A category based concept for rapid development of ink-aware systems for computer-assisted education

Georg Turban
Department of Computer Science
Technische Universität Darmstadt
Germany
turban@informatik.tu-darmstadt.de

Max Mühlhäuser
Department of Computer Science
Technische Universität Darmstadt
Germany
max@informatik.tu-darmstadt.de

ABSTRACT

Presentation centric lectures that draw on integration of multimedia content and new input devices gain increasing importance in the context of eLearning. The paper emphasizes the addition of digital ink, which is of increasing importance for public (schools, universities) and corporate (trainings, Webinars) teaching. We present a concept for flexibly developing custom ink-aware systems that support augmented presentations for a broad range of use cases and cover not only the core event but also pre- and post-event phases, addressing the needs of both lecturers and students. Our concept for ink-aware multimedia architectures is based on three main system components, which are in turn assisted by small signal processing units. The concept fosters development of clearly and consistently designed ink-aware applications. The paper emphasizes examples where digital ink is visualized atop image-based slides, but potential applications comprise a much broader range of underpinning application. To illustrate our concept and implementation, we present sample customizations i.e. realizations of all relevant component categories. We discuss evaluation results obtained from both interviews with lecturers as well as our experience with using and supporting customizations made by ourselves.

Keywords

Digital ink, eLearning, eTeaching, media, presentation, processing, components.

1. INTRODUCTION

In the field of electronically assisted presentations (at universities, in schools or for corporate training), the importance of ink-aware systems is drastically increasing. Such systems can provide presentations with a wide range of interesting functionalities and assist lecturers and students.

The presented work identifies and minds several aspects of typical presentations and derives a category-based concept for rapid development of ink-aware systems. Several key concepts of the design assure universality and flexibility of customizations and will be illustrated with example realizations.

The remainder of this paper is organized as follows: The following section describes our concept and proposals, followed by section 3 that presents several examples to illustrate our contributions. Section 4 deals with the evaluation of the whole concept, the several sub-systems and discusses our experiences with users and usage in presentations. In section 5 we summarize the content of our paper and conclude with section 6, presenting possible starting points for future research.

2. CONCEPT

A main purpose of a computer based presentation system is to augment ordinary presentation so that students and lectures can draw more benefit. We emphasize rapid development of ink-aware systems, with a concept that focuses especially on universality, simplicity and high reusability of well segmented sub-systems.

We will use the term *presentation event* in the remainder to denote the atomic components of the broad range of possible public and corporate eTeaching / eLearning scenarios considered, typically associated with a single calender entry (e.g., a 40 min. math class at a particular date). For ease of understanding, the introduction of the concept is centered on the processing and exchange of material (content and media) during presentations. In this context, several modalities and media can be distinguished, by which lecturers can mediate content to students (learners):

1. speech
2. gestures that can be global (body movements) or local (e.g. hand or even finger gestures)
3. facial expressions
4. *prepared* media, like projected videos (dynamic), slides (often static) or played audio
5. *additional* media, like handwritten content developed using traditional chalkboards or overhead projectors

This non-exhaustive list shows that different media and modalities can be, and usually are delivered to the attendance of a lecture. Any combination of these can be augmented with digital ink in many ways and with substantial

Proceedings of the Seventh IEEE International Symposium on Multimedia (ISM'05)
0-7695-2489-3/05 $20.00 © 2005 IEEE

additional benefit. While individual software tools like PowerpPoint™ and Acrobat™ were augmented by digital ink in the past, such augmentations remain isolated and incompatible. A universal concept and system support for handling digital ink is a challenge and research issue.

A first step towards such a universal concept is a general model for how different tools contribute to the handling of media before, during, and after presentation events.

We use the term *materials* to distinguish parts of a presentation originating from different sources and potentially from different authoring tools – potentially representing different media types, too. Tools or components involved in presentation events deal with topics like adding several different materials to the current presentation stream or processing of delivered materials. For example they can

- control the presentation of materials,
- create and add additional information like annotations and/or
- record the presented content for later playback.

