A Multimodal Interaction Component for Digital Television

Diogo Pedrosa, José Augusto C. Martins Jr.
Universidade de São Paulo
São Carlos-SP, Brazil
{diogo,joseacm}@icmc.usp.br

Erick L. Melo, Cesar A. C. Teixeira
Universidade Federal de São Carlos
São Carlos-SP, Brazil
{erick_melo,cesar}@dc.ufscar.br

ABSTRACT

In most current digital TV applications the user interaction takes place by pressing keys on a remote control. For simple applications this type of interaction is sufficient — however, as interactive applications become more popular new input devices are demanded. After discussing motivating scenarios, this paper presents an architecture that offers to applications running on a set-top-box the possibility of receiving multimodal data (audio, video, image, ink, accelerometer, text, voice and customized data) from multiple devices (such as mobile phones, PDAs, tablet PCs, notebooks or even desktops). We validated the architecture by implementing a corresponding multimodal interaction component which extends the Brazilian Digital TV middleware, and by building applications which use the component.

Categories and Subject Descriptors

D.2.11 [**Software Engineering**]: Software Architectures—*Data abstraction, Domain-specific architectures*; H.5.2 [**Information Interfaces and Presentation (e.g., HCI)**]: User Interfaces—*Input devices and strategies*

General Terms

Design, Documentation, Human Factors

Keywords

Multimodal Input, Digital TV, Interactive TV

1. INTRODUCTION

Currently, in most existing DTV applications, the user interaction takes place by pressing keys on a remote control – a classic device with arrows, OK, BACK, numbers, and colour keys. This type of interaction device is sufficient for applications that only display additional contents and which allow users to navigate in such contents. However, as interactive applications become more popular, new input devices are demanded – such as microphones, cameras, touch screens, electronic pens and accelerometers.

Examples of applications for iDTV are the electronic program guide (EPG), applications related to a TV show or major events (the World Cup and Elections, for example), games and News [10, 14]. These applications offer few features because they must be easy to use, even if used only with the remote control as an input device. The restriction on the remote control – compared with a full keyboard and a direct manipulation device such as a mouse, that are common on personal computers – limits the usability of iDTV applications [3]. According to Roibás et al. [17], all navigation interfaces that rely on a conventional remote control are far from expressing the interactive potential of an iDTV system. The development of richer applications may require complex interfaces, what would limit its popularization [9].

This work aims at presenting the architecture of a component that allows richer iDTV applications to be developed, offering them the possibility of receiving multimodal data from different devices. After discussing motivating scenarios, this paper presents an architecture that offers to applications running on a set-top-box the possibility of receiving multimodal data (audio, video, image, ink, accelerometer data, text, voice and customized data) from multiple devices (such as mobile phones, PDAs, tablet PCs, notebooks or even desktops). To validate the architecture, we implemented a corresponding multimodal interaction component which extends a Digital TV middleware — we experimented with the Ginga Brazilian middleware — and we built applications which use the component. This work leverages research results presented in our previous work [12] — novel contributions include the possibility to send normal key presses from external devices, the description of a chat application that helps validating the component, and improvements in the related work section.

The remainder of this paper is organized as follows. Section 2 presents the motivation and some usage scenarios for a multimodal component. Section 3 presents related works. In Section 4 we present the Multimodal Interaction Component (MMIC) developed in the context of the Ginga FrEvo and Ginga RAP project[1]. In this Section we detail each of the modules of the architecture, in particular the Event Manager, which plays a central role in the architecture. Section 5 presents two illustrative applications that aimed at allowing an initial validation of MMIC during its development. In Section 6 we discuss future work, and present our final remarks in Section 7.

[1]http://www.ctic.rnp.br/projetos.php?projeto=91

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

2. MOTIVATION AND SCENARIOS

Works such as those reported by César et al. [5] and Cattelan et al. [4] aim at allowing users to create or enrich multimedia contents. Their research exploit the case in which the contents from the television broadcast can be annotated by viewers, and the resulting authorship can be presented on the television, but the authorship itself is performed by applications that run on a portable device such as a tablet PC. The interaction with the tablet PC allows elaborate annotations such as notes made with electronic ink, for instance. In the work of Pimentel et al. [15], another tool for authoring multimedia content is presented, but this time the interaction takes place via the TV remote control. Their authoring tool gives few annotation options comparing to those that run on a tablet PC: they are restricted to the bookmarking of a scene, the selection of a time interval to be skipped when watched later, and the selection of a time interval to be reviewed later in a loop. We observe that there is an opportunity to exploit the strengths of these different types of content production: the use of devices that allow richer input formats and the comfortable environment brought by TV sets.

