Letras: An Architecture and Framework For Ubiquitous Pen-and-Paper Interaction

Felix Heinrichs, Jürgen Steimle, Daniel Schreiber, Max Mühlhäuser
FG Telecooperation, Department of Computer Science
Technische Universität Darmstadt
Hochschulstr. 10, 64289 Darmstadt, Germany
felix_h,steimle,schreiber,max@tk.informatik.tu-darmstadt.de

ABSTRACT
Paper remains a prevalent medium in mobile usage contexts due to its inherent flexibility and robustness. Mobile computing solutions begin to provide powerful and convenient functionality, while the gap between paper documents and digital applications remains unbridged in mobile settings. Current toolkits do not offer adequate support for development of mobile pen-and-paper based applications, as they lack support for important mobile characteristics of real paper: user mobility and document mobility. To overcome their limitations, we present a novel generic architecture, along with its reference implementation *Letras*, a light-weight, freely available infrastructure to develop pen-and-paper based applications in mobile settings.

ACM Classification Keywords
H.5.2 Information Interfaces and Presentation: [Input devices and strategies]

General Terms
Human Factors, Design, Experimentation

Author Keywords
Development Tools / Toolkits / Programming Environments, Pen and Tactile Input, Ubiquitous Computing / Smart Environments, Handheld Devices and Mobile Computing, Digital Pen, Anoto

INTRODUCTION

Traditional paper remains a prevalent medium in many domains of our daily lives, ranging from tasks such as knowledge work to seemingly trivial tasks as organizing shopping trips. Capturing, storing and processing information by means of pen and paper may seem archaic in times of email and word processors. However, it has been shown that paper is not easily replaceable by digital systems due to its unique affordances [6], it rather coexists with digital systems in current work practices. Paper is a truly ubiquitous, highly mobile medium: Paper artifacts are ubiquitous in our daily lives, from the handwritten grocery list to a blank sheet of paper serving note-taking purposes in a meeting, from the scientific paper printout to the postcard sent to friends. Particularly in informal and mobile settings, such as note-taking, pen and paper remain the prevalent tools for capturing and storing information [2]. In order to reconcile the advantages of pen and paper and digital information processing (i.e., to bridge the digital-physical gap), several systems successfully employed digital pen technology [12, 3, 8, 10], thereby shaping a new style of human computer interaction: *Pen-and-Paper Interaction* (PPI).

However, the design of existing PPI based applications, and consequently the design of toolkits and supporting infrastructures, are based on the assumption that all parts of the system (pen, paper artifacts, computer hardware, application software components) are deployed and controlled by a single authority. Furthermore, deployment of the system is not explicitly supported nor facilitated by available toolkits. We argue that these assumptions obviate *Pen-and-Paper User Interfaces* (PPUI) to support two important characteristics of "real" pen and paper:

User Mobility Pen and paper use is pertinent in mobile and informal settings. In these settings, a main benefit of pen and paper compared to digital applications is the easy deployment and set-up: The user can start immediately to work with a brought-along sheet of paper. This is thwarted by the lack of deployment and set-up support in existing toolkits.

Document Mobility Paper documents are passed on by individuals and organizations, *intra-organizational*, e.g., a workflow document is handed from one department in a company to another, as well as *inter-organizational*, e.g., a leaflet is distributed from a retailing company to customer households. Supporting these practices with PPUIs requires that the applications can somehow *travel* with the documents they belong to.

Consequently, the bridge across the digital-physical gap as provided by existing toolkits is incomplete. As a solution, we propose the novel concept of *Ubiquitous Pen-and-Paper Interaction* (UbiPPI), which augments existing PPUIs with support for *document mobility* and *user mobility* to allow for ubiquitous use of PPUIs. Our contribution towards realiz-

EICS'10, June 19–23, 2010, Berlin, Germany.

ing UbiPPI is twofold: We present a novel distributed interaction processing pipeline architecture which better supports UbiPPI compared to the monolithic architecture used by existing PPI toolkits. Second, we present *Letras*, a reference implementation and generic application development platform based on this architecture. *Letras* provides a distributed, highly flexible approach towards system support for pen-and-paper based interaction in ubiquitous computing settings. It is available for free download under the Mozilla Public License[1].

