# Media-oriented operators for authoring interactive multimedia documents generated from capture sessions

Didier A. Vega-Oliveros, Diogo S. Martins, Maria da Graça C. Pimentel
Universidade de São Paulo
São Carlos, SP – Brazil
{davo,diogosm,mgp}@icmc.usp.br

## ABSTRACT

Synchronous communication tools allow remote users to collaborate by exchanging text, audio, images or video messages in synchronous sessions. When generating records from captured meetings, the alternative usually adopted is to generate a linear video with the contents of the exchanged media. Such approach limits the review of the meeting for watching a video using the traditional timeline-based video controls. In scenarios in which automated tools generate interactive multimedia documents as a result of capturing a meeting, the literature reports the use of ink-based and audio-based operators that allow the identification of points of interaction in the resulting document. In this paper we extend that set of media-based operators in order to take into account user interactions with boards and videos, and to extend the audio and ink-based operators with action-based alternatives.

## Categories and Subject Descriptors

H.5.1 [**Information Interfaces and Presentations**]: Multimedia Information Systems - Audio, Video.

## General Terms

Documentation, Human Factors.

## Keywords

Interactive Video, Document Engineering, Authoring.

## 1. INTRODUCTION

In some scenarios, it is paramount that collaborative synchronous sessions be recorded for later review. Some of the more important reasons for reviewing a recorded session include keeping accurate records, revisiting portions of a session which were misunderstood, obtaining proofs and recalling certain ideas [8].

*Capture and access* (C&A) has been defined as "the task of preserving a record of some live experience that is then reviewed at some point in the future" [1]. Research regarding *capture and access systems* [19, 24] has been concerned with methods for reviewing captured sessions, generally tackling the issue of developing indexes so that users can jump to relevant points of interest within the session. Minneman et al. [15] and Geyer et al. [11] categorize these indexes into four broad classes: intentional annotations, performed explicitly by participants while the meeting is happening; side-effect indexes, produced by capturing user-media or user-equipment interactions; derived indexes, automatically obtained by content-based analysis; and post-hoc indexes, consisting of user-media interactions performed during review of the meeting recordings.

From these categorizations we notice that central dimensions are the types of indexed interactions (user-media, user-equipment or user-user) and the phases of the media lifecycle [13, 14] at which these interactions take place. Abowd et al. [2] and later Pimentel et al. [16] propose a *5-phase model* of the lifecycle of multimedia in C&A systems: each phase of the lifecycle presents opportunities to build indexes based on different types of interactions. In the *(1st) pre-production phase*, offline intentional annotations can be applied to perform content segmentation, for instance. During the *(2nd) recording phase*, online annotations and side-effect indexes are commonly built [6]. Offline indexing can occur in the *(3rd) post-production phase*, such as the generation of derived indexes [4]. Review-time explicit indexes obtained from user-media interaction can be built during the *(4th) access phase*, while the user reviews captured media elements (by means of annotations or discrimination of moments [18], for instance). Such user-media interactions can be used to enrich and generate new versions of the original media elements [7] in the *(5th) extension phase*.

Non-linear access to the content using indexes is mediated by specialized tools commonly regarded as meeting browsers or meeting players. Current approaches for developing meeting browsers have their drawbacks. These browsers are commonly built using *ad-hoc* frameworks that entail tight coupling among the capture environment and the browser tool [5], thus reducing reuse of the same tool to access sessions gathered from different environments. Moreover, these browsers are usually specialized toward one or two types of indexes [20], impairing the richness of random-access operators that users can apply while reviewing the session.