We suggest a schema for the development of ink-aware presentation systems that distinguishes three substantial subsystem types called *categories*:

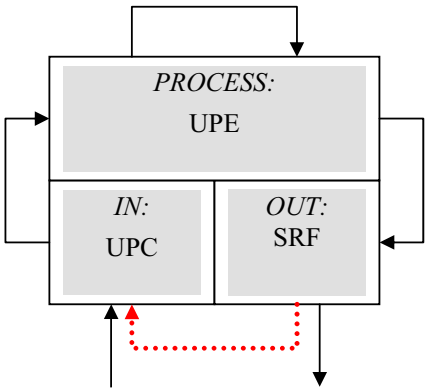

Figure 1: Schema of category-based concept.

The categories are motivated by following identified roles of materials and corresponding requirements before, during, and after lectures:

- universal *integration* of materials
- universal *processing* of materials
- universal *handling* of presented and recorded materials

Figure 1 above denotes these categories. It should be noted that circular connections can be formed, denoting possible real-time (re-)integration of output-streams into category "IN".

2.1 *IN:* Universal Presentation Controller

Category *IN* denotes the part of the system which supports integration of different materials/media for presentation in a unified way. More specifically, this category specifies what we call *universal presentation controllers (UPC).*

The proposed internal presentation model consists of a description of the current input materials, not limited to simple slide sets, such that more complex formats can be supported, including several steps on a single slide (e.g. animations), multi-layer and multiple-over-time annotations for each slide, etc.. All input materials are augmented by a meta data provided by our system, including information like category (e.g. continuous, slide-based, …) duration and amount of slides, an identifier representing the format (e.g., PDF , Power-Point, home-brew), etc..

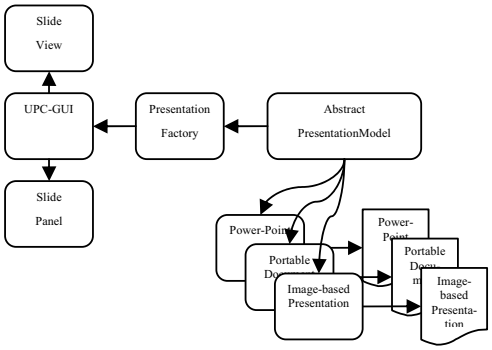

Figure 2: Representing and controlling different input materials.

Figure 2 illustrates the customization of the unique presentation controller; it draws from the universal UPC concept and provides a presentation model for several input materials that can be achieved in real-time.

In contrast to existing approaches, a fundamental idea is to avoid any tool specific conversion modules or specialized internal formats. Since real-time integration / conversion approach as applied is functionally superior to offline conversion and use of internal formats, the approach covers such functionality implicitly.

In summary, the UPC processing category is able to handle different input media in a unique way. For more information about bridging media breaks in presentations, cf. [1].

2.2 PROCESS: Universal Processing Engine

Based on the standardized output stream delivered by a so-called *universal processing engine UPC*, the processing engine (category "PROCESS" in Fig. 1) can provide maximum functionality in a unified way, with comparatively low effort. In a simplified view to start with, UPE can be considered to treat any materials, at any given point in time, as a single image. Under this assumption, a represen-

tation of an ink-aware presentation instance can be depicted as follows:

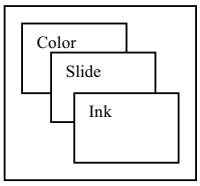

Figure 3: Ink-aware model for presentation slides.

The hindmost layer (cf. "color") represents the background color used and enables the use of transparent slides. By using "fully transparent" slides in the middle layer or by omitting this layer, a *blackboard*-mode can be provided. (The hindmost layer is a prerequisite for the presentation model and can not be omitted). To provide the user with a high degree of freedom, it must be possible to insert, modify and develop either the whole presentation model or parts/layers of it. By switching to different presentation models, it is possible to (re-)present a presentation model that has been presented and modified earlier. It is also possible to easily backup a specific presentation model that e.g. contains an overview of the content of the whole slide-set. Instead of showing the *overview*-slide between some sections of the main presentation, the backup model can be displayed at a special area (where it can be manually modified i.e. annotated). This way, presenters can e.g., highlight focused sections in the TOC and thereby avoid the usual duplication, modification and placement of overview-slides in the slide-set at authoring time.

Since our system adds these layers plus additional meta data (most are time based for later synchronization) and content (e.g. captured ink), the ink-aware model for presentation slides is a superset of the prior input-stream delivered by category "IN"; this simplifies unique treatment in the next category denoted "OUT".