BACKGROUND AND RELATED WORK

At the very basis, PPI means user interaction is realized by using a digital pen on paper for writing, drawing, etc. [3, 12, 8, 10]. A variety of technologies for implementing PPUIs are available. The most often employed technology as of today is the Anoto[2] digital pen and paper technology. It is widely used, because it preserves many of the usability aspects of traditional paper (e.g., no extra calibration, high resolution). This technology requires a special dot-pattern to be printed on paper, which is scanned by a small camera built into the pen and allows to calculate the pen position on a global 2D coordinate system. The digital pen records movements and other actions (e.g. clicks) on the paper surface. Data recorded by the pen is commonly referred to as *digital ink*. It allows applications to either produce a facsimile of the written ink, or to *interpret* the movements, e.g. using gesture recognition.

We can distinguish two modes of operation for the pen, which are fundamentally different from an interaction perspective. The digital pen operates either in *batched mode* or *interactive mode* [13]. In batched mode, pen movement data is recorded and stored in the pen. Upon user request (e.g., when the user plugs the pen into a cradle) the stored data is transferred to the application in bulk. In interactive mode, pen data is continuously streamed to the application and allows the application to provide instant feedback to the user.

Developing applications by directly using raw digital ink data is not convenient: toolkits which process this low-level data and expose more powerful abstractions to the developer are needed [13]. The most important abstraction commonly provided by existing PPI toolkits is the *interactive region*. On the one hand, an interactive region is a physical region on paper (or other surfaces as in [10]). On the other hand, applications can attach event handlers to interactive regions. The toolkit internally processes the low-level data, dispatching it to handlers, potentially aggregating and translating the data during the process.

Existing System Support for UbiPPI

User mobility means that users employ pen and paper on the move and in continuously changing settings, i.e., at their office desk, in the train, on the sofa at home and in the kitchen. Paper supports user mobility out of the box, as it is robust, light-weight and self-sustainable. Existing systems supporting user mobility focus on batched mode, e.g. ButterflyNet [12]. Using this approach, the application only needs to be deployed in one place, e.g., on the office computer, which allows to process the captured data when the mobile setting has been left. This approach is not suitable in a UbiPPI setting which implies the user can access a multitude of applications while being on the move, and further that these applications can provide real-time feedback to the user. Thus enabling the *interactive mode in mobile scenarios* becomes crucial for supporting UbiPPI. This in turn requires that the system components can be *flexibly deployed* on the available computing resources.

Document mobility implies that multiple heterogenous PPI based applications exist. Users use their pens to interact with multiple documents simultaneously, switching back and forth as their current task requires. Relating use of a digital pen to a single PPI based application does not hold anymore: *simultaneous pen usage on different applications* is required. Document mobility also means that users interact with encountered documents, i.e. documents they did not posses beforehand. Thus, it requires a flexible distribution of the knowledge on existing interactive regions. This is not problematic as long as the system is only used within one single organization which issues all artifacts. In real world, however, artifacts are issued by many different organizations. Hence, a huge number of PPI based applications and interactive artifacts (e.g. documents or surfaces) need to integrate to form a cohesive whole: a highly scalable and distributed naming service, a *distributed region lookup system*, is needed to identify and inform responsible applications or services, comparable to *AutoId*[3] or the *Domain Name System* (DNS) for interactive regions.

Current infrastructures for PPI based applications, such as PADD [3], iServer and iPaper [8] or CoScribe [10], focus on attaching digital ink recorded at physical instances of documents to their digital counterparts. Additionally, these approaches facilitate document editing in the physical domain, to allow a more natural interaction with documents. These systems consist of a monolithic client side processing approach, backed by a document server hosting the digital document instances. Flexible deployments and the usage of one pen across different applications are not supported. Interactive mobile use is limited to fixed scenarios, as the mobile access to databases [9].