In earlier research we reported the opportunity of exploiting operators which, by modeling the user interaction associated with pen-based devices such as electronic whiteboards, allow the review of the ink-based session as a doc-


*SAC'11* March 21-25, 2011, TaiChung, Taiwan.

ument [7]. The approach was extended to allow browsers to be built through the automatic generation of an interactive multimedia document (iMMD) from the captured session involving multiple media [21]. The generated iMMD is enriched with several types of timestamped media-based operators — e.g. ink-based interactions[1] and audio events[2] — called *Interactors* as a generalization of "operators based on the interaction of a user with some media": the result is an interactive multimedia document with points of interest presented on a timeline, for instance. By means of document transformations, an XML-based interchange document is converted into an iMMD that synchronizes the streams of captured media and provides timeline-based access operations for browsing [21]. The approach was demonstrated through the generation of an interactive multimedia document in a declarative language (Nested Context Language (NCL) [17]) with procedural objects in Lua [9].

Inspired by the effectiveness of the *Interactors* [21] model, in this paper we present new operators that take into account user interactions with boards, text messages and videos, and extend the audio and ink-based operators with action-based alternatives. The remaining of this paper is organized as follows: Section 2 reviews the definitions of Inkteractors and AudioInteractors, and reports new extensions to these operators. Section 3 details the new operators proposed in this paper (TextInteractors, BoardInteractors and VideoInteractors). A proof-of-concept prototype is shown in Section 4. Evaluation results are reported in Section 5. Related work is discussed in Section 6. Section 7 summarizes this paper's contributions and points out future research efforts.

# 2. EXTENDED INTERACTORS

In this section we review and extend the *Interactors* model of operators to generate interactive multimedia documents (iMMD) from captured media [21]. An *Interactor* has been defined as an operator that is applicable to a specific type of medium [21]. Examples are *Ink*-based Interactors (e.g. ink-based annotations produced with pen-based devices) [7] and *Audio*-based Interactors (e.g., related to the detection of moments of silence) [22].

Given that a captured session describes $j$ media elements and $x$ logged Interactors per medium, we define a set of interactor events as a mapping $TL(a, b)$ where $a \leq j$ is a captured media element and $b$ is a logged Interactor. By definition, each Interactor can map an unlimited number of interactor events. Thus for a specific media element $i$, the mapping $TL(i, b)$, $b \geq 0$, represents the full set of interactor events for the media element $i$. Additionally, there is the constraint that an Interactor may not be applicable to every type of medium (e.g. the color attribute of an ink stroke is not applicable to audio).

The original Interactors conceptual model defines four broad classes of features related to the user-media interaction [7, 21]: *(a) time*, given that each interaction event is timestamped to the start of a capture session; *(b) attributes*, related to media features collected during the capture, such as color and thickness (for ink strokes) as well as pitch and noise (for audio), and so on; *(c) action*, given that a user can perform several actions over the media element, such as drawing, erasing, muting, etc.; and *(d) position*, which considers the boundary limits of the interaction event when occurred over a surface, e.g, cartesian coordinates of an ink stroke.

While experimenting with Interactors, we noticed that *time*, rather than being characterized as an isolated category, can be used in conjunction with the other categories of features as well. Based on this premise, in this research we report the extension of the original set of Interactors in order to consider two new requirements: *(i)* the need for a list of time moments to be returned by an operator; and *(ii)* the need to include the time interval in which the media are to be processed.

In this context, we define a time moment as an instant in the annotation session and a time interval as a segment delimited by two time moments. The following sections include only the ink-based and audio-based Interactors redefined to fulfill these new requirements.

## 2.1 Inkteractors revisited

*Inkteractors* are a special type of Interactors obtained by processing features of user-ink interactions — which is common when information is captured from meetings, classroom or museums, for instance, by means of pen-based devices such as electronic whiteboards or tablets. These operators can be applied to ink strokes so as to allow the generation and playback of documents containing alternative views of the original ink-based interaction process. In the listing below, the original specifications of some Inkteractors have been updated to tackle the new requirements.

***Time-based.*** Used to filter or expand the media elements based solely on time constraints.

- TimeSlice(time $StartTime$, time $EndTime$): returns a list of the ink strokes generated in the specified time interval — this can be used to generate an image which aggregates the returned strokes, for instance;

***Attribute-based.*** Considers attributes of pen strokes, such as color, thickness, type of stroke (free-form ink, geometric shape, etc.).

