Producing Interactive Paper Documents based on Multi-Channel Content Publishing

Michael Grossniklaus, Moira C. Norrie, Beat Signer and Nadir Weibel
Institute for Information Systems
ETH Zurich
8092 Zurich, Switzerland
grossniklaus,norrie,signer,weibel@inf.ethz.ch

ABSTRACT

Digital pen and paper technologies provide the basis for linking digital content and services to printed materials in the form of interactive paper publications. To realise the potential of these technologies, it is important to develop platforms and tools that can support the large-scale publishing of interactive paper documents. We show how an extensible content management system that was developed to support context-aware publishing was used for the production of interactive paper documents. The publishing process consists of two phases and requires one channel to support the production of the document together with cross-media link definitions and a second channel to support interaction with the document.

1. INTRODUCTION

Nowadays it is common for publishers to produce digital materials such as a CD-ROM or web site to accompany books. We have shown previously how emerging technologies such as Anoto-based digital pens can be used to bridge the paper-digital divide, enabling users to directly access digital content and services by interacting with printed materials [14]. For the publisher, this presents two major challenges. The first concerns the design of such cross-media materials since it opens up many new exciting possibilities in terms of the various forms of interaction that can be supported and there are currently no conventions or guidelines for the design of the book as an interface. The second challenge is how to support the production of interactive paper documents as it is tedious to perform manual authoring of the cross-media links for documents of any significant size.

In this paper, we address the second of these challenges by describing how we used an extensible content publishing system (XCM) that we developed previously to support both the production and use of such documents. XCM supports multi-channel publishing through an underlying database system with integrated support for context-aware applications. Essentially, objects are used to represent all aspects of a document—content, structure and presentation—and alternative versions of objects are labelled with the contexts in which they can be used. We can support multiple channels through a specific context dimension, adapting the content, structure and layout to a given channel. To support the production of interactive paper documents, we had to introduce a two-phase process of first generating the documents and second supporting interaction with these documents. The generation of the documents also involves parallel processes of publishing a PDF rendering of the document to be printed and, at the same time, automatically generating the cross-media link definitions where the link anchors are defined as physical areas within the printed pages. We have developed a range of augmented publications which use various output channels when interacting with the documents, including speech and HTML browsers.

After discussing background work on interactive paper in Section 2, we begin in Section 3 with a discussion of our approach and the underlying technologies for interactive paper and context-aware content management. The publishing phase of the interactive paper channel is presented in Section 4, while the interaction phase is described in Section 5. Concluding remarks are given in Section 6.

2. BACKGROUND

Anoto digital pen solutions [1] were developed for the capture of handwritten information and, originally, interactivity was limited to specific command buttons for actions such as sending data or changing pen stroke attributes. Recent developments in the pens and patterns support more general forms of interactivity based on real-time streaming of data. For example, the Fly Pentop computer [7] provides a number of desktop applications where users can, not only interact with the application using the pen, but also sketch their own interfaces. We have developed a number of applications that demonstrate the potential of pen-based interfaces to applications and also the use of Anoto technologies as a basis for augmenting printed documents with multimedia materials [14, 12].

Although Anoto provides several software tools to support the generation of printed documents with Anoto pattern and also application software, these are mainly targeted at capture-based applications such as forms processing and they require significant programming effort. The HCI group at Stanford has developed a simple programming framework to alleviate some of the programming effort by providing support for the handling of pen events etc., thereby making it easier for students to experiment with innovative uses of

the technologies [16]. But it still requires some programming effort and is intended for the prototyping of small applications rather than the development and deployment of complex applications or the large-scale publishing of interactive documents. In contrast, our goal was to develop a framework and tools that, similar to the World Wide Web, turn application development into an *authoring activity* rather than a *programming activity*. By this, we mean that users may develop all sorts of paper-based interfaces to applications and interactive documents by authoring links between paper and digital content and services. A notion of *active components* is used to link to pieces of program code that may either be applications in their own right or bridges to existing applications. By providing an extensive library of generic active components, application developers can simply define links to a wide range of services without having to write any code themselves. Only in the case where the library has to be extended with new active components does the application developer have to do any programming themselves.

Just as the authoring of web documents, and the links between them, rapidly grew beyond manual authoring tools and required publishing tools that could generate and maintain the documents automatically, the same is true for interactive paper documents. Nowadays no one would consider developing a web site of any scale without the use of a content management system to support both the authoring and delivery of content. We therefore decided to investigate how we could adopt this approach for the production of interactive paper documents.