Teixeira et al. [22] exploited the established habit that users have of loudly commenting television contents and presented a scenario in which users can create ink annotations on top football matches scenes. Something similar can be imagined in the context of a soap opera, films or various other TV shows. In such a context, a generic application allowing electronic ink annotation on video contents would benefit from multimodal input mechanisms.

Several projects in distance education make use of TV. In educational programs, such as the Novo Telecurso[2], widely known in Brazil, it is interesting to enable active participation of students through the use of tools like chat and electronic whiteboard. DTV applications that explore more complex input devices like keyboards and tablets could be used both in cases where students are in their homes and when they are in a classroom sharing the same TV set.

The use of other input devices emerges as one possible solution for applications which require text input. Examples are news applications that allow user comments (a common feature in most news and blog sites), e-commerce applications that demand users to fill out the name and delivery address, and applications that allow viewers to communicate through e-mail, SMS, or chat (as shown in Figure 1) [13]. Examples of applications that allow content search by keyword are presented in the work of Wittenburg et al. [23], Ibrahim and Johansson [6], and Johnston et al. [7]. They support voice utterances as inputs. The latter also supports pen input, including both pointing and handwriting, and a combination of these modalities. Considering applications aimed at allowing users to communicate while watching TV, ChatTV[3] is an example where the phone is indirectly used as an input device for a digital TV application.

Game applications are present in many DTV environments: in this case, the use of input mechanisms that allow a more natural interaction has already been reported. The Wii platform was the first game console to provide an accelerometer control device; manufacturers of other consoles seem to be developing similar devices. It is natural to imagine DTV interactive games that also explore the data inputs of an accelerometer equipped device.

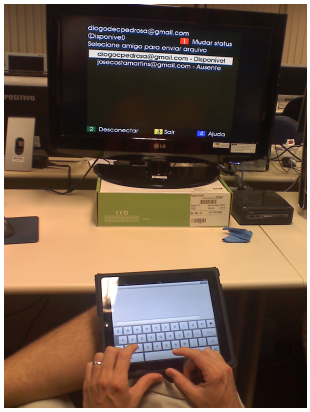

Figure 1: Text input provided via iPad

By offering easy integration with external input devices, a digital TV middleware may help to increase the accessibility of Digital TV applications. Users with special needs who are already using specific devices that help them overcome their limitations can now also use them when interacting with the DTV. It is important that, besides allowing multimodal inputs such as text, audio, video and gesture, the infrastructure also allows different devices to provide simple key presses. For example, it should allow users to change channel using their mobile phones. Thus, applications need not worry about which type of device generated the event, it simply performs the appropriate action. Nakajima [11] presents good arguments to allow existing interactive applications to be operated from mobile devices in ubiquitous computing environments. The attempt to offer to minority groups the same possibilities of using systems through alternative forms of interaction, in order to democratize access to information and entertainment, is being investigated in various studies, including in the area of Interactive Digital TV [3, 16, 17, 21]. Concerned specifically with the need to create a new physical device that allows interaction with iDTV systems, Miranda et al. [9] suggest the use of voice to meet the need of users with physical and visual disabilities, and illiterate users. The authors, however, consider that not all environments allow the use of voice and that a training is required to recognize user's voice.

3. RELATED WORK

In the context of Brazilian digital television system, whose middleware contains an imperative application environment (Ginga-J) and a declarative application environment (Ginga-NCL), Soares et al. [20] present the hierarchical control model used by Ginga-NCL to support multiple devices. The authors argue that video content producers do not wish to have their contents overlaid with additional information and suggest the display of the graphical interface of applications in a secondary device. Another argument given is that overlapping contents normally bother other viewers in the same room. They present a sample application whose graphical interface is shown on a secondary device to offer football shoes for sale while the TV screen presents an animation about a famous football player.