- ChangeOnAttribute(attribute $A$, time $StartTime$, time $EndTime$): returns a list of time moments when ink strokes were changed according to attribute $A$ within the time interval defined by $StartTime$ and $EndTime$ – this can be used to generate an index to a timeline corresponding to a change in the color of the stroke, for instance;
- FilterByAttributeValue(attribute $A$, value $V$, time $StartTime$, time $EndTime$): returns a list of time moments when ink strokes were changed so that attribute $A$ is equal to value $V$, within the time interval defined by $StartTime$ and $EndTime$ — this can be used, for instance, to generate an index to a timeline corresponding to the time moments when a specific color was used.

***Action-based.*** While interacting with the capture system, a user can perform several actions over the pen strokes, such as drawing, erasing, changing color and so on. The history of such actions is kept together with the stroke representation, as well as with author who performed each one of them.

- $ChangeOnAuthor$($time$ StartTime, $time$ EndTime): returns a list of time moments when there was a change

[1]Inkteractors [7].
[2]AudioInteractors [22].

in the author of the strokes, within the time interval defined by *StartTime* and *EndTime* – this can be used to generate an index to a timeline corresponding to a change in the author of a stroke, which may be of interest in distributed systems in which several users may draw strokes on a common surface;
- *FilterByAuthor*(*id* ID, *time* StartTime, *time* EndTime): returns a list of time moments when there was a change in the author of the strokes so that the author is identified by *ID*, within the given time interval — this can be used to generate an index to a timeline corresponding to the time moments when the author started producing strokes after someone else.

***Position-based.*** Drawing surface and strokes are represented as a set of points in cartesian coordinates. Boundary limits, i.e., minimum and maximum values on both X and Y axes, are also recorded for each stroke.

- *ChangeOnArea*(*coord X*, *coord Y*, *time* StartTime, *time* EndTime): returns a list of time moments when there was a change in the specified area;
- *FilterByArea*(*coord X*, *coord Y*, *time* StartTime, *time* EndTime): returns a list of ink strokes drawn in the given area during the specified time interval.

## 2.2 AudioInteractors revisited

*AudioInteractors* are a special type of Interactors obtained by content-based analysis of user speech files captured in a session. AudioInteractors have been categorized as time-based and attribute-based [22]. In the listing below, the specification of the operators has been updated to reflect the new requirements.

***Time-based.*** Obtained by detecting patterns on the digital audio file (during the post-production phase). They can be used to identify time moments or time intervals in the speech when a specific voice behavior applies.

- silenceMoments(time *Tmin*, time *StartTime*, time *EndTime*): returns a list of time moments when there were no voices for at least *Tmin* units of time during the time interval defined by *StartTime* and *EndTime*.
- spokenMoments(time *Tmin*, time *StartTime*, time *EndTime*): returns a list of time moments before which someone has spoken for at least *Tmin* units of time during the time interval defined by *StartTime* and *EndTime*.

***Attribute-based.*** There are digital audio attributes (e.g. frequency, pitch, noise, amplitude) that can be exploited to detect points of interest within the media element. Attribute-based operators include the following, also found in an updated version as detailed earlier.

- voiceIncrease(time *StartTime*, time *EndTime*): returns a list of time moments when there was a consistent increase in speech volume in the given time interval.
- conversation(time *StartTime*, time *EndTime*): in reference to the given interval $T$ in the audio file, returns the potential number of participants who spoke during this interval.
- outstandingMoments(time *StartTime*, time *EndTime*): return a list of time moments when there were outstanding moments in the media element. Outstanding moments are those when several people are talking at the same time with consistent increase in the volume of their voices.

