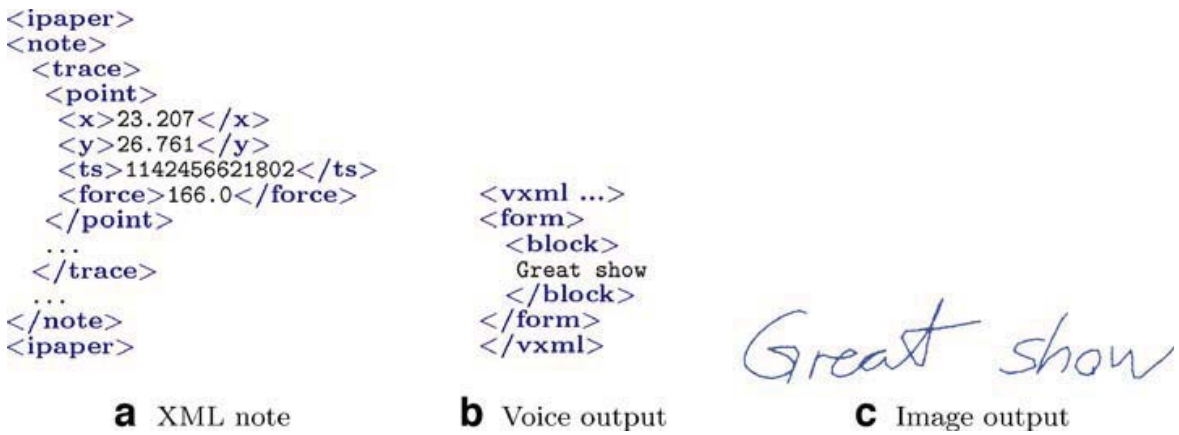


Figure 14 Comment rendering.

strokes, which can, not only generate a static image from the data stored in an XML note, but also replay an animated version of the captured comments.

Note that location-specific context can, not only be provided by the GPS sensor introduced earlier in this section, but also from pen-based interaction with one of the available interactive paper maps. If a user points to the map with the digital pen, the pen coordinate is automatically transformed to the Universal Transverse Mercator (UTM) coordinate system and sent as a request parameter to XCM. In order to perform this coordinate transformation, the different sized paper maps have to be registered with the UTM coordinate system. Again it was important that different input modalities could be used to access the same service. For example, there are two possible means of invoking the service to find the nearest venue. A user's current position provided by the GPS sensor can be used to look up the nearest venue. Another possibility to invoke the same functionality requires that a user selects a position on the map with the digital pen. The pen position is transformed to the corresponding UTM location which then provides the new contextual information for the query to find the nearest venue.

For the festival application we designed a specific application database where all the festival content was managed by XCM. Therefore, we used the iServer's link functionality mainly for the integration of the paper interface whereas links within the HTML interface were directly addressed by XCM. As soon as we start to add links to external resources and any third-party applications, iServer can be used to manage these cross-media associations. For example, based on the iWeb plug-in for XHTML content, it becomes possible to define links addressing parts of existing HTML documents. The interplay of iServer and XCM raises the question if parts of iServer's cross-media link functionality should be directly integrated into the core of XCM and a content management system should not only supported the publishing of information on different output channels but also enable the integration of arbitrary existing resources.

7 Evaluation and discussion

In this section, we first evaluate our approach of extending a web information system to support multi-channel, multi-modal and context-aware access to information and then go on to report specifically on the use of interactive paper coupled with speech as an interface to the festival guide that we developed.

Not only have we demonstrated the feasibility of our approach by implementing a relatively complex application that supports both the process of publishing interactive documents and interaction with these documents, but we have also used the platform to investigate a number of alternative interfaces and advanced services for this application. Indeed, the set of documents presented in this paper represents only one form of paper-based interface and set of services. For example, the first version of the system used speech as both an input and output channel alongside a paper brochure where areas of text were linked to services rather than using special icons. We have also developed a kiosk-based application where users can select events and print out a personalised, interactive daily schedule of events. In addition, a concept of paper event stickers allows bookmarks to digital information about an event to be attached to other documents or even to physical entities. In addition, we have had

student projects which experimented with speech only interfaces, the use of head-mounted displays, landmark navigation systems and services to locate friends and other users. All of these run alongside more traditional web interfaces on desktop PCs, laptops and PDAs. We believe that it is important to develop general platforms that not only support the development of such applications, but also experimentation with different modes of interaction. Our experience to date has convinced us of the generality of our approach and the benefits of database-driven content management. Specifically, we have demonstrated the benefits of treating all aspects of context-awareness uniformly through the single notion of multi-variant objects. Given the complexity of the festival application and the multiplicity of channels that it supports in addition to other forms of adaptation such as location-awareness, we have shown that the approach is scaleable in terms of the number of variants required.