Currently there are several commercial toolkits available to support the development of PPUIs. Anoto provides an SDK focused on form processing. Livescribe[4] also offers an SDK to support their Pulse Smartpen. The applications are deployed on the pen itself and it does not support the interactive processing mode in terms of communication with the environment. These toolkits neither support flexible deployments nor the possibility to use the same digital pen interactively in mobile scenarios. An alternative presents the freely available *PaperToolkit* [13]. It provides access to digital

[1] http://leda.tk.informatik.tu-darmstadt.de/Letras
[2] http://www.anoto.com
[3] http://www.autoidlabs.org
[4] http://www.livescribe.com/

pens in both, batched and interactive mode, and introduces an event based architecture comparable to typical graphical user interfaces toolkits (as e.g. Java Swing). However, it does not support the flexible run-time deployment required to adapt to dynamically changing environments. It also lacks support for the dynamic distribution of knowledge on available interactive regions as required to support document mobility.

A DISTRIBUTED PROCESSING PIPELINE FOR UBIPPI

Supporting pen-and-paper interaction requires processing of movement data captured by digital pens. Existing PPI based applications successively transform this data, from raw sensory information (e.g. pen tip at position x,y) to higher-level data constructs (e.g. "checkmark" gesture performed). Successive transformation of data implies a pipeline architecture, consisting of a sequence of processing stages. Although the pipeline processing approach is common among PPI based applications, there is no shared use of processing resources. Each application replicates the necessary processing stages, using a monolithic approach as shown in Fig. 1 a). Apart from being a waste of resources, this setup makes it difficult to support document mobility, i.e., to use the same digital pen in multiple applications simultanously. A pen connection typically needs manual set-up (i.e., bluetooth pairing of the pen). Such a set-up severly limits the deployment flexibilty, required to support user mobility in changing environments.

In order to overcome these limitations, we propose a generic pipeline architecture which supports sharing of pipeline components among applications. It is depicted in Fig. 1 b). Processing stages are decoupled, using clearly defined interfaces, so called *processing stage interfaces*. They can be deployed independently, indicated by a gray box around each processing stage. Applications may share stages of the pipeline. We distinguish four elementary transformation steps for the processing of digital ink. These steps yield elementary processing stages constituting the elements of the proposed pipeline architecture:

Driver Stage The driver stage connects the digital pen hardware to the rest of the pipeline. It provides appropriate hardware abstractions to higher-level processing stages, establishes data connections via appropriate bus or network protocols and transforms data into low level data structures. Typically the data gathered in this stage consists of samples describing sensory information, for example pen tip position, or elementary events such as "pen tip down".

Region Processing Stage Interactive regions provide the basic link between digital functionality and physical paper and thus constitute the most important building blocks of PPI based applications. Each PPI based application defines at least one interactive region as area of interest, although typical applications use more complex region hierarchies (e.g. printed buttons contained within a writing region). The region processing stage relates digital ink data to the interactive regions defined by the applications and channels it to successive processing components.

Semantic Processing Stage The next step transforms digital ink data to common higher-level semantic structures. Such transformation depends on the specific needs of a PPI based application. Examples for semantic processing are segmentation of digital ink into text and drawings, handwriting recognition or gesture recognition.

Application-Level Processing Stage Finally, the digital ink data and the results of its semantic interpretation are processed and interpreted. Depending on the type of application and its degree of interactivity, appropriate actions are triggered. For example, a digital notebook might render the digital ink data to produce a facsimile of the physical ink written, while a dictionary lookup application might obtain a selected (result of a gesture) word (result of the handwriting recognition) and perform the dictionary lookup.

If connections of two successive processing stages are realized using a networking middleware, it is even possible to distribute the processing stages physically, depending on the available computing resources. This set-up is referred to as a *distributed interaction processing pipeline*.

