General Framework for the Rapid Development of Interactive Paper Applications

Moira C. Norrie
Dept. of Computer Science
ETH Zurich
8092 Zurich, Switzerland
norrie@inf.ethz.ch

Beat Signer
Dept. of Computer Science
ETH Zurich
8092 Zurich, Switzerland
signer@inf.ethz.ch

Nadir Weibel
Dept. of Computer Science
ETH Zurich
8092 Zurich, Switzerland
weibel@inf.ethz.ch

ABSTRACT

We present a component-based framework that supports the rapid development of a wide variety of interactive paper applications. The framework includes authoring and publishing tools as well as a server that supports the linking of active areas on paper to a wide range of different media types and services.

1. INTRODUCTION

The Anoto digital pen solutions were originally developed for the capture of handwritten information and 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 have seen other more general notions of interactive paper documents and applications evolve based on real-time streaming of the data. For example, the Fly pentop computer provides a number of interactive desktop applications where the user can, not only interact with the applications via the pen, but also draw their own interfaces. The PaperPoint application [8] based on an Anoto pen with streaming functionality allows PowerPoint presentations to be controlled and annotated based on a printed overview of the slides. These new technologies open up a whole new range of possibilities for experimentation with interactive documents, paper-based interfaces to applications and general forms of bridging the physical/digital divide.

It is therefore vital that the necessary infrastructure and tools are available to support the rapid development of a wide variety of applications and facilitate experimentation with alternative forms of interaction and design. It should be possible to compose complex applications from existing components and to be able to easily integrate different kinds of media, information sources and application services. Not only desktop applications should be supported, but a range of possible input/output channels that could be combined with interactive paper in interesting ways. Last but not least, tools are required to support both the authoring and printing of interactive paper documents.

In the context of the European projects Paper++ [6] and PaperWorks [7], we have developed such a framework and demonstrated its flexibility by implementing a wide variety of applications including interactive paper maps and brochures, paper-based interfaces to a range of applications, for example PowerPoint and an image retrieval service, interactive notebooks for lab-based experiments and also interactive tabletops. Interactive documents can be created on the fly with links generated automatically as demonstrated by Print-n-Link [5], a system that allows users to search for PDF articles on the web and print interactive versions of them with links to information about citations and document retrieval services. The framework was developed to support any type of technology that can track user actions on paper including a prototype pen based on conductive ink [4] developed within the Paper++ project. We have also had access to a prototype version of an Anoto pen with streaming functionality and therefore already have a great deal of experience with the development of interactive paper applications based on Anoto technologies.

In this paper, we discuss the requirements of such a framework and present the main features of our solution. We start in Section 2 with a description of the software tools and printing solutions offered by Anoto. Section 3 then presents iServer and iPaper which together provide a component-based server architecture for interactive paper applications. Section 4 describes the authoring and publishing tools that we have developed to support the generation and printing of interactive paper documents. Concluding remarks are given in Section 5.

2. ANOTO FUNCTIONALITY

The Anoto Digital Pen and Paper technology [1] was originally aimed at the digital capture of handwriting and sketching on paper. One of the main business applications to date involves the automatic capture of form data in large organisations such as insurance companies and in healthcare. The form documents first have to be digitally authored and then they are printed along with the Anoto pattern which is read by an infrared camera integrated in the digital pen to track the movement of the pen across the paper. The position data is transmitted to a computer either wireless, via Bluetooth, or by means of a docking station connected directly to the computer. Once on the computer, the data is processed either by Anoto software or specialised application software such as the Forms Automation software by Hewlett-Packard.

In order to use its technology, Anoto provides a range of developer tools. We describe them in this section and

highlight some problems and limitations of the Anoto tools when presenting our general framework for interactive paper solutions in Section 3 and Section 4.

The Anoto License Model: In order to be able to use Anoto-enabled documents, a developer needs an Anoto License. Licenses confirm to a strictly defined model. The Anoto pattern space is divided into pattern pages. Pages are grouped into books, which in turn are grouped into shelves. Every shelf belongs to a segment. Licenses are issued at the level of page, book or shelf. A license has a pre-defined validity and Anoto ensures that a licensed pattern space will not be used by anyone else until the license expiration date.

The Forms Design Kit: The Anoto Forms Design Kit (FDK) allows graphical designers and software developers to build forms to be used with Anoto functionality. The main part of the FDK is the Forms Design Tool (FDT), a plug-in for Adobe Acrobat that allows the generation of Anoto-enabled documents. By using the FDT, a designer can take an existing PDF document and add *pidgets*, *user areas* and *properties* to the form layout, and then generate a PostScript file containing the pattern specified by the Anoto License in use.