[2]http://www.telecurso2000.org.br
[3]http://www.chattv.com.br

Still in the Brazilian digital television context, a more recent work from Brandão et al. [2] presents LuaTV API, which is part of the draft specification for the NCLua Extended API. It is composed of 4 modules, two of which are related to this work: *HAN*, which offers high-level access to commonly available resources on home networks, and *widget*, aimed at graphical support to applications.

Although the scenarios presented by these works use secondary devices, their main main feature is to explore the additional display that the devices offer. Unlike the architecture presented here, their main objective is not to provide multimodal data for DTV applications. The scenarios presented by Soares et al. [20] and Brandão et al. [2] allow the secondary device to provide input to DTV applications, but they don't help handling the combination of two or more input modes, neither offer specific support to all data types provided by our Multimodal Interaction Component (MMIC), such as ink, voice and raw accelerometer data.

Works that involve the use of multiple devices in the imperative environment of the Brazilian middleware are presented by Silva et al. [18][19]. These works present a Java API that allows applications to use device resources such as cameras to capture video and photo, microphones for audio capturing, telephone networks, displays, and speakers. Despite allowing the input of multiple types of data, the presented API has two major differences compared to what is offered by the component described in this paper: *(i)* The initiative of data transmission comes from different sides. By using a remote or MMIC, the data is sent to the middleware. In Silva et al.'s work is the middleware that fetches data from the device. As a result, while with their Java API the application has to be aware of the presence of the device before it can explicitly request any data from it, using MMIC, the application is constantly ready to receive data from the viewer's device without any additional line of code. *(ii)* The Java API does not support the development of multimodal applications, as it does not support the combination of natural forms of human expression, such as speech, touch, gestures and body movements. In addition to audio, video and image files supported by the work of Silva et al., MMIC supports events that combine electronic ink, accelerometer data, voice, purely textual data, and generic data to the cases not anticipated.

In the context of MHP, the middleware of the European digital television system, Lobato et al. [8] also present an architecture for the development of DTV applications that use inputs other than the ones provided by remote controls. The main difference is that the additional input types are restricted to utterances. Devices must provide audio I/O capabilities and must have speech recognition and synthesis systems implemented.

4. MULTIMODAL INTERACTION COMPONENT ARCHITECTURE

To give applications the opportunity to receive multimodal data from different devices without the need to establish an explicit connection, protocols for automatic configuration and service discovery in IP networks such as Zero Configuration Networking (ZeroConf[4]) and Universal Plug and Play (UPnP[5]) should be used.

Figure 2 illustrates the three main modules of our architecture, which we implemented in a Multimodal Interaction Component (MMIC). The figure also presents the operation of the module: communication modules, extended event manager, and parser. The *Devices* box illustrates that data is sent in an XML format (explained in Section 4.1) to the component. The data is received by one of the *Communication Modules* (Section 4.2) available and transferred to the *Event Manager* (Section 4.3), which uses a *parser* to transform the XML document received into an `IMultimodalInputEvent` object. Then, the *Event Manager* notifies the applications that registered themselves as multimodal input listeners.

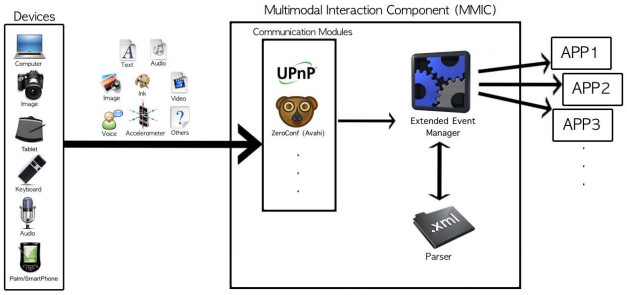

Figure 2: MMIC Architecture

The conversion of the received data to a multimodal event simplifies the application development because it allows the use of the already familiar mechanism of event listeners, and especially because it provides a standardized way to access data. The location of MMIC in the Common Core of Ginga makes it usable both by the imperative environment (Ginga-J) and by the declarative environment (Ginga-NCL), that is, data can be accessed through Java methods, in the case of Xlets, or through tables with pre-specified fields, in the case of Lua applications.