The component for developing the "ink layer" ad hoc, during presentation, is an integral yet modular (replaceable, customizable) component of UPE. It contributes 'yet another material' to the presentation. UPE contributes additional information such as timing information for later synchronization. Mukhopadhyay and Smith identified the problem of synchronization for systems that collect and combine data from multiple independent capture devices [5]. In the context of their classification of "types of multimedia synchronization" (for pair wise synchronization of collected data), those timed presentation models are synchronization points in a so called *timed-timed synchronization* problem. An example for timed-timed synchronization is the synchronization between a video recorded during the presentation and the presented and captured slides.

As described above, our concept of ink-aware systems has the advantage of a more appropriate representation of presentation events. We must account for with advantage with more fine-grained processing. Individual timing information is added to each stroke in the presentation model, enabling a much more dynamic representation of recorded slides, instead of a snapshot that only represents the last and static state of a continuously modified slide. In addition, we leverage off the time-points of the entire presentation model (see above). This approach yields timed-timed synchronization between the appearance of slides and the corresponding ink.

In summary, UPE components add content and metadata, thereby augmenting the input stream as received by the "IN" component (or by UPE itself in a prior processing step). Therefore, the input stream delivered to the next category ("OUT") is of the same type as the input stream to UPE.

2.3 *OUT:* Storage / Replay Factory (SRF)

Category "OUT" concerns the following four key design aspects:

1. editable storage of ink-aware streams as described above, including all content and meta data
2. real-time displays facilities, especially for any ink-operation
3. real-time reintegration into category "IN"
4. *dynamic* replay facility for stored content, especially ink

Item 1 above turned out to be a major challenge in the design of the internal data format. Our concept contrasts the usual approach, where final-form standards such as MPEG are used as primary storage formats. (In our approach, such final-form documents can be produced from our editable format using a dedicated SRF component.)

Another contribution of our approach is the introduction of real-time display at a very late stage in the processing chain! This design has several advantages; e.g., a lecturer's control & status panels can be hidden from the audience, material can be processed without being publicly visible, etc.

Real-time reintegration functionality (cf. item 3 above) is provided for use at different points in time. This provides for a flexible customization approach in SRF, covering learners (e.g., material can be broadcast and reintegrated during presentations for student annotation) as well as teachers (e.g., revisiting annotated material during Q&A sessions).

In summary, great flexibility is drawn from the option to loop back presentation streams at any point in time and space (during initial presentation, during Q&A, or off class, at any computer involved) and into any of the three "categories". Therefore, customizations of the system may be

applied in a broad variety of use cases ("teacher station", "player", "rehearsal station", etc.).

The ink-aware stream emphasized in item 1 above covers persistent storage for so called *rich images* (representing the slide models during processing) and can be used in a wide range of applications. Such rich images represent a specific moment of the given presentation using the additional time-information contained. Adding information like time points to the images before storage will result in *rich images*. Such information can be used, e.g., to provide several alternatives for the combination of "slides" (in terms of our model) and ink. To cite one example: an annotation stroke can be placed aside or atop a shrunk "slide", with a link to the starting point of the annotation as originally provided.

There are several different strategies for storing and replaying ink. We believe that a high quality of recorded ink depends on extremely high precision of underlying data as mentioned in section 2.2. Therefore we recommend and focus on approaches which consider the high quality of pen-enabled input devices. Such pen-enabled input devices should be able to deliver e.g. pen-coordinates that are much more precise than then the resolution of their integrated displays, e.g., Wacom's Cintiq devices [10].

Our approach adds additional information to the ink-aware model for presentation of slides (see figure 3) and of the ink itself:

When publishing a slide, we obtain the current time $t_{begin(slide)}$ and store it with the corresponding slide model. When the slide is unpublished, we also store the current slide and the time $t_{end(slide)}$. Since both moments can theoretically be identical, the relation between both points in time can be expressed by following equation:

$$t_{begin(slide)} \leq t_{end(slide)}.$$

Equation 1: Period of a presentation model

Each stroke is "timed", but instead of just adding one single time information per stroke, we store the beginning and the ending of the stroke drawing as a tuple. The two values can be identical if a sample point (in the smallest resolution available) is drawn instead of a larger annotation or synthetic combination of primitives. Similar to the relation of the moments representing the publishing / disappearing of slides, a stroke can be called *timed* since the following equation is governed:

$$t_{begin(stroke)} \leq t_{end(stroke)}.$$

Equation 2: Period of a stroke

Without any optimization, we will collect all strokes during the presentation of a specific slide in a basic fashion and always store them into the corresponding presentation model. Therefore, ink can only be created and added during the period of a slide that is currently published, representing the corresponding slide model that is currently created. In summary, a stroke fulfills the following equation derived from the two equations presented just before:

$$t_{begin(slide)} \leq t_{begin(stroke)} \leq t_{end(stroke)} \leq t_{end(slide)}$$