3. APPROACH

Our solution for producing and implementing interactive paper documents is based on the coupling of an existing framework for interactive paper (iPaper) [10] and a context-aware content management system (XCM) [4]. In this section, we will outline the main features of each system, focussing on the aspects most relevant to the work reported in this paper. We begin by describing the iPaper framework which is the component that enables the linking of paper to digital content and services. We then go on to present XCM that uses the concept of context-aware versioning to provide support for multi-channel applications.

The iPaper framework was developed to support the rapid development and deployment of interactive paper applications. Active areas can be defined on paper documents and linked to any kind of digital media and services. The framework supports different types of pen-based interaction such as the selection of active areas for accessing linked information, the execution of digital services based on gesture recognition commands and the capture of handwritten notes. Information delivered by the digital pen can be processed in real-time or stored for later processing (batch mode). Our iPaper solution for interactive paper applications supports different positional input devices including the digital pen and paper technology offered by Anoto. A wide variety of interactive paper applications have been realised based on the iPaper framework including a set of augmented paper documents for visitors to a festival based on an interactive event brochure, map and bookmark [14]. For the definition and management of paper-digital links, iPaper uses functionality offered by iServer [12], a general cross-media information platform.

iServer generalises concepts from hypermedia systems and enables the definition of links between different kinds of digital media as well as the integration of digital and physical content. Any link between two information entities is based on the abstract concepts of *resources* and *selectors*. The platform was designed to be extensible for new types of media based on a resource plug-in mechanism. By implementing media-specific instances of the resource and selector concepts, any new type of media can be integrated. Note that the general link functionality offered by iServer supports links with multiple sources as well as multiple targets. Through the generality of the underlying link metamodel [13], even links with other links as sources or targets can be defined. A user management component is integrated into the core of the system. In addition to managing user access rights, this component can manage information about user roles and profiles which in turn can be used as the basis for providing personalised and context-dependent delivery of links and resources. The delivery of information can further be controlled through iServer's layer concept. Links are associated with specific layers and these layers can be dynamically activated and deactivated as well as re-ordered which provides a flexible way of resolving the links to be activated as the result of a specific user action. The rapid prototyping of applications is supported through the re-use of existing interaction components based on the concept of active components mentioned in the previous section. Since iPaper is implemented using the resource plug-in mechanism offered by iServer, active paper areas can be linked to and from a wide range of physical and digital media including web pages, images, video, Flash movies, databases and RFID tags as well as application programs. More detailed information about iServer and the interactive paper framework iPaper can be found in [12].

The XCM extensible content management system was developed to support applications with the requirement of context-dependent content delivery [11, 3, 6]. In XCM, different notions of context can be used to achieve different facets of context-awareness. For example, in order to localise and internationalise content delivery, an application might define that the user's location and language are context. To implement multi-channel applications with XCM, it suffices to specify what characteristics of a content delivery channel should be considered in the adaptation process. By refraining from offering a predefined or in-built context model, XCM can support any notion of context as required by an application. A simple context representation based on sets of $\langle name, value \rangle$ tuples serves as an interface between an application and the content management system. Based on this context representation, adaptation in XCM is built on the concept of context-aware versioning. Each object in the system can have multiple versions, so-called variants, that represent the object in a defined context. At run-time, a matching algorithm is responsible for selecting the variant of the object that matches best to the current context of the system.

The content delivery process in XCM is defined based on the separation of content, view, structure and layout. In the first phase, the content objects referenced by the current request are gathered from an application database. With each content object, it is possible to associate a view object in the database storing the publishing metadata. If such a view object exists, XCM applies this view to the content object in

order to select the attributes to be included in the presentation and aggregate any additional information required from related objects. After applying the views, XCM consults the structure metadata that defines a hierarchy over the content objects in terms of a tree consisting of components and containers. Finally, this structure is rendered for its delivery to the client using layout metadata in the form of templates that are applied to the structure tree. XCM manages both data and metadata in databases that feature the concept of context-aware versioning. Thus, all four phases of the content delivery process can be made context-aware. For example, multi-lingual content delivery is achieved by storing language dependent versions of the content in the application database. Using the same approach, multi-channel applications are realised by defining template versions for each channel that is to be supported.