LETRAS: ARCHITECTURE AND FRAMEWORK

Letras defines a generic architecture supporting ubiquitous pen-and-paper interaction. Based on the distributed pipeline architecture, it provides a flexible, yet light-weight approach for PPI processing. Successive processing stages with clearly defined separated interfaces channel digital ink data to applications and transform it appropriately. *Letras* employs a gray-box framework approach: application developers can construct applications using a set of predefined components which only have to be "pluged together" or developers can choose to extend and customize components if required by the targeted setting. Highly flexible deployment schemes allow for easy adaption to the specific requirements of the environment at hand. Hence, *Letras* provides an extensible platform for rapid development of ubiquitous PPI based applications, empowering application developers to base on a common infrastructure and processing model. A ready to use, platform independent implementation of *Letras* is available for download. The implementation supports the development of ubiquitous PPI based applications on Microsoft Windows, Mac OS X and Linux operating systems.

Letras is constructed on top of the ubiquitous computing middleware *MundoCore* [1] offering message based publish-subscribe and remote method call communication. *MundoCore* supports language independent communication channels between logical nodes in the Mundo overlay network. Ports for multiple programming languages exist, making *MundoCore* available for broad variety of platforms. This flexibility holds for *Letras* also. It is easily possible, for example, to develop a C# based application, which uses a Java based processing pipeline. *Letras* itself is developed in the Java programming language, although, due to the flexibility of the underlying *MundoCore* middleware, any stage in the pipeline can be replaced by an appropriate stage in any other

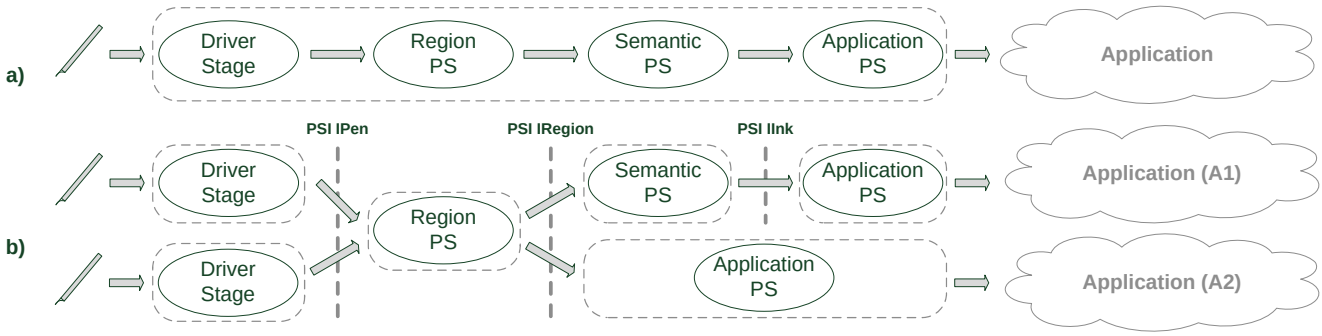

Figure 1. The Generic and Distributed PPI Processing Pipelines: Successive processing stages (PS) transform digital ink data, a) deployed in a monolithic scheme (as in traditional approaches) or b) distributed to allow for sharing of resources between applications (example setup)

programming language for which a *MundoCore* port exists[5]. Following the publish/subscribe communication paradigm, data is pushed starting from the pen as source to one or several applications as sinks.

This work draws on the insights gathered in existing infrastructures and toolkits. Essentially, the *Letras* platform provides an infrastructure to support Pen-and-Paper Interaction, similar to the *PaperToolkit* [13], or the *iPaper* plugin for *iServer* [8]. However, it extends these concepts to the ubiquitous computing domain: It serves as part of the logical driver layer of an assumed ubiquitous computing operating system, the so-called *Meta Operating System* [5], with a special focus on PPI processing. In this sense, *Letras* is less generic as for example the *OpenInterface* platform [4], which employs a generic component model to construct pipelines for multimodal interaction processing yet lacks support for ubiquitous PPI: *Letras* could be wrapped to provide one such component. However, *Letras* is designed to be more generic than for example the *PaperToolkit* [13], which supports the development of Pen-and-Paper User Interfaces, but does not handle distribution of processing stages and usage of pens in different applications: the *Letras* platform could be used as PPI processing fundament for applications designed using the *PaperToolkit*, replacing its native pen integration to leverage full support for ubiquitous PPI.

In the following we will provide a brief description of the processing stages and their respective interfaces, discuss the implementation of readily available components and highlight the most important concepts employed.