Pidgets are used by Anoto for enabling the interactivity from paper to the computer. There exist pidgets to inform the application that an interaction has been started on a new page and pidgets to signal the end of the interaction. These pidgets are composed by a special icon and a special pattern defined elsewhere in the licensed space. The insertion of a pidget actually means pasting a piece of pattern and replacing the underlying page pattern. In addition to the PostScript file, FDT creates a Paper Application Definition (PAD) document containing the specification of all pidgets and active areas to be used at runtime within the application service.

The Paper SDK: To overcome some of the limitations of the FDK and to give greater flexibility to developers, Anoto recently made available the Paper SDK. This product includes the PAD and Print Generation Module (PPGM) which was used to build the FDT. The PPGM is an MS Windows C library enabling software developers to access core Anoto functionality such as the generation of the pattern or advanced functionalities such as colour reduction or printing profiles. It can be integrated into applications and allows developers to build add-ons for any type of specific authoring tool used by the designers.

The Software Development Kit: Once an Anoto-enabled document has been provided, the Anoto SDK can be used to build the application. The SDK provides a *Pen API*, which is a basic framework for server-based applications (Java servlets or Microsoft ASP). Additionally, by using the *Service API*, Anoto provides a solution for stand-alone applications.

The Anoto Pens: Anoto-enabled pens are available from various manufacturers (Logitech, Maxell, Nokia) and they are all optimised for the existing Anoto capture applications (e.g. form filling). More interactive applications require functionality, for example for user feedback, which is not yet available on these pens. Even though the pens are equipped with LEDs and integrated vibration functionality, these feedback mechanisms are not accessible by any software API. Projects such as [3] tried to overcome these problems by coupling the pens with other devices. In the case of the PaperPoint application, we have also coupled a pen with a laser pointer to avoid the problem of the speaker having to use two different devices during their PowerPoint presentations. While such solutions allow one to experiment with the possibilities of different functionalities being integrated into a pen, clearly they remain at the prototype level and are far from optimal. Generally, a number of issues with regard to pen design arise from the development of more interactive applications where the pen is used not only as a writing device but also a pointing and selection device. The Fly pentop computer 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.

3. ISERVER AND IPAPER

The Integration Server (iServer) is a main component of our framework for interactive paper applications and enables cross-media linking between arbitrary physical or digital resources. It provides a set of concepts for link definition and a Java framework to create and activate cross-media links. Links within the iServer framework are always bidirectional and directed, which means that they have at least one target and one or more source *entities* (multi-headed/multi-tailed links). Links can not only be defined between entire entities (*resources*) but also between parts of resources addressed by the abstract concept of a *selector*. By providing specific implementations (plug-ins) for the resource and selector concepts, new types of resources can be added to the cross-media information platform. However, the general link server functionality, including user management, multi-layered links etc., is defined on the iServer level and can be shared and reused by any iServer resource plug-in. More details about iServer, including a full specification of the general link model, can be found in [8].

As part of the European project Paper++ we have developed an iServer plug-in for interactive paper (iPaper). Based on the concept of documents and pages as well as different forms of shapes (rectangles, polygons etc.) for the definition of active areas within a page, links may be defined from an active paper area to any other iServer resource. An active paper area may also be the link target of an iServer link. Note that the iPaper plug-in is general and does not depend on any specific pen technology (e.g. Anoto). The only input required is a *document identifier*, a *page number* and the *(x,y) position* with a given page as shown in Figure 1.

This brings us to the second part of the iPaper client-server architecture, the *iPaper Client*. The iPaper Client is responsible for communicating with a hardware device and transforming the captured data into the neutral *document*, *page* and *(x,y)* format to be handled by the iPaper plug-in on the server side. Therefore, the iPaper Client is based on a set of interfaces that have

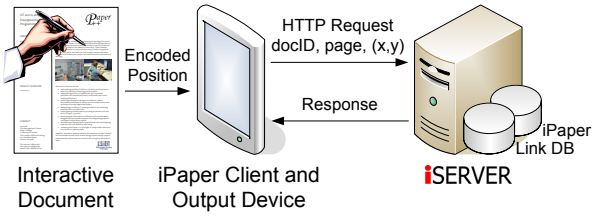

Figure 1: iPaper client-server architecture

to be implemented for any device to be used together with the iPaper architecture. The introduction of these interfaces brings flexibility in supporting different types of input technologies. Within the Paper++ project we have implemented a device driver for an inductive pen prototype. However, it was easily possible to add support for Anoto's Digital Pen and Paper functionality at a later stage. Our iPaper driver for Anoto pens is based on direct processing of binary COM port streaming data and does not make use of any Anoto SDK functionality.