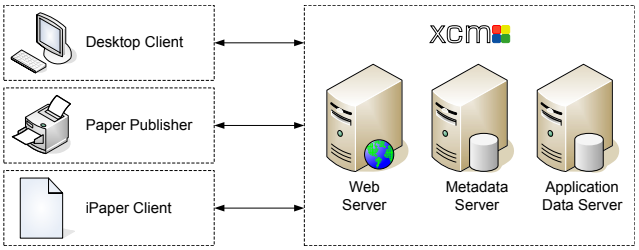

Figure 1: Architecture overview

By combining iPaper with XCM, we have created a platform that supports the production and operation of applications that use interactive documents as a user interface. An overview of the resulting architecture is shown in Figure 1. As can be seen from the figure, our platform is a client-server architecture with the context-aware content management system on the right-hand side and the various clients on the left-hand side. There are two clients and hence two delivery channels associated with paper. A paper publisher is used to produce the interactive documents, whereas the iPaper client is active at run-time when users interact with the system using the digital pen and paper. In addition to these channels, we also show a traditional desktop client with a web browser interface. While most applications nowadays offer content through this delivery channel in one way or another, it also illustrates the fact that, from the point of view of the content management system, the desktop client is no different from the paper channel. The context-aware content management system on the server side consists of three components. An extended web server handles incoming requests by generating responses based on the information managed by the metadata server. During response generation, the actual content that is delivered to the client is retrieved from an application database that is independent of the actual publishing process.

4. PUBLISHING

Having presented an overview of our approach in terms of an integration of interactive paper and content management technologies, we now introduce an example application that will serve as a scenario to explain the publishing process. Nowadays, television broadcasting companies such as the BBC often offer various supplementary materials along with the documentary shows that they produce. As an example, let us consider the series "The Blue Planet" [5] about ocean life that was originally transmitted in 2001. The BBC made the series available on DVDs and video tapes as well as publshing an accompanying book. In addition, the Blue Planet web site offered access to various quizzes, games and fact files. Further, the Open University developed a course on oceanography based on the series and a special course book with in-depth information on the topics presented in the documentaries. At the end of each section, the book indicates which parts of the documentaries the students should watch by indicating the relevant programmes and also timing information for specific topics. Questions are also provided to enable students to evaluate their learning progress. In this and the following section, we will show how our platform could be used to produce cross-media materials that could link the paper and digital materials in various ways to enrich the learning experience.

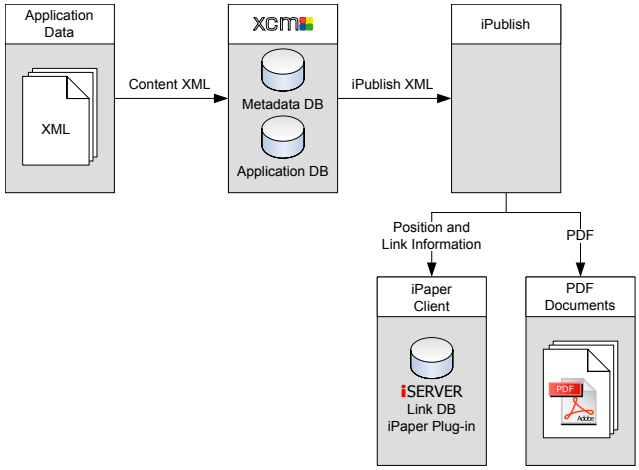

Figure 2: Publishing process

In the publishing phase, the interactive paper documents would be generated from the content publishing system that manages both the "The Blue Planet" data and the publishing data that provides access to information via standard web channels such as web browsers on desktop PCs or PDAs. Therefore, XCM would be responsible for both the publishing of information as PDF documents with defined active areas, and the publishing of information in a web channel in response to requests generated by activating areas within a printed document using a digital pen. The three main issues to be addressed in the production of an interactive paper document are:

- 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?

Figure 2 shows the interactive paper publication process outlining our approach. The publishing process is mainly based on three components: the extensible content management system XCM, iServer which maps interactions on

paper to requests to XCM and a third component, iPublish, responsible for publishing the paper documents based on information stored in XCM.

The first step is to structure the "The Blue Planet" data managed by XCM in a format suitable for publication as a paper document. We have defined a document model based on the concept of chapters, sections, sentences, words, tables and images. This enables us to 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 3.

```
<?xml version="1.0" encoding="UTF-8"?>
<document>
...
<chapter id="c6-fish_design">
  <body>
    ...
    <section id="s2-fish_shape">
     <paragraph>
      <text id="c6.s2.p3">
        The pectoral and pelvic fins are mainly used for
        manoeuvring, turning and stopping, while the
        ...
      </text>
      <image ilink="true" ac="get_info?i=i-23"
        id="get_info.i-23"src="fishFins.png"/>
     </paragraph>
     ...
    </section>
    ...
  </body>
</chapter>
...
</document>
```

Figure 3: iPublish XML document

The active elements within a document that will be linked to digital content by iServer via its iPaper plug-in are defined using a tagging mechanism. A tag describes the functionality to be linked and assigns a unique identifier to this particular piece of information which is then used to create the necessary active areas for iPaper. An example where an image has been tagged to provide additional information when pointed to with the digital pen is shown in Figure 4.