This paper addresses only the case in which a communication module receives input data through computer networks, but the architecture is not limited to it. The case in which input peripherals are connected directly to the decoder via a USB port, for example, may also be considered.

The Multimodal Interaction Component was developed using the virtual machine containing version 0.11.2 of the Ginga Reference Implementation[6].

4.1 Multimodal events

Just as keystroke events are represented by the existing `IInputEvent` interface, so are multimodal events represented by the created `IMultimodalInputEvent` interface. A single multimodal event can transport at the same time all the available types of data, and it is composed of:

- `id` – A string that identifies the event. If it is one of the values specified for the key atribute of the simpleCondition element in the Ginga-NCL standard [1], a regular input event is also dispatched.

[4]http://www.zeroconf.org

[5]http://www.upnp.org

[6]The source code is available at the Ginga Community of the Brazilian Public Software Portal (http://www.softwarepublico.gov.br) and the virtual machine is available at http://www.gingancl.org.br/ferramentas.html.

- `strings` – A vector of strings, used so that ready texts on secondary devices can be sent to applications.
- `ink` – An object of a class that stores electronic ink data as a result of the interaction between the user and a touch-sensitive device or an electronic pen, for example
- `accel` – An object of a class that stores data resulting from the interaction between the user and an accelerometer.
- `binaries` – A vector of objects that store data (name, mimetype and contents) of binary files, such as any format of audio, video and image files.
- `voice` – An object of a class that stores data representing speech, which may be the result of speech recognition systems or may serve as input to speech synthesizers systems.
- `valuesMap` – An object that maps keys to values. It provides greater flexibility because it gives the application the possibility to receive types of data that were not predicted by the API. Another of its possible uses is to carry metadata, such as the width and height of an image present in the event.

A protocol in XML format was defined in order to allow devices to send multimodal data to Ginga. The protocol provides a header containing the device ID, the user ID, and the start/end time of the generated event. The data itself appear in the event body. The XML document excerpt that follows shows an example of a multimodal event.

```
<?xml version="1.0"?>
<multimodal ... id="testEventID">
  <head>
    <device id="DEADBEEF-DEAF-BABA-FEED-BABE00000006"
            model="iPhone 3GS"/>
    <user id="59616261-6461-6261-4E50-472050325033"/>
    <timestamp begin="2010-05-19T09:30:10.5"
               end="2010-05-19T09:30:17.8"/>
  </head>
  <body>
    <value id="width">400</value>
    <value id="height">300</value>
    <text>Beach picture</text>
    <text>What a beautiful place!</text>
    <inkml:ink>
      <inkml:trace>10 0, 9 14, 8 28, 7 42, 6 56, 6 70,...
      </inkml:trace>
      <inkml:trace>130 155, 144 159, 158 160, 170 154,...
      </inkml:trace>
      ...
    </inkml:ink>
    <accel xValue="3" yValue="-2" zValue="1"/>
    <voice>Here goes voicexml data</voice>
    <binary filename="beach.jpg" mimetype="image/jpeg">
            PD94bWwgdmVyc2l...</binary>
  </body>
</multimodal>
```

We chose to use the InkML[7] W3C standard to represent electronic ink data. The developed parser uses code from the inkMLLibcpp[8] library, which was adapted to have the TinyXML[9] library replaced by the Xerces[10] library because the latter was already being used in other parts of Ginga. We also aim to use the VoiceXML[11] standard but the current implementation of the component does not support voice data.

4.2 Communication modules

As mentioned previously, communication between devices and the decoder should be done using protocols for automatic configuration and discovery of services over IP networks, such as UPnP and ZeroConf. The current version of MMIC uses Avahi[12] library and supports ZeroConf. A communication module that supports UPnP is being implemented so that MMIC can offer more flexibility for devices that wish to send data to DTV applications.

To be able to send multimodal data, an application running on a secondary device must search for the ZeroConf service offered by the communication module and connect to a socket using the IP address obtained from the service. Multiple connections can be opened simultaneously.

