A Case Study of Prototyping a Multimodal User Interface for a Media Annotation Tool

Dominik Ertl, Marie Kavallar, David Raneburger
Vienna University of Technology, Institute of Computer Technology
A-1040 Vienna, Austria
{*dominik.ertl, marie.kavallar, david.raneburger*}*@tuwien.ac.at*

***Abstract*—Media annotation is the process of adding annotations to media, like audio or video data. Annotations are, e.g., emotion descriptions of human emotions. The manual creation of annotations typically requires to repeat small tasks many times. Manual annotation is time-consuming and erroneous because user interfaces (UI) for such annotation tools often lack the possibility of multimodal interaction. In this work, we present a case study were we prototyped multimodal UIs for media annotation. First, we identified time-consuming tasks in the process of media annotation. Then we studied the human-computer interaction, to find out which modality combinations fit well for these tasks. This led us to suitable variants for modality combinations, like speech input, mouse gestures, earcons and an adapted GUI. We used the OpenInterface platform to implement prototypes of these multimodal UI variants for an existing GUI-based media annotation tool. Our prototyping approach allows easy change and adaptation of the multimodal UI. This supports the designer during the multimodal UI development and leads to UIs for media annotation tools that have a well-balanced set of modalities for interaction purposes.**

***Keywords*-Multimodal, Media Annotation, Prototyping**

I. Introduction

Media annotation is the process of adding meaningful annotations like the emotional state of a human to media content, e.g., audio or video data. Research and industry provide dedicated annotation tools for different kind of media, like audio/video streams and files, virtual worlds, etc. Manual media annotation for audio and video is typically done with dedicated time line-based annotation tools like ANVIL [2] or ELAN [3] that usually come with a WIMP (Window-Icon-Menu-Pointer) based UI. Manual media annotation, however, is still an error-prone and expensive task.

In our work, we present a case study where we improved such a time line-based media annotation tool with multimodality. Multimodality — as presented in the work of Reeves et al. [13] — allows to make human-computer interaction more robust.

In our case study we focused on prototyping of such a multimodal UI and the interaction with them. Therefore, we studied the interaction for various modalities and modality combinations like speech input, mouse gestures, a 3D-mouse, etc. This helps to better understand the interaction needs of annotation tool users and therefore, to further improve the process of manual annotation. In our approach we used an existing media annotation tool, the Smart Sensor Integration (SSI) [8], which we coupled the OpenInterface (OI, Lawson et al. [11]) platform for prototyping the UI. This combination supports rapid prototyping of multimodal UIs for media annotation tools and thus, supports the designer in finding a suitable combination of modalities. Figure 1 provides an overview in chronological order of the work done in our case study.

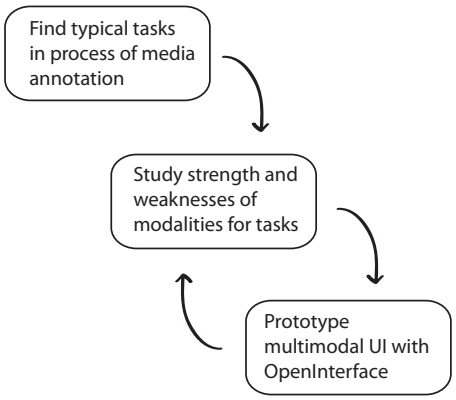

Figure 1. Overview of our Prototyping Approach.

The remainder of this work is organized in the following way: First, we provide background information about media annotation and prototyping of multimodal UIs. Subsequently, we describe how we studied interaction and implemented a multimodal UI prototype for media annotation. Then we discuss the lessons learned, considering benefits as well as drawbacks and pitfalls. Finally, we present related work.