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 [8]. 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. The PDF file is then analysed to find any metadata tags and, for each tagged element, data about the corresponding active area defined by the element's surrounding shape (position and dimension) are extracted. Finally, 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 printing phase, we also have to choose a specific Anoto pattern license to make the paper documents interactive. While Anoto defines licenses for different page sizes,

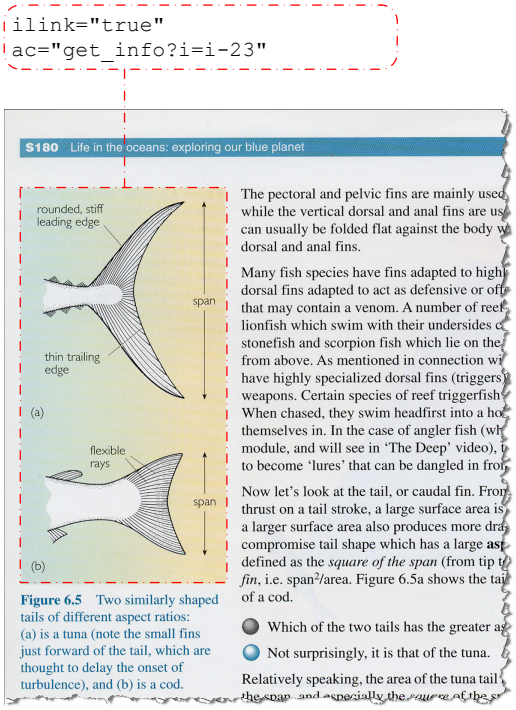

Figure 4: Document element tagging

all pages within a segment have the same size. Practically, this means that in order to have pages of different sizes, different licenses have to be used. As the dimensions of the pages increase, the number of available pages within a single license decreases, limiting the amount of pages which can be generated with a single license. To become more flexible in allocation of Anoto pattern space, we have developed a printer driver which is based on the Anoto PPGM library [2] and, acting as a virtual printer, enables the printing of documents of different sizes, the merging of multiple Anoto licenses, the definition of printing profiles and the use of black colour reduction. Together with a document database, it manages the Anoto pattern space, keeping track of the pattern already used. Furthermore, our printer driver may run best-coverage algorithms in order to cover, for example, a single A0 license page with up to 24 A4 pages as defined in the iPaper framework. The use of our virtual printer reduces the printing of interactive documents to a single step and provides direct access to Anoto-enabled printing functionality from any application.

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.

Successive pages defined in the augmented paper address space within the Anoto encoding space are arranged in a grid of a number of columns and a number of rows as shown in Figure 5. 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 ($pageHeight$). Op-

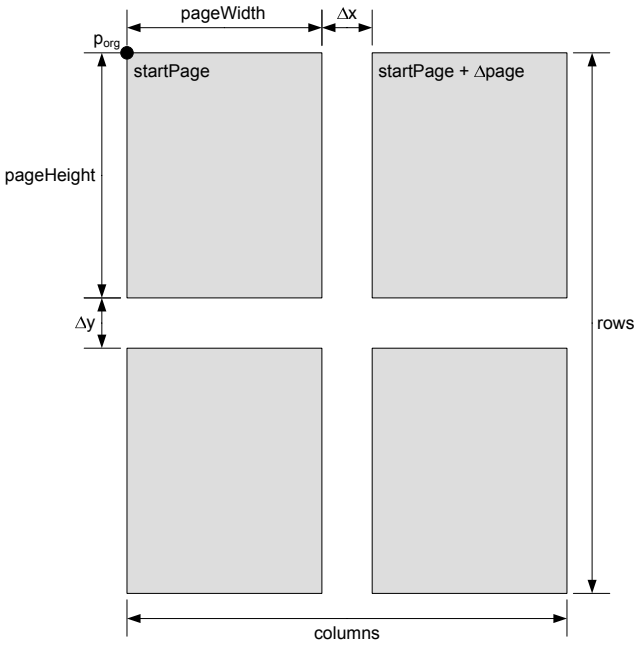

Figure 5: Page mapping

tionally, 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 Anoto-defined page, there is a reserved area to define special Anoto functionality such as paper-based buttons. Any position p_1 lying somewhere within the space covered by the matrix of document pages can be mapped automatically to the corresponding *documentID*, *page* and *position* of the interactive paper framework's address space. In contrast to our solution, an application based directly on Anoto technology does not cover a page with a unique pattern. Instead, for each paper-based button, 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.