Equation 3: Period of a slide model including rich ink

By augmenting the model for presentation slides with all the time information mentioned, we can simply reconstruct a realistic queried moment of the original presentation including the full state of our ink data. By keeping track of both moments (beginning and ending of the creation of a stroke), we can introduce a time point labeled $t_{keyframe}$ that represents the queried moment of the original presentation. For a realistic and dynamic replay of ink-aware recordings, we are now able to determine the following three states for each stroke that has been drawn in a slide model (for clarity, we will omit terms of the equation that express the reference to a specific slide in following paragraphs):

(i) $t_{keyframe} \leq t_{begin} \leq t_{end} \Leftrightarrow$ invisible

(ii) $t_{begin} \leq t_{keyframe} \leq t_{end} \Leftrightarrow$ partially visible

(iii) $t_{begin} \leq t_{end} \leq t_{keyframe} \Leftrightarrow$ visible

Equation 4: Basic lifecycle of a stroke

In relation to the corresponding model for a presentation slide, all strokes fulfill the following equation during every possible point of time during the replay of a recorded slide model:

$$t_{begin(slide)} \leq (i),(ii),(iii) \leq t_{end(slide)}$$

Equation 5: Handling different stages of ink

In theory, value $t_{end(slide)}$ must be initialized to infinity at the beginning of the recording of a slide; all equations hold for the whole process of the creation of the slide model.

The *lifecycle* of a stroke starts in the state invisible. In most cases, transitions to different partially visible states follow (depending on the frequency of queries); the final state is "fully visibile".

The intermediate states are handled by a specific routine that mainly returns a subset of the partially visible stroke related to the current key frame. Given this routine, the first and last state are easy to handle.

Note that all the above basic (and rather obvious) equations are given just to introduce our internal model. As we move to annotation editing and even to rehearsal including distributed annotations, the time model becomes more elaborate. This is where the formal foundation pays off.

As a still rather simple example, we will show how removal and movement of strokes are covered. We introduce another point in time called $t_{removed}$ and augment equation (iii) in order to cover the deletion of strokes:

(iii') $t_{begin} \le t_{end} \le t_{keyframe} \le t_{removed}$ ⇔ still visible

(iii'') $t_{begin} \le t_{end} \le t_{removed} \le t_{keyframe}$ ⇔ again invisible (or no more visible)

Equation 6: Extended handling of strokes

The following figure illustrated a typical lifecycle of a stroke that has been removed after period $t_{removed} - t_{end}$ since it was created and thus visible:

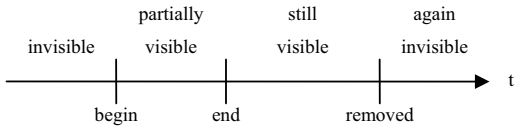

Figure 4: Typical lifecycle of a stroke

A single frame is now able to represent nearly the original and full state of the scene [4]. The World Wide Web consortium works on the "Ink Markup Language" (InkML) [7] that meets our requirements and is able to store time-information even for each sample point of a stroke. The working group summarizes the requirements for InkML in "Requirements for the Ink Markup Language" [6], where the a discussion has already started about importance of time-information for ink capture and processing.

The markup language is able to represent ink data captured by pen-enabled multimodal systems and enables different systems to exchange such information. The *time channel* of InkML allows the detailed recording of the timing information for each sample point within a single stroke. Earlier work on standards that are able to exchange ink and discover the importance of timed ink are the *Unipen* project [8] and the *Jot* specification format for storage and interchange of ink [9].

2.4 Comprehensive use of media-modules

As opposed to the three categories described before, the concept of *media-modules* is mainly motivated by technical reasons, such as high reusability. It enables an overall intelligent architecture that can be realized rather easily yet remains concise and clear.