II. Background

The work of Wagner et al. [9] points out that we have to give the computer access to human generated signals and provide adequate models to recognize and interpret behavioral patterns. The annotation process can be done either automatically or manually. Automatic annotation is often based on pattern matching with statistical models (Lavrenko et al. [10]). For example, an audio file is automatically

Copyright (c) IARIA, 2011. ISBN: 978-1-61208-117-5

screened for specific emotions like giggling or crying. If an emotion pattern matches, the automatic annotation tool stores metadata that includes start and end time of the detected pattern, as well as the emotion description. Even if the automatic annotation tools are improving, a human user usually has to check the resulting annotations and adapt them, to achieve a good level of quality. Adapting the annotations has to be done manually if there are no existing algorithms that can detect specific patterns. For example, detecting the color of t-shirts in video files (if this is of interest). In principle, the annotations of media can be stored in a data base. It is then possible to search for specific media sequences via a search mechanism. For our work it does not matter what the annotations are used for, since we mainly focus on the multimodal UI of the annotation tool.

We developed a prototypical multimodal UI based on the SSI [9] tool. SSI is a framework for multimodal signal processing in real-time. It has two functionalities, the acquisition of audio and/or video data and its annotation. In our work, we do not enhance the underlying signal processing tool, but the ModelUI, which is a WIMP-GUI that runs on top of SSI.

For a robust interaction the designer has to optimize the interplay of several modalities. Prototyping a multimodal UI is often helpful to get a deeper understanding of what users need when interacting with a special purpose annotation tool like SSI. Prototyping, moreover, allows to easily attach and detach modalities at design time. In our case study we used OI from Lawson et al. [11], to implement our multimodal UI. OI is a free visual programming environment to prototype interaction. OI enables the UI designer to link components together to design the interaction of a multimodal UI. The documentation of OI is up-to-date and comes with a tutorial and in-depth information about using OI.

III. Multimodal UI for Media Annotation

In this work, we studied and enhanced the SSI annotation tool introduced by Wagner and Andre [8], which is intended to train models for improved recognition of human input behavior. As shown in Figure 1, we identified common tasks for timeline-based media annotation as a first step. Then we iterated over the steps *defining interaction* and *prototyping (implementing) the UI*. In the following, we present our approach in more detail.

A. Tasks in Media Annotation

The decision for the SSI tool as a basis was based on two reasons : First, it is a straightforward media annotation tool that only has a graphical UI. Second, the source code is available for free.

Since some of the code for annotation functionality was mixed up with the UI code, we first had to extract the modality independent functionality from the UI code, to clearly separate the application logic from the UI (see Figure 2, the yellow boxes depict our work). The extracted functionality is typical for timeline-based media annotation tools: start/stop playback of media content with a media player, add/remove annotation elements (segments), edit annotations, add several tracks (for different annotation levels), etc. In Figure 3 we depict the adapted SSI interface where some of the extracted functionalities are marked.

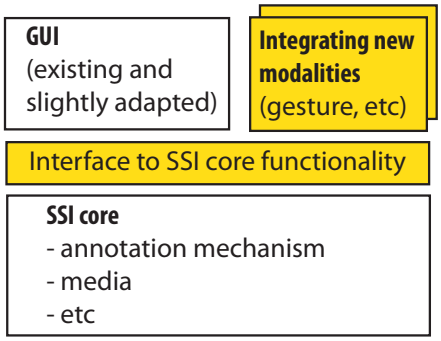

Figure 2. SSI Architecture.

As a second step, we studied the interaction with several media annotation tools, including other tools like ANVIL or ELAN besides SSI. Subsequently we annotated several audio and video files (with a lengths between 1 and 5 minutes) with the existing SSI tool. We measured the time for the different working steps and identified the following set of tasks as the most time consuming ones in terms of human-computer interaction:

- *Create and select annotation tracks.* An annotation track is a container to which annotation segments can be added. An annotation segment has a defined start and endpoint on the time line and contains the annotation value. For example, the annotation track *Emotions* is intended to store annotation segments in a timely order. Each segment contains an emotion description (e.g., laughing, crying, etc.) of humans that appear in this media. In SSI the assignment of emotions to segments on the time line has to be done manually.
- *Segment and reorder segments on the time line.* This includes finding the correct start- and endpoint for an annotation segment. Segments can also be shifted on the time line. Moreover, the user selects/deselects the segments she deals with.
- *Edit segment annotation values.* Creating or editing an annotation value of a segment differs from tool to tool. It is therefore, important to know if the annotation has to be machine readable (e.g., XML format) or just understandable for humans.