Communication modules run on separate threads, which are initiated by the Event Manager constructor and stop running only when Ginga is finalized. This ensures that the middleware is always ready to receive multimodal events.

4.3 Event Manager

The Event Manager is the main module of the component. It extends and replaces the Event Manager of the Ginga Reference Implementation in a way that it is now also responsible for receiving multimodal data from communication services, encapsulating them into events, and dispatching the events to interested applications.

Due to the componentized way the Ginga Reference Implementation has been developed, the integration of the extended events manager in Ginga was an easy task to accomplish. We only needed to edit the configuration file of the Ginga Common Core Component Manager, in order to indicate that the new Event Manager, `EnhancedInputManager`, should be used in place of the `InputManager` present in Ginga Common Core System.

One of the responsibilities of this module is to maintain a new list of listeners, specific to multimodal events, allowing interested applications to add/remove a listener to/from the list. Every time input data are generated by a device and passed to the Event Manager via communication module, the manager creates an object that represents a multimodal input event, fills it with the received data, and iterate through the list of listeners to notify them about the occurrence of the event.

5. PRELIMINARY VALIDATION

We developed two applications to initially validate MMIC. Both are composed by two parts: one that runs on a secondary device and transmits the multimodal data and one that runs in the set-top box and uses the received data.

5.1 Multimodal interaction: testing production and consumption

The first part of the first application (client-side) was developed in C and runs on a terminal on any device that is located on the same network as the set-top box and supports

[7]http://www.w3.org/2002/mmi/ink
[8]http://sourceforge.net/apps/trac/inkmltk/wiki/InkMLLib
[9]http://www.grinninglizard.com/tinyxml/
[10]http://xerces.apache.org/xerces-c
[11]http://www.w3.org/TR/voicexml21
[12]http://avahi.org

the ZeroConf protocol. Its function is to send multimodal events to Ginga. As soon as it starts, it searches the ZeroConf service provided by the communication module and starts a loop that sends the contents of an XML code located in the same directory each time the ENTER key is pressed. This pre-defined XML code is very similar to that presented in Section 4.1.

The second part of the first application (set-top-box-side) was developed in C++ as a resident application that is initiated by the main function of Ginga. Its goal is to show all contents of multimodal events received as log messages in a terminal. Multimodal applications must implement the `IMultimodalInputEventListner` interface, register themselves as listeners using the method `addMultimodalInputEventListener` of the `EnhancedInputManager`, and process the events received within the method `userMultimodalEventReceived`. In addition to showing all the event data using log messages, this test application also draws the traces contained in the ink tag of the event, as shown in Figure 3. The traces shown have been copied from the InkML W3C standard example.

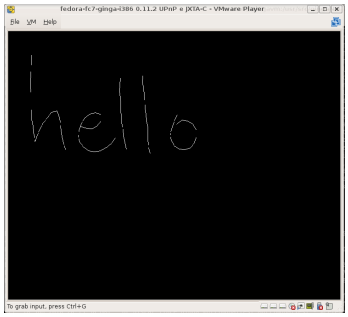

Figure 3: Screenshot of a MMIC test application that shows the traces of multimodal events received

5.2 Multimodal interaction: a Chat application

The second application is interested only in the text strings and binary files of multimodal events. However, differently from the previous application, its goal is not just to test the MMIC component: it is an application for communication which allows users to exchange text messages with each other, and to share files.

The external device (client) part of this application runs on an iPad. It has a graphic interface with only a virtual keyboard, a text field, and a send button, as shown in Figure 4 (left). Each time a button is pressed, it creates an XML message containing the text present in text field and sends it to the middleware.

The *chat* application itself is also a resident application developed in C++. It is composed of several screens and some other extra functionalities. In this presentation, we focus our attention on the chat screen, shown in Figure 4 (right), because it is the one that handles multimodal events. At this screen, when a multimodal event arrives, the application sends all the strings contained in the event as text messages to the friend with whom the user is talking to.