Based on this document handler, we can register the different documents as shown in Figure 6. In this example, the Open University course book about life in the oceans (s180_life_in_the_oceans) is registered with the Anoto position (55651363, 2516592) as its origin. We further define the document size (210 by 263 mm) and the number of pages (152) that the document contains.

```
<?xml version="1.0" encoding="UTF-8"?>
<documents>
<document>
  <identifier>s180_life_in_the_oceans</identifier>
  <connectedPages>
   <startPage>1</startPage>
   <pageOffset>1</pageOffset>
   <origin>
    <point>
     <x>55651363</x><y>2516592</y>
    </point>
   </origin>
   <size>
    <width>210</width><height>263</height>
   </size>
   <columns>152</columns><rows>1</rows>
   <spacingX>189</spacingX><spacingY>0</spacingY>
  </connectedPages>
</document>
...
</documents>
```

Figure 6: Anoto pattern registration

5. INTERACTION

In the interaction phase, users either access the content of the application through a traditional web browser from a desktop computer or through the interactive paper client. Both channels communicate with the server using HTTP requests and responses. In the case of the web interface, these responses contain (X)HTML documents that can be rendered by the web browser. In contrast, when the digital pen is used as the input device, some other form of output device must be used to display the results. This could be some form of visual display used to present images, video, web pages etc. or possibly an audio device to deliver speech or sounds. In the latter case, this could be achieved by the server sending responses to the paper client containing VoiceXML [15] that is then transformed to speech by the client using text-to-speech technologies. With the Blue Planet scenario, one could even consider that visual and audio displays be used in different contexts. For example, audio might be used to provide supplementary information when a student is doing general reading, whereas a visual display would be used when they are working on exercises at their desk. In the various interactive paper applications that we have developed, we have used both visual and audio displays, and sometimes a combination of both. To highlight the possibility of supporting speech as well as the usual HTML browsers in applications and also the difference between the two interaction channels, we will assume an audio output device is used in conjunction with the digital pen as input in the remainder of this section.

While links are activated in the web browser interface by the user clicking on them with the mouse, this functionality is provided in the paper client by the digital pen. When a user touches the paper with the pen, the corresponding coordinates are sent to our cross-media link server (iServer) which determines whether the point is contained within an active area defined on paper. If so, iServer resolves the link to be activated and returns the target resources. For the sake of simplicity, we shall assume for the moment that there is a single target resource, although it is possble to have more than one. Depending on the resource, a renderer for the link target is then activated to "display" the content to the user. iServer imposes no restrictions on the kind of media that can be used as sources and targets for links, as long as a corresponding media plug-in is available and registered with the server. In our scenario, we will use paper as the source for the links and voice as their targets.

Before going into the details of the information services provided by the application during the interaction phase, we will first present the information model used to manage the information content. Figure 7 gives an overview of this conceptual model using the graphical notation of the OM data model [9], an extended entity-relationship model for object-oriented data management. As can be seen in the