Driver Stage

The *Driver Stage* provides hardware access to digital pens. Since there are many different pen models available, *Letras* employs a flexible plugin mechanism: pen drivers can be loaded dynamically, allowing installation of a new digital pen into a running system, reducing the burden of a painful manual installation. These pen drivers could either be wrappers for drivers of the underlying operating system or access and connect the pen hardware directly. Currently *Letras* supports two native pen drivers (developed as part of *Letras*, no access to OS drivers needed): the *Nokia SU-1B* and the *Logitech IO2 (Bluetooth)*. Further native pen drivers, especially for the recent Anoto ADP-301 pen, are under development. Communication with these pen models is handled via bluetooth[6]. Each pen driver instantiates a pen service for each connected pen. This mechanism is shown in Fig. 2. The Nokia Pen Driver connects to pens P_1 and P_2, and spawns the pen services S_1 and S_2. Similarily, the Logitech Pen Driver connects to P_3 and spawns pen service S_3.

The data produced by this stage consists of a set of events indicating the current pen state (e.g. *pen set down* or *pen lifted*) and a series of samples describing the movements of the pen on a surface. The pen events published by a pen service follow the pattern described in table 1. Note, that the implementation provided with *Letras* abstracts from the differences between the supported pen models and synthesizes events that are not supported by the pen hardware in software. Pen samples contain spatial information, i.e., the x and y coordinates in the global 2D coordinate system, a time-stamp and a value sensed by a pressure sensor in the pen tip. Although a broad variety of additional sensor information might theoretically be included and thus is reflected in the W3C standard *InkML*[7] (e.g. tilt angle), the aforementioned collection reflects the data available by contemporary digital pens.

At the processing stage interface (PSI *IPen*), only the pen

[5]Including Java, C++, C# and various flavors of JavaME

[6]based on Bluecove, a JSR-82 wrapper library for various native bluetooth stacks (`http://bluecove.org/`)

[7]`http://www.w3.org/TR/InkML/`

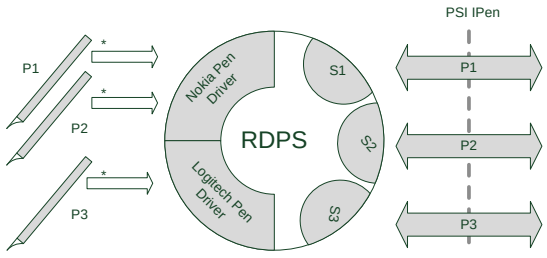

Figure 2. The Driver Stage supporting multiple pens

services and their associated channels are visible. Following this approach, multiple pens of multiple models can be hosted by the same Driver Stage. However, since the successive stages only notice the provided pen services and channels, it is also easily possible to deploy several Driver Stages in the environment as shown in Fig. 1, while other pipeline stages still see only the sum of their connected pens.

Region Processing Stage

After having provided the raw digital ink data in the Driver Stage, the samples and pen events need to be transformed into higher-level data structures and – most importantly – related to interactive regions. This is handled in the Region Processing Stage, taking input of a configurable set of pen services. In addition, it maintains a model of all currently known interactive regions as published by available PPI based applications. This publishing of interactive regions follows a 2-level approach (local means here either on the same logical node or on another node in the local Mundo overlay net and remote means knowledge only available accessing the internet):

i) locally available knowledge of interactive regions is shared between instances of the Region Processing Stage in a Peer-to-Peer approach

ii) not-locally available knowledge of interactive regions is fetched using a global naming system providing bindings of interactive regions to applications

Currently, *Letras* realizes local region retrieval only. Remote access is simulated through a dummy implementation, as the required naming system still needs to be developed. Although Guimbretière argues that DNS could be adapted for such a task [3], the highly dynamic nature and the unclear hierarchical organization of interactive region publishers suggests otherwise.