Various content-based methods can be employed to derive audioInteractors. As an example, the computation of the time-based operator *SilenceMoments()* using wavelet transformations [12], in particular the *Haar transform*, has been detailed elsewhere [22]. An important category of AudioInteractors not explored in previous work is the one associated with a user explicitly activating the audio mute function:

***Action-based.*** Obtained by capturing moments in which a user activated the mute function of the microphone; examples are:

- enterAudioMute(time *StartTime*, time *EndTime*): returns a list of time moments, within the given time interval, when the mute function was activated.
- exitAudioMute(time *StartTime*, time *EndTime*): same as above, but with the mute function deactivated.

# 3. NOVEL INTERACTORS

It is very common that users exchange text-based messages and interact with board-like surfaces to present slides during collaborative activities. Targeting at these requirements, the following sections will define novel Interactors in order to tackle these issues, namely *TextInteractors* to encompass interactions via text messages and *BoardInteractors* to encompass interactions with board-like surfaces. Additionally, *VideoInteractors* are also defined to encompass user-image interactions.

## 3.1 TextInteractors

*TextInteractors* are a special type of Interactors obtained by identifying user-user interaction by means of textual (typed) messages: the messages can, for instance, be exchanged in a special Chat window which provides the message exchange service. These are also divided into two categories:

***Attribute-based.*** Many attributes of text messages can be collected during capture, including font type and color.

- ChangeOnAttribute(attribute *A*, time *StartTime*, time *EndTime*): returns a list of time moments when the text message were changed according to attribute *A* within the time interval defined by *StartTime* and *EndTime* – this can be used, for instance, to generate indexes to a timeline corresponding to a change in the given attribute (e.g., the time moments when there was a change of the font color of the text message);
- FilterByAttributeValue(attribute *A*, value *V*, time *StartTime*, time *EndTime*): returns a list of time moments when text messages were changed so that attribute *A* is equal to value *V*, within the time interval defined by *StartTime* and *EndTime* — this can be used, for instance, to generate an index to a timeline corresponding to the time moments when the text color was set to blue.

***Time-based.*** Obtained by detecting intervals of message exchange or their inexistence.

- silenceMoments(time *Tmin*, time *StartTime*, time *EndTime*): returns a list of time moments when there were no messages exchanged for more than $T$ ($T >$

$Tmin$) units of time, during the time interval defined by $StartTime$ and $EndTime$.
- textMoments(time $Tmin$, time $StartTime$, time $EndTime$): returns a list of time moments before which someone has typed messages for at least $Tmin$ units of time during the time interval defined by $StartTime$ and $EndTime$.

## 3.2 BoardInteractors

There are many situations in which users make use of board-like surfaces to deliver slide show presentations in meetings and lectures, for example. Some web conferencing systems offer board surfaces to present slides for discussion by the group; Webcast[3] applications may use boards for the presenters to deliver their talks using slides and audio, for instance. There are also applications in which the slide show may be the only information presented, such as Slideshare[4], for example. Given the wide use of boards independently of ink-based annotation, we observed the need for defining operators directly associated with boards. These are also divided into two categories:

***Time-based.*** Obtained by monitoring user-image interaction in the capture phase. The operators can be used to build a timeline with time moments corresponding to when one particular slide was presented.

- ChangeBoard(time $StartTime$, time $EndTime$): returns a list of time moments when there was a change of slide within the specified time interval – this can be used to generate a corresponding timeline;
- IdleBoard(time $T$, time $StartTime$, time $EndTime$): returns a list of time moments when there was no change of slide for at least $t$ seconds within the specified time interval.

***Atribute-based.*** There are attributes from the slides (whether a slide has text, image, animation, etc.) which can be used to detect slides of interest.

- ChangeOnAttribute(attribute $A$, time $StartTime$, time $EndTime$): returns a list of time moments when there was a change according to attribute $A$ within the time interval defined by $StartTime$ and $EndTime$ – this can be used to generate an index to a timeline corresponding to a change in the attribute (e.g., the time moments when there was a change from text-based slide to a slide containing an animation);

- FilterByAttributeValue(attribute $A$, value $V$, time $StartTime$, time $EndTime$): returns a list of time moments when slides where changed so that attribute $A$ is equal to value $V$, within the time interval defined by $StartTime$ and $EndTime$ — this can be used, for instance, to generate an index to a timeline corresponding to the time moments when the attribute contains the given value (e.g., the time moments when the slide contained an animation).