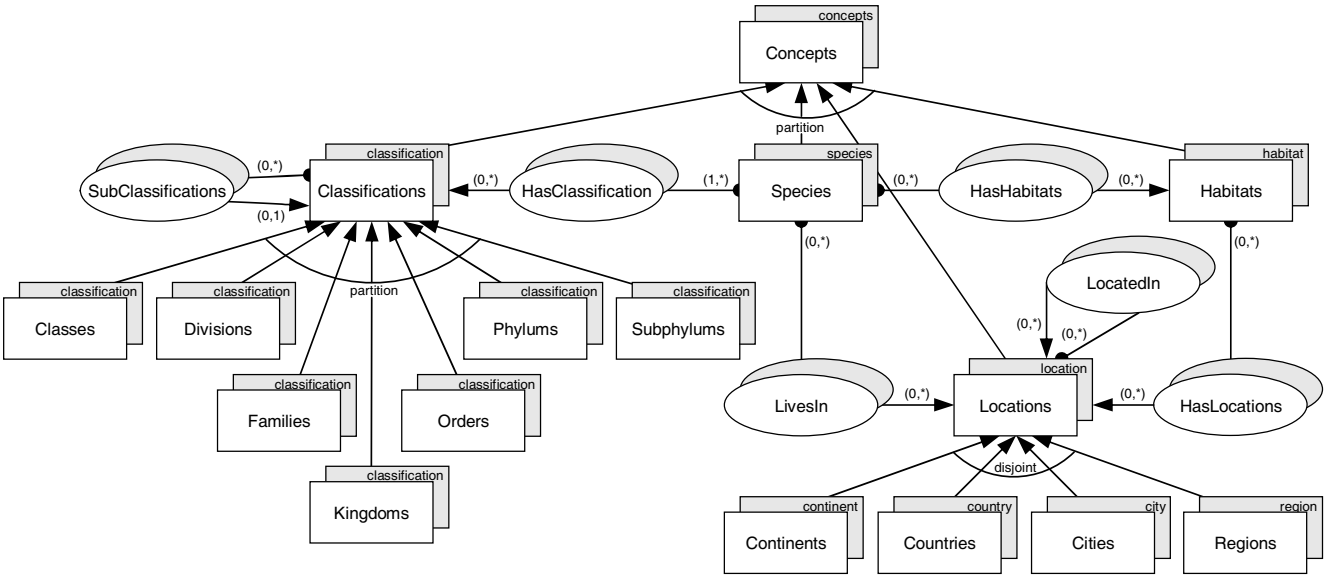

Figure 7: Conceptual application model

figure, the application domain consists of the information concepts relevant to the "Blue Planet" application. The set of `Concepts` is partitioned into `Species`, `Locations`, `Habitats` and `Classifications`. In the graphical notation of OM, such object collections are represented by rectangular boxes, whereas the relationships between them are captured by oval shapes. A relationship associates the members of a source collection with those of a target collection according to the associated cardinality constraints. For example, the `hasClassification` association relates species objects to classifications. The `(1:*)` cardinality constraint at the source collection of the association defines that each species object has to belong to at least one classification. Although the application database as presented in the figure provides an even finer classification of the application concepts than we have discussed here, we will not go into further details of the application model.

Within the "Blue Planet" application scenario, information concepts are stored using a database management system that provides context-aware versions as introduced in Section 3. As an example, we assume that our application offers a `getSpecies` information service that would allow additional information about a certain species to be retrieved and delivered to the user. Such an information service could be linked to any occurrence of the name of a species in the text or to pictures of the fish and mammals appearing in figures. Figure 8 shows how objects of type `species` are represented with context-aware object versions. As can be seen in the figure, the type defines three attributes, a name for identification, a URI pointing to content media and a description giving background information about the animal. The object shown in the figure has three context-dependent versions. As indicated by the context values associated with each variant, the first version contains English content with a high level of detail. The second is also in English but the level of detail is low. Finally, the third variant is in German and the level of detail has been specified as high. The first version is intended for desktop clients and therefore the media URI will point to a high-resolution video and the description will offer detailed information. In contrast, the media URI of the second variant points to a low-resolution image and the description is a short summary of all available information. The default version that will be returned if no context is specified by the client or if the matching process returns an ambiguous result is the first version as indicated by the solid line connecting it to the object.

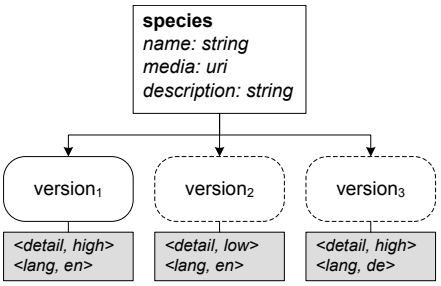

Figure 8: A content object with versions

At runtime, a desktop client would set the context to $C(S) = \{\langle format, html\rangle, \langle lang, en\rangle, \langle detail, high\rangle\}$. When the species object is accessed, the database management system uses this context to select the appropriate variant of the object. This matching is done by comparing the context specified by each version to the context of the client and assigning each version a score. As it is out of the scope of this paper to discuss how the context matching process works in detail, we will simply assume here that the version selected is the one with the most context values in common with the client's context. Full details of the process and all cases are given in [4].