As far as applications are concerned, there are very few lines of code related to receiving multimodal inputs. The first thing an application must do is to register a multimodal input listener in the input manager, as follows:

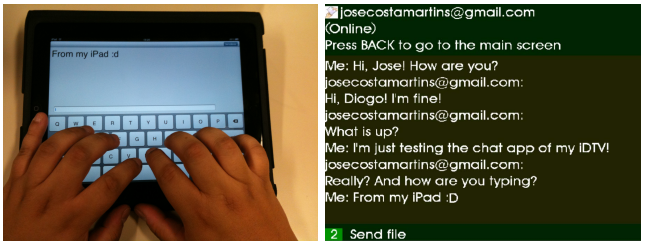

Figure 4: iPad interface for input messages and files to a DTV chat application (left) and chat screen of a DTV resident application (right)

```
static IComponentManager* cm = IComponentManager::
        getCMInstance();
IEnhancedInputManager* eim  =
        ((EnhancedInputManagerCreator*)
        (cm->getObject("InputManager")))();
eim->addMultimodalInputEventListener(this);
```

As the P2PTestApp implements the *IMultimodalInputEventListener* interface, when a multimodal input event arrives, its method *userMultimodalEventReceived* is called. At this moment, the application has access to all data contained in the event and, thus, it can do whatever it needs, as shown in the code extract bellow:

```
bool P2PTest::userMultimodalEventReceived(
        IMultimodalInputEvent* ev) {
    ...
    vector<string>* strs = ev->getStrings();
    for (vector<string>::iterator j = strs->begin();
            j != strs->end(); j++) {
        ...
        temp << "Me: " << *j;
        ...
    }
    ...
}
```

6. DISCUSSION

The current version of our component allows resident applications written in C++ to receive multimodal events. To allow a broader use of the component, APIs that enables Java and Lua applications to receive those events will be implemented.

To further facilitate the use of the component, the protocol will be extended to allow the use of different formats for the same data type. Therefore, to use another ink data representation format, for example, it would be necessary to add an extension to our MMIC component that allowed the interpretation of this new format.

After this step, the implementation of a more robust application may be done. It is intended to extend the interactive document editor prototype presented by Pimentel et al. [15] in order to allow it to accept multimedia data such as those offered by the MMIC component.

Regarding the Event Manager, it is intended to add 3 new functionalities: 1) To extract additional data of the contents received by the protocol and to include these metadata in the multimodal event created, thus providing more complete information to applications. Examples of metadata are height and width of an image. 2) To allow applications to add listeners for events containing specific data types (voice and/or

ink, for example) instead of all multimodal events. 3) To capture all occurred interactions. In other words, the Event Manager is responsible for recording each event, in order to allow automatic authoring of corresponding multimedia documents or knowledge extraction regarding user actions.

7. CONCLUSIONS

The component we have presented broadens the range of options available to Ginga developers, and facilitates the development of innovative applications, since it helps to break one of the main constraints of DTV application development, which is to rely solely on inputs from the remote control. One of the application types that can benefit most from the features introduced in the component is the one that enables end users to create contents. Also, applications dependent on text input will be highly benefited. With regard to the application domain, the contributions are also comprehensive. One aspect that should be reinforced is that the possibility of integrating the decoder with external input devices helps increasing the accessibility of Digital TV applications.

During definition of the architecture and the specification of the protocol, we tried to adopt previously established standards such as InkML, VoiceXML, ZeroConf and UPnP. The Ginga componentized architecture helped to achieve this second goal.

The lack of support for multimodal interaction in DTV applications was the main motivation of this work. Although the Ginga standard does not foresee multimodal interaction, the good performance of the component in more robust implementations can result in a proposal to include this in future versions of the standard.

Acknowledgments. We thank RNP/MCT, CAPES, CNPq, FAPESP and FINEP for their financial support, and Maria da Graça Pimentel for her suggestions and support.

8. REFERENCES

[1] Associação Brasileira de Normas Técnicas. ABNT NBR 15606-2 — Digital terrestrial television – Data coding and transmission specification for digital broadcasting – Part 2, 2007. Corrected version 2 2009.04.17.

[2] R. R. M. Brandão, C. E. C. F. Batista, G. L. Souza Filho, and L. F. G. Soares. Extended features for the Ginga-NCL environment — Introducing the luaTV API. In *ICCCN '2010*, 2010.