However, one of the main complexities is that of managing the design and development of applications and while our approach offers the necessary mechanisms and infrastructure for context-awareness, clearly it needs to be coupled with a model-based approach to the design and development of web information systems. In the future, we therefore plan to investigate the possibility of integrating our platform with tools for web information system design and development such as those proposed by WebML.

Although it is beyond the scope of this paper to report in detail on the user studies carried out on the specific set of interactive documents described in this paper, we think it is useful to present some of the key issues raised by these studies. Two forms of user studies have been carried out to date—one in the setting of the festival and the other lab-based. Both were qualitative rather than quantitative and involved studying video recordings of users working with the documents after a brief introduction to the system.

During the Fringe Festival in 2005, both system and user trials were carried out during a five-day period at various locations in the city, including public places as well as locations in and around festival venues. For the user studies, more than twenty sessions, each lasting about 30 min, were carried out with individual and pairs of festival visitors. The tourists were provided with the documents, a single digital pen and an earpiece each. In this setting, we had both the client and the server running on a laptop with a Bluetooth connection to the digital pen and wired connections to the earpieces. This enabled us to not only log the sessions, but also to monitor the user interactions via a visual display and additional earpieces. At the same time, video recordings provided us with a record of how users responded to the system and, importantly, what they said to each other when in pairs. Since the laptop was remote from the users and they could not see the screen, the focus of their interaction was around the paper documents, digital pen and earpiece as it would be in a truly mobile setting where the client device would be either wearable or located in a pocket or bag and the connection to the server would be through a wireless network.

After a short introduction to the use of the digital pen to activate pictograms printed in the brochure and map functionality, the users were left to freely experiment with the system. They very quickly got the idea of the system and generally the response was positive. Users found the map-based interaction, including the locator functionality, to be particularly intuitive and very useful and they were also very positive about the means of inputting and getting event ratings as well as for setting the reminders.

However, a number of issues appeared when it came to the more complex services that involved several interaction steps and possibly more than one document. An example of this is the ticket reservation service that involves both the bookmark and the brochure. Users initiate the service by selecting the appropriate pictogram on the bookmark and then they should select an event within the brochure followed by one of its performances by selecting a date from the list given in entry. After this, they should select the number of tickets on the bookmark and then complete the transaction by selecting the reserve pictogram on the bookmark. Although the users were guided at each step by instructions given through the speech output channel, they would tend to get confused when they had to move to another document and would sometimes even lose the context by selecting an event and then selecting a date from the brochure listing of another event on the same page or even from the calendar on the bookmark intended to be used for specifying searches. The problem is due to the non-linearity of the interface, especially when multiple documents are involved. Since printed documents are static and no order is imposed on the selection of areas within a document, the application has to be extremely flexible to take all possibilities into account and adapt to the specific transactional context. This is very much an issue of how to design the documents, how to design the dialogue to ensue that instructions are clear and how to design the services to ensure this flexibility. Since interactive documents open up a whole new area of publishing possibilities, this is something that will require a great deal of experimentation in order to establish good design guidelines.

A second problem that arose in the case of complex interactions was that of knowing when to abort a transaction. Interaction with a document was often dependent upon a transactional context. For example, as outlined in the above example selecting a date for an event in the context of a transaction for reserving tickets would be interpreted as providing the necessary parameter to that transaction. If on the other hand, a user selected an event date outside of that context, then they would be given information about the availability of tickets for that performance. Frequently users would decide not to complete a transaction and would simply use the pen to activate some other service. We therefore used timeouts to abort a transaction when the expected next step had not been completed within a reasonable time period. Of course the problem is deciding on appropriate timeout periods and it was often the case that when users had to switch between documents, there would be significant delays and the transaction would be terminated. Because all of the interactions produced some effect, but it could be dependent on the transaction context, this would mean that the user would always get a response but perhaps not what they expected. Since they were not aware that the transaction had terminated, they would often go back and repeat the previous step but, in some cases, this caused even more confusion. Clearly we would need to refine our use of timeouts so that they allow longer times for inexperienced users, but it is still an issue to know how best to make handle complex interactions and make users aware of transaction contexts.