Typically, a manual annotation process consists of these small tasks that are repeated over and over again.

Copyright (c) IARIA, 2011. ISBN: 978-1-61208-117-5

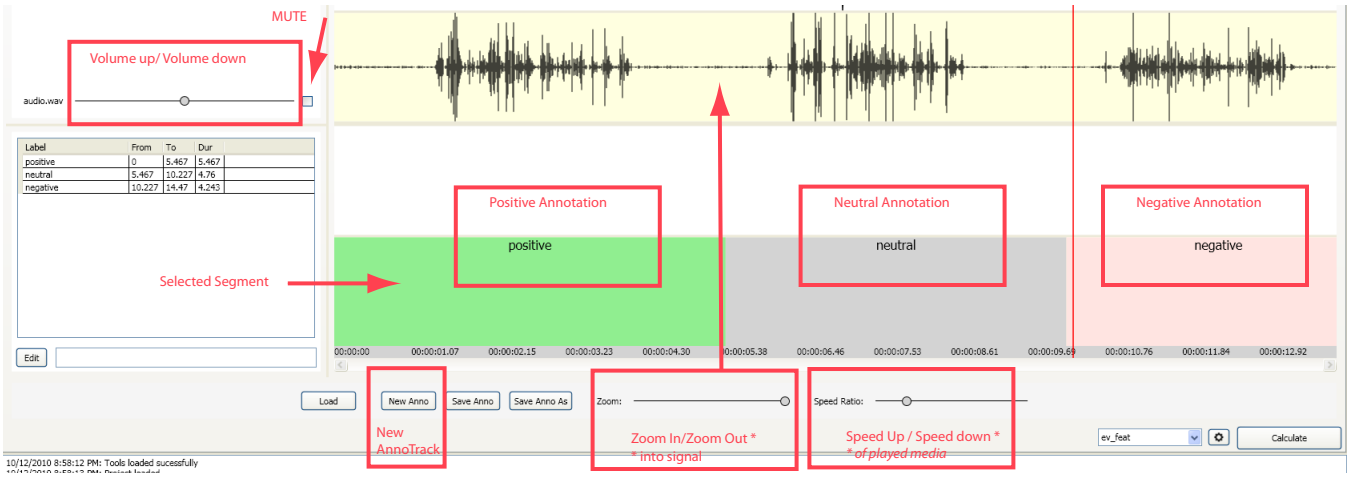

Figure 3. Adapted SSI [9] Graphical UI.

B. Modalities for Interaction

To better understand how interaction can be improved, we studied several modalities with different interaction devices. This way we identified how the inherent strengths and weaknesses of the modalities allow for a robust interaction with media annotation tools.