Having gathered the knowledge on interactive regions, their spatial relations are used to compute a containment hierarchy tree. The tree enables *Letras* to efficiently compute the interactive region in which a data sample is located. Because of the high spatial proximity of sample series (the samples on one stroke lie close together), the reference implementation uses a caching system to boost its performance. This is required to be able to handle the data rates of digital pens (∼100Hz). The samples then are aggregated to *traces* (samples between putting down a pen and lifting it up), which can be further aggregated. The organization of such higher-level data structures in *Letras* follows the *InkML* standard: digital ink data is organized in a tree. Samples constitute the tree leafs. Traces consisting of samples form the next higher level. Ink structures group traces or other ink structures, reflecting semantic relations among recorded data, with one ink structure (the toplevel structure) forming the root of the tree. After having detected these ink data structures at the trace level, samples along with pen events and events indicating the trace data structures are dispatched to the interested parties using a dedicated channel for each region (again employing topic based publish/subscribe approach).

Event	Description	IPen	Nokia	Logitech
ON	Pen connected (or reconnected)	Y	N	N
OFF	Pen disconnected	Y	N	N
UP	Pen tip lifted from paper	Y	Y	Y
DWN	Pen tip put on paper	Y	N	Y
ERR	Communication error	Y	N	N
OOR	Pen out of reach	Y	N	N

Table 1. Events emmited by a *Letras* Pen Service

Semantic Processing Stage

Semantic processing in *Letras* follows the service oriented, highly configurable component model employed in *MundoCore*. Combining services as needed, applications can construct individual low-level pipelines according to their specific requirements. It is possible to discover available services in the environment, or to instantiate required services in case such a service is missing (or exclusive access to a service's functionality is required). Instantiated services can be configured to restrict their processing either to a single application, or allow sharing of their processing capabilities. Available services at this stage include a generic and highly configurable segmentation service used to split digital ink structures into semantically related units (based on clustering algorithms), several gesture recognition services (a wrapper for the iGesture toolkit [7] and an implementation of the 1$ gesture recognizer [11]), as well as handwriting recognition services (a wrapper around the Microsoft handwriting recognition service).

Application Stage

Digital ink data and the results of semantic processing are interpreted by a PPI based application. This is not part of the toolkit. However, some tasks are common to many applications, e.g., the storage of digital ink in a document database, or the rendering of digital ink onto a GUI. Therefore *Letras* provides some convenience services at this stage. It includes a persistence service capable of storing digital ink data and a generic digital ink rendering service.

CONCLUSION AND OUTLOOK

Letras provides system support for integration of PPI into applications by accessing digital pen hardware and managing interactive regions. In addition to the system support already provided by existing approaches, it addresses the aspects of user and document mobility.

User Mobility

Specifically, *Letras* supports the pen's interactive mode in mobile settings. This, combined with the concept of a distributed interaction processing pipeline allows to support interactivity to the high degree needed by ubiquitous PPI based applications. Ubiquitous PPI involves using paper artifacts in varying, heterogenous environments, as the medium paper affords highly mobile use. The distributed PPI processing

pipeline supports this by defining clear processing stage interfaces and flexible "wirings" between the processing stages. It allows to discover available processing stages, and to adapt the pipeline layout dynamically, using the service discovery mechanism of *MundoCore*. Connections are handled completely network transparent, that is: it is not important whether a processing stage is hosted locally, or is distributed in the Mundo overlay network. Additionally, different interaction resources (i.e., different pen models) can be supported and managed in the same pipeline.

Document Mobility

Ubiquitous PPI broadens the scope of PPI based applications by supporting document mobility. First, interaction of a digital pen with several applications at once needs to be supported. The distributed processing pipeline solves this: As a separate layer between the used communication middleware and the applications, it allows to share processing and interaction resources between applications. Second, support for interaction with encountered documents demands for distributed and highly scalable mapping of interactive regions to applications. At the same time, the dispatching of digital ink to interested parties must be handled efficiently and the solution must scale to a global dimension. In *Letras* this is solved using a 2-level peer-to-peer based approach combined with a local cache. Employed in the pipelines region processing stage, this 2-level Peer-to-Peer architecture, allows to distribute the knowledge on regions efficiently in the local network by sharing the local cache between instances. As it is also backed by a global overlay net for region publishing, it spans the dynamic definition of interactive regions needed to support the integration of interactive surfaces for both paper-like and other interactive surfaces, as well as the publishing of interactive paper documents.