[3] A. Carmichael, M. Rice, D. Sloan, and P. Gregor. Digital switchover or digital divide: a prognosis for usable and accessible interactive digital television in the UK. *Univers. Access Inf. Soc.*, 4(4):400–416, 2006.

[4] R. G. Cattelan, C. Teixeira, R. Goularte, and M. D. G. C. Pimentel. Watch-and-comment as a paradigm toward ubiquitous interactive video editing. *ACM TOMCCAP*, 4(4):1–24, 2008.

[5] P. César, D. Bulterman, and A. Jansen. An Architecture for End-User TV Content Enrichment. *Journal of Virtual Reality and Broadcasting*, 3(9), Dec. 2006.

[6] A. Ibrahim and P. Johansson. Multimodal dialogue systems for interactive tvapplications. In *ICMI '2002*, pages 117–222, 2002.

[7] M. Johnston, L. F. D'Haro, M. Levine, and B. Renger. A multimodal interface for access to content in the home. In *ACL '2007*, pages 376–383, 2007.

[8] V. Lobato, G. López, and V. M. Peláez. Mhp interactive applications: Combining visual and speech user interaction modes. In *EuroITV '2009*, pages 10–13, 2009.

[9] L. C. de Miranda, L. S. G. Piccolo, and M. C. C. Baranauskas. Artefatos físicos de interação com a TVDI: desafios e diretrizes para o cenário brasileiro. In *Brazilian IHC 2008*, pages 60–69, 2008.

[10] S. Morris and A. Smith-Chaigneau. *Interactive TV Standards*. Focal Press, 2005.

[11] T. Nakajima. How to reuse exisiting interactive applications in ubiquitous computing environments? In *SAC '06*, pages 1127–1133, 2006.

[12] D. Pedrosa, J. A. C. Martins Jr., E. Melo, and M. G. C. Pimentel. Componente de interação multimodal no Ginga. In *WebMedia '2010*, volume 2, pages 197–202, 2010.

[13] D. Pedrosa, D. A. Vega, M. d. G. C. P. Pimentel, and R. P. M. Fortes. Text input in digital television: a component prototype. In *EuroITV '2010*, pages 75–78, 2010.

[14] C. Peng. *Digital Television Applications*. PhD thesis, Helsinki University of Technology, November 2002.

[15] M. d. G. C. Pimentel, R. G. Cattelan, E. L. Melo, A. F. Prado, and C. A. C. Teixeira. End-user live editing of iTV programmes. *Int. J. Adv. Media Commun.*, 4(1):78–103, 2010.

[16] M. Rice and N. Alm. Designing new interfaces for digital interactive television usable by older adults. *Comput. Entertain.*, 6(1):1–20, 2008.

[17] A. Roibás, R. Sala, S. Ahmad, and M. Radman. Beyond the remote control: Going the extra mile to enhance iTV access via mobile devices & humanizing navigation experience for those with special needs. In *EuroITV '2005*, pages 133–141, 2005.

[18] L. D. N. Silva, C. E. C. F. Batista, L. E. C. Leite, and G. L. Souza Filho. Suporte para desenvolvimento de aplicações multiusuário e multidispositivo para TV digital com Ginga. *T&C Amazônia*, 12:75–84, 2007.

[19] L. D. N. Silva, T. A. Tavares, and G. L. Souza. Desenvolvimento de programas de TVDI explorando as funções inovadoras do GINGA-J. In *WebMedia '2008*, pages 26–29, 2008.

[20] L. F. G. Soares, R. M. Costa, M. F. Moreno, and M. F. Moreno. Multiple exhibition devices in DTV systems. In *ACM MM'2009*, pages 281–290, 2009.

[21] M. V. Springett and R. N. Griffiths. Innovation for inclusive design: an approach to exploring the iDTV design space. In *UXTV '08*, pages 49–58, 2008.

[22] C. A. C. Teixeira, E. L. Melo, R. G. Cattelan, and M. d. G. C. Pimentel. User-media interaction with interactive TV. In *ACM SAC '2009*, pages 1829–1833, 2009.

[23] K. Wittenburg, T. Lanning, D. Schwenke, H. Shubin, and A. Vetro. The prospects for unrestricted speech input for TV content search. In *AVI'06*, pages 352–359, 2006.