Since the iPaper plug-in does not depend on a specific pen technology, all existing iPaper applications could immediately also be controlled by the iPaper driver for Anoto. It is even possible to control a given interactive paper application from different pens (e.g. inductive pen and Anoto pen) at the same time. Basically, any technology that allows the position within a document to be tracked can be used as an interaction device for our interactive paper platform (e.g. ultrasonic tracking as used by the Mimio whiteboard solution or camera-based tracking systems). Note that the use of the iPaper architecture in combination with Anoto pens introduces some form of flexibility when it comes to the definition of active paper areas. While Anoto's FDK only supports the definition of rectangular shapes, iPaper supports a variety of shapes, including arbitrary polygons. It is further possible to define an active area by composing a set of shapes to form a *complex shape*. In addition to the single-layered (non-overlapping) shapes supported by Anoto's FDK, our solution also supports the concept of multi-layered active areas to control the link granularity. Note that this flexibility in the design of active areas becomes important in the design of interactive paper solutions with a high link density. Furthermore, none of our active areas are based on the "copy/paste" that is used by Anoto pidgets as described in the previous section as this may result in layout problems since pidgets always require a few millimetres distance between them.

The specific Anoto driver also deals with the mapping of an Anoto license to the corresponding iPaper coordinates. Again we are flexible in the way that our approach is not based on the concept of pages as defined in the Anoto license model. By using our own mapping algorithm, we can for example use parts of the pattern space of a single Anoto A0 page to cover multiple A4 pages defined by the iPaper framework.

In addition to the automatic authoring of interactive paper applications, described in the next section, we support the definition of links in XML format as well as based on the iServer cross-media authoring tool. Thereby, special attention has been paid to the definition of active areas that have to be repeated on multiple pages (e.g. page header and footer functionality). With the FDK, these elements have to be manually repeated on each page. Our iPaper framework supports the concept of *templates*. A template contains a set of shapes and may be applied to a set of pages or to entire documents. By defining the shapes only once (in the template) it becomes easier to make changes to elements that appear on multiple pages. At the same time, by eliminating any redundancy, we can reduce the space that is required to store this information in iServer. Note that, in the authoring process, active paper areas may be linked to any media supported by an iServer resource plug-in. At the moment we have plug-ins for web pages, movie clips, Flash movies and other resources.

In addition to these links to rather "static content", iServer introduces the concept of *active content* represented by active components that can be used as link source or target. An active component is basically a piece of program code that gets executed when the corresponding link has been selected. After an active component has been activated, its specific program code is loaded on the client and on the server side and the active component becomes the handler for any information, for example coming from a digital pen, until it is terminated. The active component concept has proven to be very effective since a developer can focus on the functionality to be handled by a single active component resulting in a component-based architecture where specific functionality is encapsulated in small active components. For the iPaper plug-in we have for example developed active components representing paper-buttons, paper-based captured areas, paper-sliders and many others. Active components can be reused across different applications and the growing set of active components tremendously supports the rapid prototyping of new interactive paper applications since a developer may choose from a rich set of existing active components.

4. PUBLISHING INFRASTRUCTURE

Our iDoc publishing framework brings together three different types of authoring: *database-driven authoring*, *automatic authoring* based on the analysis of a given document and *manual authoring* as provided by the Anoto FDT. These different types of authoring are integrated into iDoc via a plug-in mechanism of the Semantic Mapper component shown in Figure 2.

iDoc is based on three main components: the *Semantic Mapper*, which is responsible for mapping a position within a paper document back to its digital representation, the *Printer Driver*, which enables flexible printing of interactive paper documents and the *Document Database*, which stores all the information about the printed documents, such as the pattern used and the position of the different document elements. The Semantic Mapper plug-ins enable a flexible definition of the paper to digital mapping.

By using the iDoc publishing framework provided, some limitations of the Anoto FDT and Paper SDK can be removed. Since the FDT is a plug-in for Adobe Acrobat, it can be used only within that application.