## 3.3 VideoInteractors

In lecture webcasts, such as those available in MIT OpenCourseware and Google Tech Talks, slides are captured by recording a video of the projected content. In order to enable a proper retrieval of the content, systems such as Talkminer [3] analyze the captured video by segmenting it in keyframes that represent changes in the presented slides. Aiming at such scenarios, in this section we define *VideoInteractors* as an operator obtained by identifying user-image interactions: the images may, for instance, be a set of slides or a set of photos. These are also divided into two categories:

***Time-based.*** Obtained by monitoring user-image interaction in the capture phase. The operators can be used to select moments or intervals when an image was reviewed.

- blankMoments(time $Tmin$): moments when no image was presented for $T$ ($T > Tmin$) units of time.
- imageMoments(time $Tmin$): returns the moments before which a user was presented with an image for $T$ ($T > Tmin$) units of time.
- imageIntervals(time $Tmin$): returns the start moment of time intervals in which a user was presented with an image for $T$ ($T > Tmin$) units of time.

***Attribute-based.*** There are attributes from the image (type, size, etc.) that can be exploited to detect images of interest.

- imageType(string $Type$): returns moments when an image of the given type was presented (e.g. PNG or JPG)
- imageSize(int $Size$): returns moments when an image of the given size was presented.
- others: one can also define operators based on features of images such as edges and corners (for JPEG), and level of alpha channel (for PNG).

# 4. PROOF-OF-CONCEPT PROTOTYPE

DiGaE (*Distributed Gathering Environment*) is a capture environment for collaborative meetings that can be used both in instrumented rooms or in webconferencing mode. When used for webconferencing purposes, a special configuration called DiGaE Home provides a web-based tool to capture audio and video streams from the desktop computer, as well as other communication tools such as instant text-based messaging and synchronous software-based whiteboard.

We employed the DiGaE Home tool in a meeting with a remote lecturer and a group of attendants that used whiteboard software to load a set of slides and make ink-based annotations, a chat session to provide text-based message exchange, and video+audio conferencing. User-board and user-ink interactions as well as textual messages and the audio+video files were recorded. The dynamics of the meeting required that attendants took turns with respect to talking, asking, drawing and typing text.

**Table 1: Interactors used in the prototype.**

| Category | Operator | Attributes |
|---|---|---|
| Inkteractors | FilterByAuthor | author |
| AudioInteractors | spokenMoments | - |
| TextInteractors | filterByAttributeValue | authors |
| BoardInteractors | filterByAttributeValue | slides titles |
| BoardInteractors | changeBoard | - |

The recorded session was exported to an XML data interchange document, which was processed and enriched with the Interactors listed in Table 1. Given that the enriched

[3]http://www.webopedia.com/TERM/W/Webcast.html
[4]http://www.slideshare.net

document must be transformed into an appropriate interactive multimedia format for playback, in Figures 1 and 2 we show the use of an iMMD in NCL (declarative) [17] with Lua (procedural) [9].

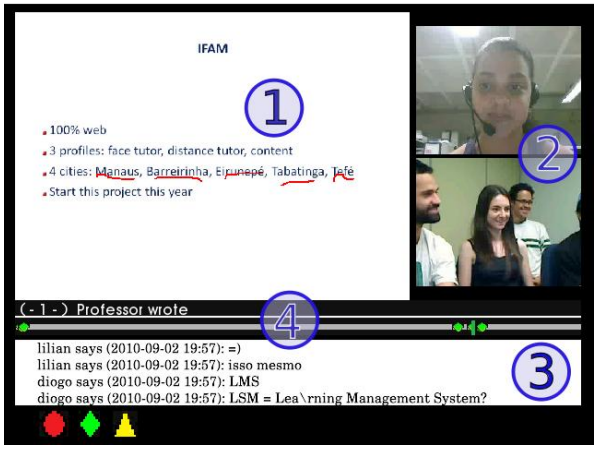


**Figure 1: Structure of the final interactive multimedia document illustrating the timeline decorated with text-based Interactors.**