- *WIMP-based GUI*: This was the existing UI of the SSI tool. A GUI normally supports mouse and keyboard input. The inherent strength of a GUI are the parallel and permanent presentation of content, as well as the *100% recognition rate* (for example you can be sure the user clicked a button). We suggest to use a GUI as the main output modality and as an important input modality.
- *Key binding*: Key bindings are typical short-cuts of keyboard entries that call a certain functionality in the program. This is fairly common in any programming or editing environment. However, it was not included in the SSI toolkit. So we added freely configurable interaction via key bindings. For example, we coupled key *1* to the functionality *set emotional state to positive emotion*.
- *Mouse gestures*: In contrast to normal mouse scrolling and clicking behavior, mouse gestures are like key bindings. We assume that a user that works with an annotation tool often uses mouse and keyboard together. This means that performing a mouse gesture does not require changing the interaction device — which is important for acceptance of a hand-based modality (see Figure 4 for a subset of gestures that we defined). These gesture definitions can also be used for pen-based gestures, WII-mote or other modalities. Such a mouse gesture can be performed within 1-2 seconds and does not require extra skills for anyone who can use a mouse. We used a constrained set of at most 8 gestures for often used commands, because this is a memorable number of gestures.
- *Vocal speech input*: In contrast to the other modalities presented here, speech input does not require an interaction device that is used with one or two hands. Vocal speech input is useful for command-based interaction, like *play*, *add track* and so on.
- *Speech output*: We distinguish between earcons and vocal speech output. Earcons are brief, structured sound patterns that represent an event. They are helpful during media annotation (e.g., an earcon is played if a segment has been added). In contrast, we found that vocal speech output is not helpful in the context of media annotation. First, the serial nature of speech output does not allow to present several tasks in parallel, e.g., to inform a user what she can do next. Second, it hinders a user to concentrate while annotating the media.
- *Jogwheel*: A jogwheel like the shuttleXpress device [4] is a special purpose interaction device that combines buttons with key bindings and scroll wheels. Jogwheels are a good choice for media annotation tools: the issuing of commands via buttons and scrolling on the time line with the wheel(s) is frequently needed.
- *WII-mote*: A WII-mote [6] is a device to perform hand gestures in the air. It is connected via bluetooth to the computer and can be used for a wide number of 3D gestures. However, the process of media annotation requires a lot of precisely given commands that are executed in a chronological order in a relatively short timespan (few seconds between two tasks). People start to get tired just by performing gestures with the WII after a few minutes. So, using a WII-mote did not turn out to be helpful in the process of media annotation — even if the fun-factor is big at the beginning.
- *3D space navigator*: A 3D space navigator [1] is is intended to ease the navigation in a virtual 3D environ-

Copyright (c) IARIA, 2011. ISBN: 978-1-61208-117-5

ment. It also has several buttons like a normal computer mouse. Such a device is mainly useful for CAD-tools or GoogleEarth-like applications. As it is not possible to lock the 3D-maneuverability for 2D scrolling only, using the device is not intuitive compared to the scroll wheel of the jogwheel. We found out that a user often triggered other commands unintentionally, since three axes allow for several possible commands. Therefore, we did not consider the 3D space navigator for the final UI variants.

- *Pen/finger-based gesture*: Pen-based gestures are popular on smart phones and PDAs for navigation and simple commands like starting an application. The media annotation discussed in this work, however, usually involves a desktop or personal computer with a big screen. So, pen-based gestures are out of focus.
- *Multi-touch display*: A multi-touch display is useful for scrolling and zooming in/out, but not for the process of media annotation. Moreover, after a few minutes it gets annoying for the user to have the finger as main input device. Here, we require further studies about satisfaction with multi-touch displays to fulfill longer-lasting work tasks. For this reason we did not consider multi-touch displays for our multimodal UI after studying it for the purpose of media annotation.

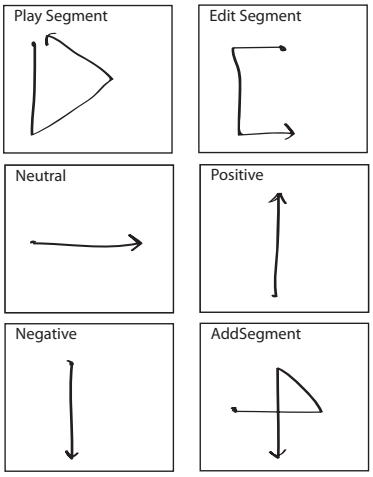

Figure 4. A Set of Gestures Intended for Media Annotation.

C. Prototyping the Multimodal User Interface

We prototyped the multimodal UI with the visual programming platform OI [11]. We carefully assigned the modalities to the defined tasks in order to maximize the advantage of each modality according to Reeves et al. [13]. Furthermore, we only used a constrained grammar set for speech input, to keep the UI consistent. With OI, we defined the interaction of a UI by coupling modalities for human input and output with application components. Besides, *helper* components like data logging units can be included.