In summary, *Letras* provides support for ubiquitous pen-and-paper interaction through the introduced distributed processing pipline, enabling developers of PPI based applications to provide interactivity to physical paper, without compromising its inherent mobility.

ACKNOWLEDGMENTS

Thanks to Niklas Lochschmidt, Jannik Jochem and Erwin Aitenbichler for their efforts in implementing parts of *Letras*. Part of this research was conducted within the ADiWa project funded by the German Federal Ministry of Education and Research (BMBF) under grant number 01IA08006.

REFERENCES

1. E. Aitenbichler, J. Kangasharju, and M. Mühlhäuser. Mundocore: A light-weight infrastructure for pervasive computing. *Pervasive and Mobile Computing*, 3(4):332 – 361, 2007. Middleware for Pervasive Computing.

2. P. Brandl, C. Richter, and M. Haller. Nicebook - supporting natural note taking. In *CHI 10: Proceedings of the SIGCHI conference on Human factors in computing systems*, 2010. [in print].

3. F. Guimbretière. Paper augmented digital documents. In *UIST '03: Proceedings of the 16th annual ACM symposium on User interface software and technology*, pages 51–60, New York, NY, USA, 2003. ACM.

4. J. L. Lawson, A. Al-Akkad, J. Vanderdonckt, and B. Macq. An open source workbench for prototyping multimodal interactions based on off-the-shelf heterogeneous components. In *EICS '09: Proceedings of the 1st ACM SIGCHI symposium on Engineering interactive computing systems*, pages 245–254, New York, NY, USA, 2009. ACM.

5. M. Roman, C. Hess, R. Cerqueira, A. Ranganathan, R. H. Campbell, and K. Nahrstedt. A middleware infrastructure for active spaces. *Pervasive Computing, IEEE*, 1(4):74–83, 2002.

6. A. J. Sellen and R. H. R. Harper. *The Myth of the Paperless Office*. MIT Press, Cambridge, MA, USA, 2003.

7. B. Signer, U. Kurmann, and M. Norrie. igesture: A general gesture recognition framework. In *ICDAR '07: Proceedings of the Ninth International Conference on Document Analysis and Recognition*, pages 954–958, Washington, DC, USA, 2007. IEEE Computer Society.

8. B. Signer and M. C. Norrie. Paperpoint: a paper-based presentation and interactive paper prototyping tool. In *TEI '07: Proceedings of the 1st international conference on Tangible and embedded interaction*, pages 57–64, New York, NY, USA, 2007. ACM.

9. B. Signer, M. C. Norrie, M. Grossniklaus, R. Belotti, C. Decurtins, and N. Weibel. Paper-based mobile access to databases. In *SIGMOD '06: Proceedings of the 2006 ACM SIGMOD international conference on Management of data*, pages 763–765, New York, NY, USA, 2006. ACM.

10. J. Steimle. Designing pen-and-paper user interfaces for interaction with documents. In *TEI '09: Proceedings of the 3rd International Conference on Tangible and Embedded Interaction*, pages 197–204, New York, NY, USA, 2009. ACM.

11. J. O. Wobbrock, A. D. Wilson, and Y. Li. Gestures without libraries, toolkits or training: a $1 recognizer for user interface prototypes. In *UIST '07: Proceedings of the 20th annual ACM symposium on User interface software and technology*, pages 159–168, New York, NY, USA, 2007. ACM.

12. R. Yeh, C. Liao, S. Klemmer, F. Guimbretière, B. Lee, B. Kakaradov, J. Stamberger, and A. Paepcke. Butterflynet: a mobile capture and access system for field biology research. In *Proc. CHI '06*, pages 571–580, New York, NY, USA, 2006. ACM.

13. R. B. Yeh, A. Paepcke, and S. R. Klemmer. Iterative design and evaluation of an event architecture for pen-and-paper interfaces. In *Proc. UIST '08*, pages 111–120, New York, NY, USA, 2008. ACM.