Figure 1 depicts the layout of the final iMMD: its playback is controlled by a player which offers several interaction options by means of a keyboard when the playback occurs in a computer-based platform, for instance, or a remote control when the playback occurs in a TV-based platform. The interface contains: *1*) the whiteboard region; *2*) the video region; *3*) the chat region; and *4*) the timeline region.

In Figure 1, the color buttons in the bottom left corner (red circle, green diamond and yellow triangle) can be used to toggle media-specific Interactors to decorate the timeline. The timeline enables users to navigate to different points of interest within media elements. For example, the decorated timeline of Figure 1 shows the interactor events related to the filterByAttributeValue() TextInteractor, with author as attribute, after pressing the green button, and it indicates time moments in which there were professor-text interactions during the meeting.

Additionally, a label is included above the timeline to indicate the specific Interactor that is active. If other Interactors are available for the same media element, both up/down navigation keys, and also a numerical code, can be used to select them. In Figure 2, the red button and then the up/down keys were used to activate the ChangeOnAttribute() BoardInteractor, regarding the text title as attribute; it is clear that the presentation updates the label and the corresponding points of interest in the timeline. In order to navigate to a particular point of interest, users employ the right/left navigation buttons in the playback interface to focus on the desired point, and press the *ok* (or *enter*) button when finished.

## 5. EVALUATION WITH USERS

The evaluation aimed at measuring the effectiveness of Interactors while assisting users during search tasks to lookup specific facts in a captured session. We carried out the think aloud protocol [23] with four participants which were present in the recorded proof-of-concept scenario. All participants had previous experience browsing the Web and using TV remote control.

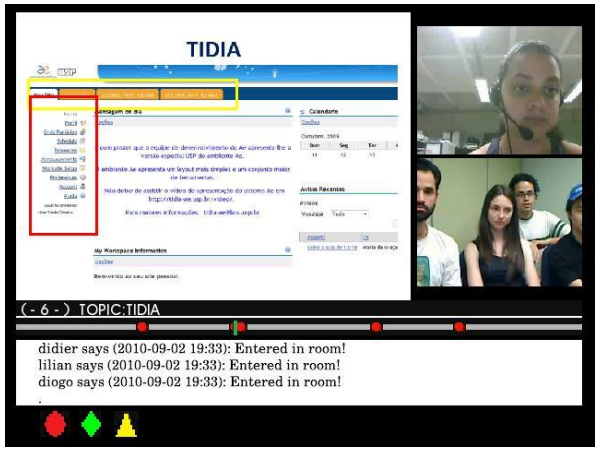


**Figure 2: Screenshot illustrating the timeline decorated with ink-based Interactors.**

The participants received a brief introduction to Interactors, the main interface elements and a list of four simulated search requests about the meeting. Users were asked to answer each request by navigating the iMMD timeline enriched with operators. Through the test, all user navigation events (keystrokes) were captured and documented in an XML document. At the end of the test, each user participated in an interview regarding his understanding of the concept and usefulness of the tool.

The majority of the users felt comfortable with the possibility of navigating points of interest in the recorded session and acknowledged about the usefulness of the media-oriented operators. As a shortcoming, users had trouble to associate each available button to the corresponding Interactor, revealing learnability problems in the interface. Furthermore, choosing the best set of operators to answer the request was also pointed out as a problem, specially in the beginning of the interaction, when users were not very aware of the possibilities of the tool. Aside from these problems, users felt comfortable navigating the session and appreciated the easiness to jump to points of interest. On average, users spent 11 min. to complete a search task, a clear speed up when compared to the need of linearly watching the whole session (25 min.). Overall, users also explicitly acknowledged that Interactors enabled faster review of the recorded meeting.