Media-modules encapsulate functionality and represent small processing units containing (in principle) a single input and output channel similar to "data sinks" and "processors" in the Java Media Framework [2]. They can be combined to represent more complex processing units and represent entire controls similar to logical circuits.

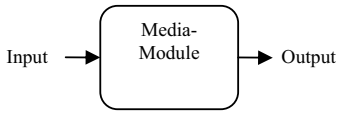

Figure 5: Schematic drawing of a media-module.

Among others, the following section covers the realization of different categories powered by media-modules in order to illustrate how everything fits together.

3. REALIZATION

The realization of the entire systems was divided into three core categories as described plus support for media-modules; this approach provides for parallel development of implementations for each category as well as realization of several media-modules. Integration of all necessary parts is rather simple due to a minimalist approach to interfaces and media flows. Media-modules support loose coupling of sub-systems or even of whole categories and enable fast set-ups for testing purposes and rapid development.

The following figure displays user-interfaces of a concrete system realization, representing all three categories plus the underlying use of media-modules.

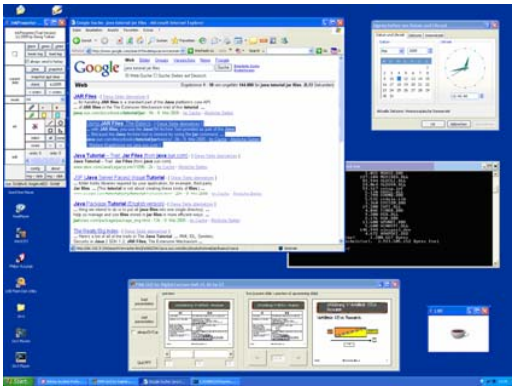

Figure 6: Example realization of all three categories.

It should be noted that in the lower right corner, a media-module was placed on the lecturer's desktop (showing a cup in a frame). This window would normally be configured for presentation on an external display device and in full-screen mode.

3.1 Relization of UPC

A realization of the *IN* part i.e. a UPC has been heavily used for more than six months, mainly in combination with other processing elements. The realization followed the basic concept of easy-to-use control for different materials (via input modules) in a unique way; thereby the realization followed the design principles and presentation model described in section 2.1.

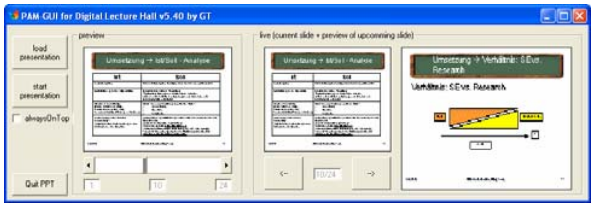

Figure 7: Sample realization of UPC.

The above figure shows the version of UPC that is currently in use in class; it eliminates limitations of other presentation software based on a multiple-display approach, preview functionality, and the possibility to navigate within entire slides-sets; the widget-like user interface style ensures that other portions of the desktop remain reachable, such that the processing component itself (see below) and other applications are not permanently occluded, as is the case, e.g., with the presentation controller in Powerpoint™ [3].

A simple example of such an external application used in parallel to other applications is a large clock, which was requested by a lecturer who wanted it to remain visible on the desktop (cf. figure 6).

3.2 Realization of UPE

As mentioned above, UPE is mainly used during presentation. The realization described here makes extensive use of media-modules and models as introduced in former sections. A snapshot of its user interface gives a first idea of the features offered in a realization which is currently in heavy use:

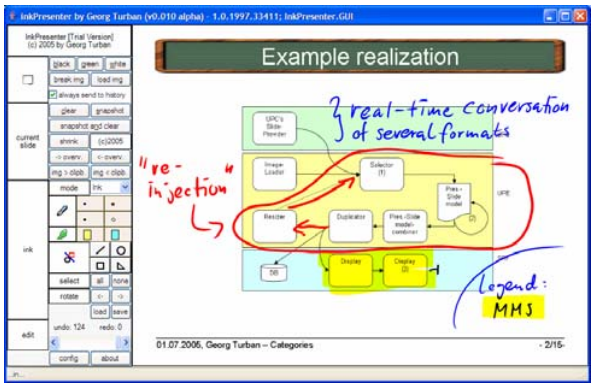

Figure 8: User interface of UPE

An example for a typical scenario media is visualized in the following simplified schematic drawing. It contains several media-modules to be coordinated and controlled:

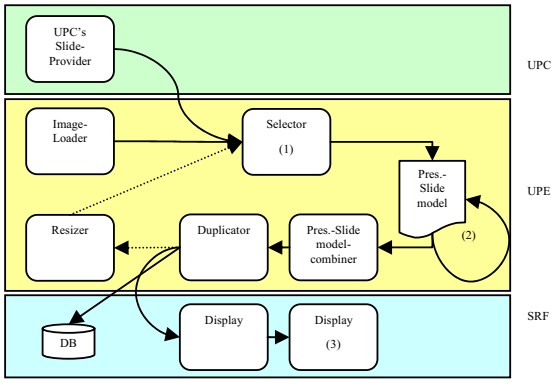

Figure 9: Example interaction of all three categories.

As a first part of a processing task, a user (presenter) chooses between slides delivered by UPC and images loaded into UPE directly (1). Typical processing operations performed (2) include manual inking and adding drawing primitives to the slide model. In the example given, the subsequent processing step is covered by a media-module that adds meta-information and finalizes a presentation slide model after flattening all layers (cf. section 2.2).

In the example realization depicted, dotted arrows feed back the output of modules *duplicator* and *selector,* thereby traversing module *resizer*. This feedback loop can be used, e.g., by a feature called shrink (cf. figure 8): it reduces the size of the content currently presented, such that extra free space is available around the slide for additional annotations.

The other routing paths outbound from module *duplicator* will be discussed in the next section.

3.3 Realization of SRF

Apart from the internal feedback loops described, output from category *PROCESS* i.e. from UPE may be linked to SRF for display and storage of content delivered. In the lower part of figure 9, the expressive power and simplicity of the concept is illustrated (3): by simply connecting media-module *display* with additional modules of the same type, multiple-monitor support and support for multiple additional virtual displays i.e. screen-sections can be realized. Such multiple displays can be used, e.g., for realizing a virtual multiboard: when the content of the actual slide changes, former content is still be presented on one or more monitors; other displays may even present "history slides" which date further back in time.

Due to the data-driven implementation of the display modules, it is very easy to determine, e.g., that the content of a display is shifted to another display whenever the first display in a chain receives new input data. Thereby, each display triggers its successor.

The basic level of storage support, based on rich images of the presented slides, is depicted in the screenshot below by way of a sample output.

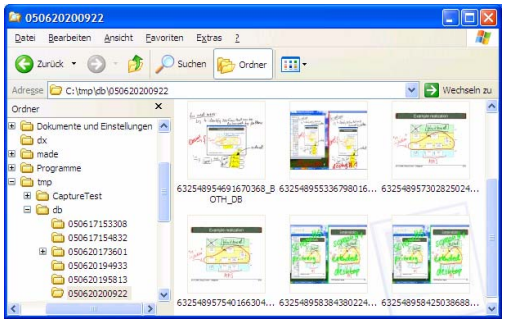

Figure 10: Exemplary output of image-based storage.

Similar to *display* modules, *storage* modules basically consume images as delivered by *duplicator* modules. Using a single instance of a storage module and relying on the plain file system instead of a database, the entire storage functionality can be implemented in a few lines of codes. After an initial setup (creating a folder for each session to be recorded), each flattened slide model is stored in a respective folder.

4. EVALUATION

We developed several category realizations for test purposes and an entire presentation system for daily use based on the modular design introduced in this paper. The system succeeded an older, traditionally developed system. A first proof of concept was the possibility to development realizations of all categories in parallel. UPC could be launched ahead of time while the other two category realizations were still internally tested and improved. The comprehensive usage of media-modules proved to be very convenient: on one hand, several modules were easily incorporated into different categories; on the other hand, modules proved to be very handy for building up test and demonstration scenarios.

The remaining paragraphs will emphasize the evaluation of UPC. In a prior version, it was only possible to navigate to the previous or next slide for preview. Most users reported that they would appreciate a possibility to navigate arbitrarily in their entire slide set instead of having to carry out a (potentially large) number of single-step navigations in order to skip a wide range of slides. In addition, a couple of users requested that navigation in the slide set preview would not automatically change the slide actually presented.

Both requirements were combined and supported by replacing the leftmost pane of the prior version by a new navigation pane. In summary, faster and more privacy-protecting navigation support was realized.