In Figure 5 we show the visual programming for one variant of our multimodal UIs. Figure 5 depicts the coupling of our multimodal UI with speech input, mouse gesture and the adapted SSI component. An OI component is either used to forward and transform signals, like the multicast component, a modality component or an application logic component. An OI modality component is an interface to byte code of a modality, e.g., to an external speech input toolkit. We implemented the components SSIOI, LoggerOI, JuliusOI and the according coupling between the components. We executed the interaction pipeline presented in Figure 5 to start the SSI application and its UI with OI.

The *SSIOI* component on the right side of 5 is the interface shown in Figure 3. This SSI component starts the SSI core. Additionally, it starts the GUI of SSI, which is not an OI component. The speech input modality component *JuliusOI* provides the coupling to the Julius [5] speech input toolkit. We configured the recognition grammar for Julius at design time. Furthermore, we included the mouse gesture modality. Here, we did not use one gesture component, but coupled several already existing components together to implement the intended functionality. In principle, a mouse gesture is sent to the SSI component when the user presses the right mouse button, moves the mouse in 2D-space according to a defined gesture and releases the button again. This interaction is realized with the components *DirectXMouseComponent*, *IfThenElse*, *OnInputChanged* and the *GestureRecognizer*. DirectXMouse forwards the x/y coordinates and the pressed button status to a multicast component. The multicast component sends this data to the GestureRecognizer and the IfThenElse component at the same time. The IfThenElse component forwards the status of the mouse button to the OnInputChanged component. The OnInputChanged component sends a continuous signal to the GestureRecognizer, indicating that the right mouse button is pressed. As long as this signal is active, the GestureRecognizer records the gesture. If the right mouse button is released, OnInputChanged sends a signal to stop the gesture recognition, and the GestureRecognizer interprets the recorded data and forwards the best matching gesture to the SSI component.

IV. Lessons Learned

The prototyping of media annotation tools with OI has benefits as well as pitfalls. In this section we will describe those that we encountered during our case study and how they can be generalized for similar projects.

A. Proposed UI variants

We suggest to use the GUI with key bindings, speech input and mouse gestures as modalities of the prototyped UI. Alternatively, if a jogwheel is available , we suggest a variant with jogwheel, keyboard and GUI. The jogwheel is in this context a very interesting device, due to its strength

Copyright (c) IARIA, 2011. ISBN: 978-1-61208-117-5

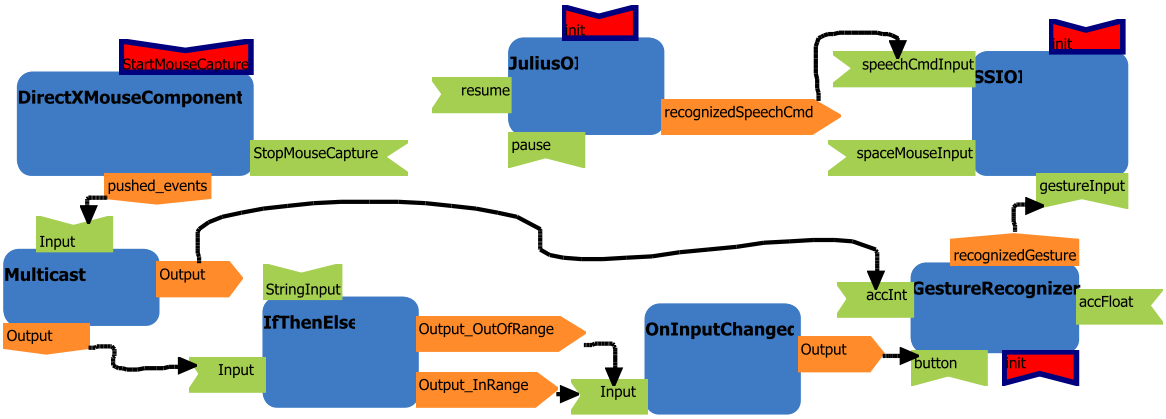

Figure 5. Building a Multimodal UI for Media Annotation with OI.

for scrolling and button-bindings. Both variants should have earcons as output to confirm typical commands.