Similar problems arise when users switch between interaction and capture modes. When the pen enters a capture area, then the pen switches from interaction mode to capture mode. The return to interaction mode occurs either when the pen leaves the capture area or when there is a timeout caused by the user removing the pen from the surface of the paper for a significant amount of time. Again there is the issue of how best to make users aware of this switch between modes. Ideally, the pen itself should

provide a mechanism to switch between modes, for example by having a retractable writing stylus to indicate the switch from writing to interaction mode.

As described above, we used laptops as the client devices in the user studies where the users were actually not mobile during a session, although possibly situated in a mobile setting. In system trials, we have also experimented with mobile client devices such as PDAs and Xybernaut wearable computers, and are also tracking developments in wearable computers such as the belt-based QBIC [1]. Our experiences have shown some mobile devices to be problematic in terms of reliability and also ready availability of some software components required for complex multi-modal applications such as our festival system. However, clearly mobile devices are rapidly evolving and such problems should soon disappear. In the meantime, recent developments in digital pen technology may help to overcome this need for a separate wearable computer. The Fly Pentop Computer [11] demonstrates the tendency to integrate more functionality into the pen in terms of not only processing power but also input/output devices such as speakers. One could also consider better support for mobility in terms of WiFi connectivity and GPS. Clearly, the decision of what functionality should be integrated into the pens remains an open issue but we feel that this is a promising direction for future mobile applications based on interactive paper.

8 Conclusions

We have motivated the potential use of paper-based access to digital services in mobile environments by describing some of the advantages it offers in terms of portability, annotation and continued usefulness as an information medium even when no power is available. We previously designed and implemented a special cross-media link server that, through the use of plug-ins, supports the linking of paper and digital resources. In this paper, we described how this server has been combined with a multi-channel web information system to build a platform for highly context-aware applications where paper can co-exist as a new form of web channel alongside traditional web browser interfaces.

To demonstrate the functionality of the platform and test its flexibility, we developed a paper-based mobile tourist guide for a large international arts festival. This guide is based around a set of three interactive documents that can be used independently or together to provide access to a range of services such as information about navigation, access to venues, reviews and also event bookings. It serves to demonstrate how different modes of publishing and interaction can be combined within the same framework to support new technologies and requirements for multi-channel and multi-modal interfaces. In particular, we described in detail how we supported the publishing of the paper documents as well as the interaction from these documents within a single web information system. Further, in addition to the more traditional forms of synchronous request-response processing, we had to support fine-grained interaction processes required for transactions such as ticket reservations and the asynchronous push of information required for event reminders.

One of the key concepts underlying our approach is that of multi-variant objects and we have shown how they enable us to handle many of these issues in a uniform way. Together with an algorithm that selects the appropriate variant depending on

the current context, multi-variant objects provide a simple yet powerful basis to implement systems that can adapt content, structure, presentation and behaviour according to all sorts of situational factors ranging from the user and their device to environmental factors such as the location and weather.

Acknowledgements We thank Rudi Belotti, Corsin Decurtins, Ljiljana Vukelja and Nadir Weibel for their contributions to the EdFest project.