Therefore, in the given example, the first version would be selected as it matches two out of three context values. However, a second example illustrates that this simple matching approach can also produce unsatisfactory results. Assume, for instance, that the paper client requests the German version of the content. To do so, it would specify the context

$C(S) = \{\langle format, vxml\rangle, \langle lang, de\rangle, \langle detail, low\rangle\}$. When this context is matched to the variants of the object, both the second and the third variants will be assigned the same score and thus the default variant is returned which does not match at all. To prevent situations such as this, a client can also declare certain context values as required or illegal. In the example, we could decide that to obtain the correct level of detail is more important than the language setting. This behaviour could be expressed by marking the context value denoting the level of detail as *required*. In the syntax of our approach this would mean to rewrite the corresponding value to $\langle detail, +low\rangle$. Prefixing a value with plus (+) marks it as required, while the minus (−) prefix marks it as illegal.

Another issue that needs to be addressed in this example is the fact that content delivery via voice cannot include images or movies. As all variants of the content object point to such media, we have to define a view in XCM that removes the corresponding attributes before rendering it for the client in the case of voice delivery. The resulting view object with its two versions is shown in Figure 9. The default version of the object handles the general case and thus does not define a specific context. The second variant addresses the case of voice deliveries and its context requires that the client sets the *format* context value to *vxml*. In the case of the default variant, the query attribute of the view object is left blank prompting XCM to return the species object as defined in the database. To remove the media attribute, the query attribute of the variant for voice delivery would be set to `"project (name, description) of $input"`. This query performs a projection of the object given as input to obtain its name and description value.

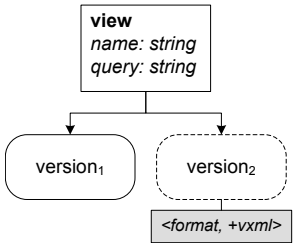

Figure 9: A view object with versions

In the interactive phase, structure is often not as important as during the publishing phase because few information services involve more than one content object. We therefore omit the description of this stage of the content publishing process and continue with the discussion of the final step. In XCM, this step involves the rendering of the content in a mark-up that is acceptable to the client. In the case of the desktop channel, the mark-up will be HTML, whereas VoiceXML has to be generated for the interactive paper client. During the first three steps—content aggregation, application of views and content structuring—of its content delivery process, XCM builds an XML document that contains all content data in the desired hierarchical structure. As each content and structure component can be associated with a layout object in the database used by XCM to manage publishing metadata, XCM can also generate an XSLT stylesheet parallel to the rendering of the content. In the final phase, the stylesheet is applied to the XML document and the result of the transformation is sent back to the client. Figure 10 shows how layout components are defined in the XCM publishing database. Apart from a unique name, a layout object has an attribute to hold a template. In most cases, this template corresponds to one `<xsl:template>` element in the final stylesheet. For each delivery channel supported by the application, a layout variant storing the corresponding template has to be defined. In the "Blue Planet" application, we have assumed that we use HTML and VoiceXML and thus the layout for the `getSpecies` information service has two versions annotated with the appropriate context values.

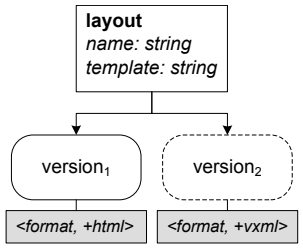

Figure 10: A layout object with versions

To conclude the example given in this section, assume that a user requests information about the species of Seahorses from the paper client by clicking on an image of a seahorse or on the word `seahorse` in the text. The XML document representing the content that would result from such a request is shown in Figure 11. It represents content information according to the XML schema used by XCM to process all requests. Each XML response consists of one `<webobject>` that contains a set of properties and optionally further nested web objects. The `<property>` elements correspond to the attributes of the content objects in the application database. Note how the application of the view as discussed above results in the omission of the property that would contain the media URI. XCM inserts a `stamp` attribute into every web object that it renders in XML. This attribute is the basis for the automatic generation of match clauses in the XSLT stylesheet that is created in parallel.

```
<webobject stamp="we0" oid="o279"
  type="view:getSpecies">
<property name="sys:name">
  <string>getSpecies</string>
</property>
<property name="name">
  <string>Sea Horse</string>
</property>
<property name="description">
  <string>Seahorses are a species of fish belonging
   to the fish family Syngnathidae, which also includes
   pipefish.</string>
</property>
</webobject>
```

Figure 11: XML document of getSpecies

The XSLT template that is chosen for the `getSpecies` information service in the context of the interactive paper client is shown in Figure 12. As can be seen in the figure, the template consists of one `<xsl:template>` element that will generate a VoiceXML prompt when applied to the previously discussed content XML. Note that the developer is not required to specify a match condition and therefore the attribute `match` of the template is set to an empty string.