As expected, the leftmost pane could be disposed of most easily; during sequential presentation, it had shown the past slide, which is by far more easily displosable for lecturers than for students (cf. multiple-display functionality above). Removing the preview and current-view feature was not a choice since both – especially the preview feature - were acknowledged by presenters as a means for smoother and easier transition between slides.

Care was taken to simplify the additional interaction required in case of non-sequential (left-pane based) navigation (as mentioned above, it became necessary since navigation did not automatically change the slide publicly presented any more): a slide selected via left-pane navigation was now publicly presented after a simple *single-click* anywhere on the preview. Synchronization of the two other panes (current-view, preview) was again straightforward based on the modular design discussed in this paper.

As mentioned, a full-fledged version of a presentation system following our design (and platform) was widely used by lectures in a total of five scheduled weekly lectures two block courses of several days duration each. In addition, a considerable number of single presentations were successfully supported by the controller.

The application turned out to be very robust and – after a short introductory instruction of less than 15 minutes – easy to use. The installation is automated for the most part (minor OS modifications are required that require a few manual steps). As can be expected, the most popular materials types are Microsoft PowerPoint™ 2002 and 2003.

5. CONCLUSION

In this paper, we presented two interwoven major contributions, both motivated by theoretical and practical aspects at the same time: The concept of three basic categories and the concept of media-modules. Combined into a systematic approach, these two approaches enable rapid development of ink-aware presentation systems for all kinds of classrooms.

The recommended categories were presented and discussed in corresponding sections, covering theoretical aspects first and realizations afterwards. The comprehensive concept of media-modules and of ink-aware processing units provides for lose coupling of the categories and enables both parallel development and isolated use, reaching beyond pure ink-aware presentation scenarios.

We introduced several realizations of the different categories, moving from preliminary stages to extensive testing and heavy use. The realization of category UPC as presented above was successfully used over a period of several months and became a major subject of evaluation as presented. The focus of present work is on the improvement of the other two categories, plus UPC improvements as described below.

6. OUTLOOK

We plan to implement a number of additional features in subsequent UPC realizations, e.g., support for additional presentation formats, and enhancement of the existing internal format. This internal format enhancement should lead to dynamic slide models in category *IN,* too, and support formats which allow modification of presentations during presentation.

In a first step, such dynamic slide models should support unchanged dynamic content as obtained from streaming devices such as Web cams.

In a next step, a fully dynamic presentation model should cover, e.g., videos delivered either from preexisting files or real-time captures of video devices, and other computed content to be re-computed each time presented.

In addition, more research will be devoted to the coupling of UPC, UPE, and SRF realizations.

7. REFERENCES

[1] Turban, Georg; Rößling, Guido; Trompler, Christoph: Bridging Media Breaks, in: Proc. ITiCSE 2005, ACM Press, Monte de Caparica, Portugal, 2005.

[2] Sun: Java Media Framework, http://java.sun.com/products/java-media/jmf/index.jsp, seen on June 20, 2005.

[3] Turban, Georg: Partly usage of Presentation Systems in Lectures, Technical Report TU Darmstadt, 2005 (in print).

[4] Wolber, David: A Multiple Timeline Editor for Developing Multi-Threaded Animated Interfaces, ACM Symposium on User Interface Software and Technology, p. 117-118, 1998.

[5] Mukhopadhyay, Sugata; Smith, Brian: Passive capture and structuring of lectures, ACM Multimedia, p. 477-487, 1999.

[6] W3C: Requirements for the Ink Markup Language, http://www.w3.org/TR/inkreqs/, seen on July 11, 2005.

[7] W3C: Ink Markup Language, http://www.w3.org/TR/InkML/, seen on July 11, 2005.

[8] Guyon, Isabelle; Schomaker, Lambert et al: "UNIPEN Project of Data Exchange and Recognizer Benchmarks," unpublished standards document, 1993, http://hwr.nici.kun.nl/unipen/, seen on July 14, 2005.

[9] JOT: A Specification for an Ink Storage and Interchange Format," unpublished standards document, 1993, http://hwr.nick.kun.nl/unipen/jot.html, seen on July 14, 2005.

[10] Wacom: Interactive pen displays, http://www.wacom.com/lcdtablets/index.cfm, seen on July 14, 2005.