References

1. Amft, O., Lauffer, M., Ossevoort, S., Macaluso, F., Lukowicz, P., Tröster, G.: Design of the QBIC wearable computing platform. In: Proceddings of ASAP 2004, 15th IEEE International Conference on Application-Specific Systems, Architectures and Processors, pp. 398–410. Galveston, USA (September 2004)
2. Ashman, H., Simpson, R.M.: Computing surveys' electronic symposium on hypertext and hypermedia: editorial. ACM Comput. Surv. **31**(4), 325–334 (December 1999)
3. Belotti, R., Decurtins, C., Grossniklaus, M., Norrie, M.C., Palinginis, A.: Modelling context for information environments. In: Proceedings of UMICS 2004, 2nd International Workshop on Ubiquitous Mobile Information and Collaboration Systems, CAiSE 2004, pp. 43–56. Riga, Latvia (June 2004)
4. Belotti, R., Decurtins, C., Grossniklaus, M., Norrie, M.C., Palinginis, A.: Interplay of content and context. J. Web Eng. **4**(1), 57–78 (2005)
5. Brown, B., Chalmers, M.: Tourism and mobile technology. In: Proceedings of ECSCW 2003, 8th European Conference on Computer Supported Cooperative Work, pp. 335–355. Helsinki, Finland (September 2003)
6. Brown, B., Chalmers, M., Bell, M., MacColl, I., Hall, M., Rudman, P.: Sharing the square: collaborative leisure in the city streets. In: Proceedings of ECSCW 2005, 9th European Conference on Computer-Supported Cooperative Work, pp. 427–447. Paris, France (September 2005)
7. Ceri, S., Daniel, F., Matera, M.: Extending WebML for modeling multi-channel context-aware web applications. In: Proceedings of MMIS 2003, International Workshop on Multichannel and Mobile Information Systems, WISE 2003, pp. 225–233. Rome, Italy (December 2003)
8. Ceri, S., Fraternali, P., Bongio, A.: Web modeling language (WebML): a modeling language for designing web sites. Comput. Netw. **33**(1), 137–157 (2000)
9. de Spindler, A., Norrie, M.C., Grossniklaus, M., Signer, B.: Spatio-temporal proximity as a basis for collaborative filtering in mobile environments. In: Proceedings of UMICS 2006, 4th Workshop on Ubiquitous Mobile Information and Collaboration Systems, CAiSE 2006, pp. 912–926. Luxembourg, Grand Duchy of Luxembourg (June 2006)
10. De Virgilio R., Torlone, R.: A general methodology for context-aware data access. In: Proceedings of MobiDE, 4th ACM International Workshop on Data Engineering for Wireless and Mobile Access, pp. 9–15. Baltimore, USA (June 2005)
11. Fly Pentop Computer. LeapFrog Enterprises, Inc. http://www.flypentop.com
12. Franklin, M.J.: Challenges in ubiquitous data management. In: Informatics—10 Years Back. 10 Years Ahead, pp. 24–33 (2001)
13. Frăsincar, F., Barna, P., Houben, G.-J., Fiala, Z.: Adaptation and reuse in designing web information systems. In: Proceedings of ITCC 2004, International Conference on Information Technology, pp. 387–391. Las Vegas, USA (April 2004)
14. Grossniklaus, M., Norrie, M.C.: Information concepts for content management. In: Proceedings of DASWIS 2002, International Workshop on Data Semantics in Web Information Systems, WISE 2002, pp. 150–159. Singapore, Republic of Singapore (December 2002)
15. Grossniklaus, M., Norrie, M.C., Signer, B., Weibel, N.: Putting location-based services on the map. In: Proceedings of W2GIS 2006, 6th International Symposium on Web and Wireless Geographical Information Systems. Hong Kong, China (December 2006)
16. Guimbretière, F.: Paper augmented digital documents. In: Proceedings of UIST 2003, 16th Annual ACM Symposium on User Interface Software and Technology, pp. 51–60. Vancouver, Canada (November 2003)

17. Houben, G.-J., Barna, P., Frăsincar, F., Vdovják, R.: Hera: development of semantic web information systems. In: Proceedings of ICWE 2003, International Conference on Web Engineering, pp. 529–538. Oviedo, Spain (2003)
18. Ink Markup Language, W3C Working Draft 28 (September 2004) http://www.w3.org/TR/InkML/
19. Ljungstrand, P., Redström, J., Holmquist, L.E.: WebStickers: using physical tokens to access, manage and share bookmarks to the web. In: Proceedings of DARE 2000, Designing Augmented Reality Environments. Elsinore, Denmark (April 2000)
20. Magicomm G303 Digital Pen. http://www.magicomm.co.uk
21. Nokia Digital Pen, Nokia. http://www.nokia.com
22. Norrie, M.C., Palinginis, A.: Versions for context dependent information services. In: Proceedings of COOPIS 2003, 11th International Conference on Cooperative Information Systems, pp. 503–515. Catania, Italy (November 2003)
23. Reilly, D.F., Rodgers, M.E., Argue, R., Nunes, M., Inkpen, K.: Marked-up maps: combining paper maps and electronic information resources. Personal Ubiquitous Comput. **10**(4), 215–226 (2006)
24. Scalable Vector Graphics (SVG) 1.1 Specification, W3C Recommendation. http://www.w3.org/TR/SVG11/
25. Signer, B.: Fundamental concepts for interactive paper and cross-media information management. Ph.D. thesis, ETH Zurich (2006)
26. Simpson, J.E.: XPath and XPointer. O'Reilly, Cambridge (August 2002)
27. Wellner, P.: Interacting with paper on the Digitaldesk. Comm. ACM **36**(7), 87–96 (July 1993)