B. Benefits

The open source tool SSI comes with the benefit of relying on existing and tested functionality. Thus we were able to focus on the interaction with this tool. An open source tool may be helpful for any designer, if she wants to define interaction for an application logic that is still under development or only planned to be developed. The open source tool, of course, must have a comparable functionality. This is valid not only for media annotation tools, but any other software with a UI as well.

One of the main tasks for a multimodal UI designer is to combine *suitable modalities* for a dedicated task in a *suitable way*. This also includes taking care of not cluttering a workspace with too many interaction devices. OI is a good support for the designer during the modality selection process, because it offers an easy way to define interaction. The way of visual programming of OI is easy to understand and the online repository provides a lot of already implemented components. The development of new OI components, like the JuliusOI component that we needed, can be completed without much effort by simply following the instructions in the OI documentation. Moreover, OI supports Java and C++, which are two established programming languages.

Another benefit our approach is that the UI designer can change a UI prototype and fine tune it for an improved interaction without much effort. Finally, the stand-alone multimodal UI can be implemented based on the prototyped interaction. The interaction has then already been tested and evaluated with the prototype, thus saving time and money because the prototyping approach is faster than starting from scratch with the *real* UI code.

C. Drawbacks and Pitfalls

Since OI is embedded in an Eclipse-based environment it is for prototyping purposes only. This means that the UI prototyped with OI can only be used when started from the OI platform. This drawback is mitigated by the fact that the final UI can be implemented rapidly, based on the already prototyped and thus tested and evaluated, multimodal interaction.

While working with OI, we found some (minor) shortcomings of the program itself. For example, let us consider the coupled components in Figure 5. There is no support to copy and paste this group of connected components between OI interaction diagrams. Support for copy and paste would be very useful, as it further eases the reusability of coupled components, allowing to create *patterns of interaction*. Another feature that would ease the use of OI would be the opportunity to see the current input data at a component's input. This comes in handy, if several input-pipes are connected to one component. Besides, OI could provide more hints while the designer creates the interaction. Such hints could be how to clone an output for the multicast component for example. Even if these are not major issues, they cost the designer some time when defining the interaction.

V. Related Work

Previous work of Oviatt et al. [12] points out that building a multimodal UI is not just connecting several modalities together. This leads to common misconceptions of multimodality. Thoughtless coupling of modalities can be rather counterproductive as it means a multimodal UI may be ineffective or disadvantageous. Instead, the modalities have to be combined carefully, considering which modality to choose for which purpose. In many cases where modalities are combined to a multimodal UI, a straightforward addition of modalities is a good way to couple their different expressivenesses and reduce their inherent drawbacks. Such a predictable creation process does not lead to mysterious properties and totally unpredictable effects of the resulting multimodal UI [7]. Reeves et al. [13] define

Copyright (c) IARIA, 2011. ISBN: 978-1-61208-117-5

six main categories of guidelines for multimodal UI design. These are requirement specification, designing multimodal input and output, adaptivity, consistency, feedback and error prevention/handling. Multimodal UIs have two important goals: to achieve an interaction closer to human-human communication and to increase robustness of the interaction by using redundant and complementary information.

Previous work of Bigbee et al. [15] discusses analytical annotation tools with multimodal interfaces. They present dedicated modalities for next-generation tools like ink (pen-gesture) and eye-gaze tracking that reduce workload in time-consuming and frequent tasks of an annotation process. In our work, we considered their results and enhanced an already existing annotation tool that comes with a GUI with new input functionalities. However, we did not use the modality eye-gaze tracking (not really widespread), but the modality ink.