## 6. RELATED WORK

Much research on meeting browsers have focused on navigation of a single type of medium, for instance audio, video or whiteboards. Concerning navigation of audio recordings, Ehlen et al. [10] describes an interface for timeline-based meeting review and user feedback that concerns only navigation of speech transcripts via events of interest (e.g. decisions, minutes). Similar limitations are observable in video-only browsers: Yu et al. [25] presents a system to navigate segmented video recordings through events of user-user interaction (e.g. propose, request information, acknowledge, etc.). Another video-only approach is reported by Behera et al. [4], which provides a browser to query a repository

of SMIL documents generated from recordings of projected slides: using the SMIL browser, users can navigate segments of the retrieved documents. Even though an approach to generate interactive multimedia documents is advocated, it focuses only on one type of medium.

whilst recurrent in the literature, such exclusive focus on just one type of medium hinders applicability of a meeting browser to environments that can record heterogeneous media sources, such as webconferencing systems and smart meeting rooms. A limitation of these researches is the restriction to a single tool, hindering their generalization to environments with additional synchronous collaboration tools.

# 7. CONCLUSIONS AND FUTURE WORK

We have formalized the concept of Interactors as media-oriented operators to assist the review of collaborative activities, through the automatic generation of interactive multimedia documents from the captured media. We have extended previous work by redefining ink and audio Interactors and, more importantly, we defined novel operators related to video, text and board Interactors. We demonstrated that Interactors provide means for generating media indexes related to several types of timestamped media-based operators, in order to enrich iMMDs of recorded sessions.

For evaluation purposes, we have experimented with a proof-of-concept prototype using multimedia information captured in a webconferencing tool and generated interactive multimedia documents enhanced with Interactors. Our evaluation results suggest that Interactors are effective means to assist the review of captured collaborative sessions by navigating points of interest. Even though we noticed problems related to the selection of the operators in the interface, in general, users acknowledged about the easiness and effectiveness to review the session using points of interest.

As far as future work is concerned, we plan to formalize the definition of new operators, to improve the reviewing interface to tackle the problems elicited in the evaluation, and to perform usability inspections and new user tests.

**Acknowledgments**. We thank FAPESP, CNPq, CAPES, FINEP, MCT and the users, specialists and anonymous reviewers for the many important suggestions.

# References

[1] G. Abowd, E. Mynatt, and T. Rodden. The human experience [of ubiquitous computing]. *IEEE Pervasive Computing*, 1(1):48–57, 2002.

[2] G. D. Abowd, C. G. Atkeson, J. Brotherton, T. Enqvist, P. Gulley, and J. LeMon. Investigating the capture, integration and access problem of ubiquitous computing in an educational setting. In *ACM CHI'98*, pages 440–447, 1998.

[3] J. Adcock, M. Cooper, L. Denoue, H. Pirsiavash, and L. A. Rowe. Talkminer: a lecture webcast search engine. In *ACM MM '10*, pages 241–250, 2010.

[4] A. Behera, D. Lalanne, and R. Ingold. DocMIR: An automatic document-based indexing system for meeting retrieval. *Multimedia Tools and Applications*, 37(2):135–167, 2007.

[5] M.-M. Bouamrane and S. Luz. Meeting browsing. *Multimedia Systems*, 12(4-5):439–457, October 2006.

[6] S. Branham, G. Golovchinsky, S. Carter, and J. T. Biehl. Let's go from the whiteboard: supporting transitions in work through whiteboard capture and reuse. In *ACM CHI'10*, pages 75–84, 2010.

[7] R. G. Cattelan, C. Teixeira, H. Ribas, E. Munson, and M. Pimentel. Inkteractors: interacting with digital ink. In *ACM SAC'08*, pages 1246–1251, 2008.