When including templates in the stylesheet during the processing of a request, XCM automatically adds the required match conditions. In our example, the match condition `webobject[@stamp='we0']` would be inserted. If one template is used to transform multiple web objects, XCM generates a complex matching condition expressing a conjunction of such simple conditions.

```
<xsl:template match="">
<vxml version="2.0" xmlns="http://www.w3.org/2001/vxml">
  <form id="xcmprompt">
   <block>
    <prompt>
     <xsl:value-of
       select="/property[@name='description']/string"/>
    </prompt>
   </block>
  </form>
</vxml>
</xsl:template>
```

Figure 12: XSL template for getSpecies

As soon as the content XML and the XSLT stylesheet have been generated, they are passed to the XSLT transformer on the web server component of XCM. By applying the stylesheet to the XML document, the transformer generates the final content representation that is then sent back to the client by the web server. On the client, the response is interpreted and rendered. In our example, the VoiceXML file is passed to the text-to-speech engine causing it to read out the content of the description attribute of the original content object.

6. CONCLUSIONS

We have shown how a content publishing approach can be used to support both the production and use of interactive paper documents that rely on emerging digital pen technologies to link to digital content and services. Specifically, we have described how paper was introduced as an additional channel into the XCM multi-channel content management system by coupling it with the iPaper framework for interactive paper applications. In addition, the flexibility of the XCM system used, enabled us to support the actual process of publishing the interactive documents in a first phase by generating both the PDF files and the cross-media link definitions automatically.

7. REFERENCES

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

[2] Anoto AB. Paper SDK Specification and Description. Technical Report 410 045, 2006.

[3] P. Barna, G.-J. Houben, and F. Frăsincar. Specification of Adaptive Behavior Using a General-Purpose Design Methodology for Dynamic Web Applications. In *Proceedings of Adaptive Hypermedia and Adaptive Web-Based Systems (AH 2004)*, Eindhoven, The Netherlands, August 2004.

[4] R. Belotti, C. Decurtins, M. Grossniklaus, M. C. Norrie, and A. Palinginis. Interplay of Content and Context. *Journal of Web Engineering*, 4(1):57–78, 2005.

[5] The Blue Planet, http://www.bbc.co.uk/nature/blueplanet/.

[6] S. Ceri, F. Daniel, M. Matera, and F. M. Facca. Model-driven Development of Context-Aware Web Applications. *ACM Transactions on Internet Technology*, 7(2), 2007.

[7] Fly Pentop Computer, LeapFrog Enterprises, http://www.flypentop.com.

[8] F. Guimbretière. Paper Augmented Digital Documents. In *Proceedings of 16th Annual ACM Symposium on User Interface Software and Technology (UIST 2003)*, Vancouver, Canada, November 2003.

[9] M. C. Norrie. An Extended Entity-Relationship Approach to Data Management in Object-Oriented Systems. In *Proceedings of 12th International Conference on the Entity-Relationship Approach (ER '93)*, Arlington, USA, December 1993.

[10] M. C. Norrie, B. Signer, and N. Weibel. General Framework for the Rapid Development of Interactive Paper Applications. In *Proceedings of 1st International Workshop on Collaborating over Paper and Digital Documents (CoPADD 2006)*, Banff, Canada, November 2006.

[11] J. Pascoe, N. Ryan, and D. Morse. Issues in Developing Context-Aware Computing. In *Proceedings of International Symposium on Handheld and Ubiquitous Computing (HUC '99)*, Karlsruhe, Germany, September 1999.

[12] B. Signer. *Fundamental Concepts for Interactive Paper and Cross-Media Information Spaces*. PhD thesis, ETH Zurich, May 2006. Dissertation ETH No. 16218.

[13] B. Signer and M. C. Norrie. As We May Link: A General Metamodel for Hypermedia Systems. In *Proceedings of 26th International Conference on Conceptual Modeling (ER 2007)*, Auckland, New Zealand, November 2007.

[14] B. Signer, M. C. Norrie, M. Grossniklaus, R. Belotti, C. Decurtins, and N. Weibel. Paper-Based Mobile Access to Databases. In *Demo Proceedings of ACM International Conference on Management of Data (SIGMOD 2006)*, Chicago, USA, June 2006.

[15] VoiceXML Forum, http://www.voicexml.org/.

[16] R. B. Yeh, S. R. Klemmer, A. Paepcke, M. Bastéa-Forte, J. Brandt, and J. Boli. Iterative Design of a Paper + Digital Toolkit: Supporting Designing, Developing, and Debugging. Technical report, Stanford University, 2007.