Related work of Reidsma et al. [14] defines design guidelines for focused and efficient annotation tools. Two already existing tools are used as examples to show how properties of specific annotation tasks affect the design of specialized tools. Among other evaluation criteria the Audio/Video interface has to offer an easy to use technique for playing audio and video sections. Another important requirement for an annotation tool is to have an extendable architecture. The latter was the key criteria for our work, as we wanted to improve the UI. We used the ideas of their work as a basis for our work.

VI. Conclusion

When a user works with timeline-based media annotation tool, she usually repeats of a lot of small tasks like adding annotation segments, moving them on the timeline or editing their content. In this work, we presented a case study of prototyping a multimodal UI for a media annotation tool where we focused on the interaction of such small tasks. In particular, we used the rapid prototyping platform OI to define variants of multimodal UIs where we improved interaction with the open-source media annotation tool SSI. We suggest two variants of such a UI: One with GUI, speech input, earcons and mouse gestures; another with GUI, earcons, speech input and a jogwheel. Moreover, we point out that prototyping a multimodal UI for media annotation tools supports the process of UI generation and leads to robust interaction.

Further research, however, is needed to perform user studies about the usability of the proposed interfaces with focus groups and to further improve the robustness of the multimodal UIs for annotation purpose.

Acknowledgment

We would like to gratefully thank Lionel Lawson, Johannes Wagner, Christian Frisson and Hermann Kaindl for their support in the course of this work.

References

[1] 3d space navigator: Last visited on December 10, 2010. http://www.3dconnexion.com/index.php.

[2] Anvil: Last visited on December 10, 2010. http://www.anvil-software.de.

[3] Elan: Last visited on December 10, 2010. http://www.lat-mpi.eu/tools/elan.

[4] Jogwheel: Last visited on December 10, 2010. http://retail.contourdesign.com/?/products/5.

[5] Julius speech recognizer: Last visited on December 10, 2010. http://julius.sourceforge.jp/en_index.php.

[6] Wii mote: Last visited on December 10, 2010. http://www.mynintendo.de/wii-mote-controller.

[7] N. O. Bernsen. *Multimodality Theory*, pages 5–29. Springer, Berlin-Heidelberg, 2008.

[8] F. J. Johannes Wagner, Elisabeth André. Smart sensor integration: A framework for multimodal emotion recognition in real-time. In *Proceedings of the Fourth International Conference Affective Computing and Intelligent Interaction (ACII 2009)*, 2009.

[9] M. K. D. L. Johannes Wagner, Elisabeth André. SSI/ModelUI - A Tool for the Acquisition and Annotation of Human Generated Signals. In *Proceedings of 36. Jahrestagung fur Akustik (DAGA 2010)*, 2010.

[10] V. Lavrenko, S. Feng, and R. Manmatha. Statistical models for automatic video annotation and retrieval. In *Proceedings of the IEEE International Conference on Acoustics, Speech, and Signal Processing (ICASSP '04)*, volume 3, pages iii–1044–7, May 2004.

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

[12] S. Oviatt. Ten myths of multimodal interaction. *Commun. ACM*, 42(11):74–81, 1999.

[13] L. M. Reeves, J. Lai, J. A. Larson, S. Oviatt, T. S. Balaji, S. Buisine, P. Collings, P. Cohen, B. Kraal, J.-C. Martin, M. McTear, T. Raman, K. M. Stanney, H. Su, and Q. Y. Wang. Guidelines for multimodal user interface design. *Commun. ACM*, 47(1):57–59, 2004.

[14] D. Reidsma and D. Hofs. Designing annotation tools based on properties of annotation problems. In *Proceedings of the 5th International Conference on Methods and Techniques in Behavioral Research*. Technology, 2004.

[15] L. H. Tony Bigbee, Dan Loehr. Emerging requirements for multi-modal annotation and analysis tools. In *Proceedings in the 7th European Conference on Speech Communication and Technology (EUROSPEECH-2001)*, pages 1533–1536, Aalborg, Denmark, 2001. ISCA Archive.

Copyright (c) IARIA, 2011. ISBN: 978-1-61208-117-5