[8] M. Czerwinski, D. W. Gage, J. Gemmell, C. C. Marshall, M. A. Pérez-Quiñones, M. M. Skeels, and T. Catarci. Digital memories in an era of ubiquitous computing and abundant storage. *Comm. ACM*, 49(1):44, 2006.

[9] R. de Mello Brandão, G. de Souza Filho, C. Batista, and L. Gomes Soares. Extended Features for the Ginga-NCL Environment: Introducing the LuaTV API. In *Computer Communications and Networks (ICCCN)*, pages 1 –6, 2010.

[10] P. Ehlen, M. Purver, J. Niekrasz, K. Lee, and S. Peters. Meeting adjourned: off-line learning interfaces for automatic meeting understanding. In *ACM IUI '08*, pages 276–284, 2008.

[11] W. Geyer, H. Richter, and G. D. Abowd. Towards a Smarter Meeting Record-Capture and Access of Meetings Revisited. *Multimedia Tools and Applications*, 27(3):393–410, 2005.

[12] R. C. Guido, J. F. W. Slaets, R. Köberle, L. O. B. Almeida, and J. C. Pereira. A new technique to construct a wavelet transform matching a specified signal with applications to digital, real time, spike, and overlap pattern recognition. *Digital Signal Processing*, 16(1):24–44, 2006.

[13] L. Hardman. Canonical Processes of Media Production. In *MHC '05: Proceedings of the ACM workshop on Multimedia for human communication*, pages 1–6, 2005.

[14] D. Kirk, A. Sellen, R. Harper, and K. Wood. Understanding videowork. In *ACM CHI '07*, pages 61–70, 2007.

[15] S. Minneman, S. Harrison, B. Janssen, G. Kurtenbach, T. Moran, I. Smith, and B. van Melle. A confederation of tools for capturing and accessing collaborative activity. In *ACM MULTIMEDIA'95*, pages 523–534, 1995.

[16] M. G. C. Pimentel, Y. Ishiguro, B. Kerimbaev, G. Abowd, and M. Guzdial. Supporting educational activities through dynamic web interfaces. *Interacting with Computers*, pages 353–374, 2001.

[17] L. F. G. Soares, R. F. Rodrigues, R. Cerqueira, and S. D. J. Barbosa. Variable handling in time-based XML declarative languages. In *ACM SAC '09*, pages 1821–1828. ACM, 2009.

[18] C. A. C. Teixeira, G. B. Freitas, and M. Pimentel. Distributed discrimination of media moments and media intervals: a watch-and-comment approach. In *ACM SAC'10*, pages 1929–1935, 2010.

[19] K. N. Truong and G. R. Hayes. Ubiquitous computing for capture and access. *Found. Trends Hum.-Comput. Interact.*, 2(2):95–171, 2009.

[20] S. Tucker and S. Whittaker. Accessing Multimodal Meeting Data: Systems, Problems and Possibilities. In *Proc. Work. Machine Learning for Multimodal Interaction*, pages 1–11, 2004.

[21] D. A. Vega-Oliveros, D. S. Martins, and M. G. C. Pimentel. "This conversation will be recorded": automatically generating interactive multimedia documents from captured media. In *ACM DOCENG '10*, 2010.

[22] D. A. Vega-Oliveros, D. S. Martins, and M. G. C. Pimentel. Interactors: operators to automatically generate interactive multimedia documents from captured media. In *Proc. Brazilian Symposium on Multimedia and the Web*, 2010.

[23] P. C. Wright and A. F. Monk. The use of Think-Aloud Evaluation Methods in Design. *SIGCHI Bull.*, 23(1):55–57, 1991.

[24] Z. Yu and Y. Nakamura. Smart meeting systems: A survey of state-of-the-art and open issues. *ACM Computing Surveys*, 42(2):1–20, 2010.

[25] Z. Yu, Z. Yu, H. Aoyama, M. Ozeki, and Y. Nakamura. Social interaction detection and browsing in meetings. In *Proc. Intl. Conf. Ubiquitous Computing*, pages 40–41, 2008.