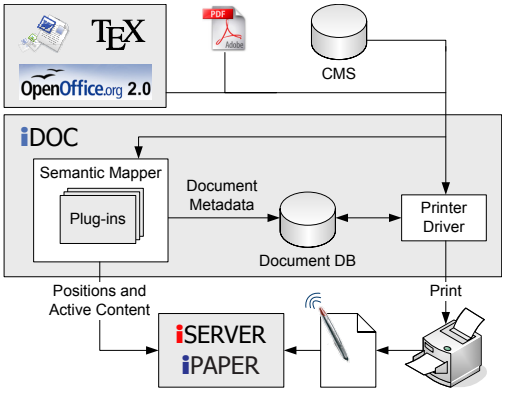

Figure 2: iDoc publishing framework

Furthermore, Adobe plug-ins are only available in Acrobat Professional and the FDT is tightly coupled with a GhostScript library for Windows. As a result, the authoring process is typically split into two separate steps based on two different tools: developers may use their favourite authoring tools (e.g. MS Word, Adobe Illustrator or CorelDRAW) for the document design, but they must switch to Adobe Acrobat to enable the Anoto functionality. iDoc addresses this problem by providing plug-ins for different authoring tools. In order to automatically track the position of single document elements and enable a mapping from the paper version back to the corresponding digital component, we are currently investigating plug-ins for MS Word, OpenOffice Writer and CAD systems. If an Anoto document becomes very complex in terms of the quantity and layout of user areas, authoring based on the Anoto FDT may become difficult. Since the PAD file is an XML file, developers may directly edit this XML document to overcome these problems. However no tools are provided for the automatic authoring of complex documents such as the interactive event brochure and map that we used in the EdFest project [2]. The database-driven authoring of the EdFest brochure was based on the *iPublish* plug-in. The position of all active areas was automatically calculated based on a PDF version of the brochure created by our content management system (CMS). It was therefore possible to automatically create a PDF from the CMS and, at the same time, to publish information about the position of active areas and the active components to iServer. Furthermore, we have plug-ins that automatically detect and calculate the position of document elements based on specific patterns. An example of this class of services is the Print-n-Link plug-in mentioned earlier.

While Anoto defines licenses for different page sizes, 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. Moreover, the space is partitioned into streaming and non-streaming pattern. The streaming pattern allows some of the new Anoto pens to transmit real-time information from the pen to the computer via Bluetooth. In order to use the streaming functionality, a special streaming license is required. The PPGM library provided by Anoto tries to overcome some of these problems. The functionality which is available does not go much further than the one found in the FDT and the dependency on GhostScript libraries remains an issue. Since the PPGM offers rather low-level Anoto functionality, a tool offering a high-level interface is required. Nevertheless, because of its compactness, it is convenient to use the PPGM library as a starting point for an authoring and publishing tool with a more flexible and complete interface. Our printer driver is based on the Anoto PPGM 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 the 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 one single step and provides direct access to Anoto-enabled printing functionality from any application.

5. CONCLUSIONS

We have presented the state of the art of existing Anoto tools for developing interactive paper applications. While most of these tools mainly focus on supporting the development of form filling applications, we introduced our general framework for interactive paper applications and compared it to Anoto's solution. We have outlined how our publishing infrastructure supports the automatic authoring of interactive paper documents and introduced the concept of active components enabling the rapid component-based development of interactive paper applications.

6. REFERENCES

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

[2] EdFest Project, http://www.edfest.ethz.ch.

[3] C. Liao, F. Guimbretière, and C. E. Loeckenhoff. Pen-top Feedback for Paper-based Interfaces. In *Proc. of UIST '06*, Montreux, Switzerland, Oct. 2006.

[4] P. Luff, C. Heath, M. C. Norrie, B. Signer, and P. Herdman. Only Touching the Surface: Creating Affinities Between Digital Content and Paper. In *Proc. of CSCW 2004*, Chicago, USA, Nov. 2004.

[5] M. C. Norrie, B. Signer, and N. Weibel. Print-n-Link: Weaving the Paper Web. In *Proc. of DocEng 2006*, Amsterdam, The Netherlands, Oct. 2006.

[6] Paper++, Disappearing Computer Initiative, EU Research Project, IST-2000-26130.

[7] PaperWorks: Interweaving Paper and Digital Documents, EU Research Project, FP6-516895.

[8] B. Signer. *Fundamental Concepts for Interactive Paper and Cross-Media Information Spaces*. PhD thesis, ETH Zurich, 2006.