Centrum voor Wiskunde en Informatica

REPORTRAPPORT

INS

Information Systems

INformation Systems

INS

Designing interactive ambient multimedia applications: requirements and implementation challenges

Z. Obrenovic, F.-M. Nack, L. Hardman

REPORT INS-E0605 JULY 2006

ISSN 1386-3681

Designing interactive ambient multimedia applications: requirements and implementation challenges

ABSTRACT
Ambient intelligence opens new possibilities for interactive multimedia, leading towards applications where the selection, generation and playback of multimedia content can be directed and influenced by multiple users in an ambient sensor network. In this paper, we derive the basic requirements for a flexible infrastructure that can support the integration of multimedia and ambient intelligence, and enable rapid tailoring of interactive multimedia applications. We describe our implementation of the proposed infrastructure, and demonstrate its functionality through several prototype applications.

1998 ACM Computing Classification System: H.5.1; H.1.2
Keywords and Phrases: Adaptive multimedia; ambient intelligence; software platform
Note: This work was carried out under project INS2 - Passepartout

Designing Interactive Ambient Multimedia Applications: Requirements and Implementation Challenges

Zeljko Obrenovic, Frank Nack, Lynda Hardman
CWI, P.O. Box 94079, 1090 GB, Amsterdam, The Netherlands
Firstname.Lastname@cwi.nl

ABSTRACT

Ambient intelligence opens new possibilities for interactive multimedia, leading towards applications where the selection, generation and playback of multimedia content can be directed and influenced by multiple users in an ambient sensor network. In this paper, we derive the basic requirements for a flexible infrastructure that can support the integration of multimedia and ambient intelligence, and enable rapid tailoring of interactive multimedia applications. We describe our implementation of the proposed infrastructure, and demonstrate its functionality through several prototype applications.

Categories and Subject Descriptors

H.5.1 [**Multimedia Information Systems**], H.1.2 [**User/Machine Systems**], D.2.11 [**Software Architectures**]

General Terms

Design, Human Factors.

Keywords

Adaptive multimedia, ambient intelligence, software platform.

1. INTRODUCTION

The vision of ambient intelligence is to create environments that are sensitive to the presence of people and responsive to their needs [1]. Ambient intelligence is an umbrella term for a set of technologies that lead us toward this goal, going beyond today's common mode of interaction where a single user operates a mouse and keyboard in front of a single display controlled by a single machine. Ambient intelligence opens new possibilities for interactive multimedia. Multimedia is not just "powerpoint on the screen", but it is also concerned with different input and output modalities, often simultaneously used by several users to communicate multimedia information in a shared environment [4]. Embedding multimedia in sensor-enhanced ambient environments can enhance the range of interaction techniques even further.

However, designing applications that can exploit the potentials of the marriage between multimedia and ambient intelligence is a challenging task. There are many open issues, and to date there is no clear model of how this integration could be done. One of the main problems is the huge space of possible solutions, with different devices, sensors, usage scenarios, and personalization factors. It is practically impossible to hardcode solutions for each of the individual combinations. If we want to develop ambient multimedia applications efficiently, we need to rethink how they are specified and implemented.

In this paper, we describe the requirements for a generic platform for the integration of multimedia and ambient intelligence. We also discuss the main implementation challenges, based on our implementation of this platform. This platform is used as a core part of a broader project that explores the design of new interactive multimedia applications, where the selection, generation and playback of multimedia content can be directed and influenced by an ambient sensor network. Our goal is to enable multimedia developers to experiment with interaction techniques in sensor-enhanced ambient environments, while allowing for as much reuse of their previous work as possible.

We first describe a number of existing solutions, and discus their common issues. We then outline basic requirements for a flexible ambient multimedia platform. The main part of the paper describes a platform we have developed based on these requirements. We illustrate the flexibility of the platform through a number of prototypes of interactive multimedia applications, which explore the usage of camera-based techniques, speech, and biometric data to control and adapt multimedia content and environment.

2. EXISTING SOLUTIONS

Combining multimedia with ambient intelligence has already been explored by several research groups, in particular in gaming and artistic settings. Our goal is not to create a comprehensive list of existing solutions, but to illustrate the range of scenarios that even simple forms of ambient intelligence make possible. The common issues extracted from theses scenarios form the basis for deriving requirements for the flexible infrastructure that can support more efficient integration of multimedia with ambient intelligence.

2.1 Ambient Interaction Applications

Enriching working spaces, such as offices or meeting rooms, with ambient sensors has been explored with several research groups. For example, Stanford's "Interactive Workspaces" project explores new ways for people to work together in technology-rich environments. Interactive Workspaces combine large displays with smaller interaction devices [13].

Artistic performances and multimedia events have exploited a number of new sensing and interaction techniques. For example, the "Multisensory Integrated Expressive Environments" framework has been used in performing arts, such as interactive dance, music, or video installations [6]. In these performances, sensors captured player movements, face expressions, or gestures, and transformed them into parameters used to automatically control scene elements, such as music.

Enriching public physical spaces with sensors can extend the experience of visitors. For example, the eXspot system uses small radio-frequency identifier (RFID) reader packages mounted on museum exhibits, together with a radio-frequency (RF) tag carried by visitors on a card or necklace, connected in a wireless network, and used with a registration kiosk to dynamically generate Web pages [11]. The Situating Hybrid Assemblies in Public Environments (SHAPE) project has also explored how emerging ubiquitous technologies and ambient intelligence can support museum visiting experiences. The project has investigated how to support visitors to manipulate physical and digital material, and to collaborate with other visitors [2].

Games are another area where multimedia and ambient technologies begin to merge. The aim is to unchain the game players from the console using alternative interaction modalities. Systems like Human Pacman [3], where users have to run in order to control avatars, extend the gaming experience into the real world — a living room, streets, or the remote wilderness. In such systems, sensors capture information about players' current context, such as location, actions, and emotional state, and use this data to adapt the game flow. By supporting learning through physical role play, these games also have an educational potential. In their platform for physical play, Wakkary et al. explored the more general design issues of sensing, displaying, user modeling, and using interaction models in designing a system based on a game structure [24]. Their system supports game structures such as word puzzles, levels, body states, goals and game skills, connecting them with body movements and positions, through real-time motion capture.

Interactive music systems have also been used with ambient sensing techniques. For example, Healey et al. developed a system that aids in music selection by incorporating physiological variables that might indicate the user’s present mood. They developed a wearable computer that perceives and responds to the wearer’s affective state, capturing patterns from many kinds of user behavior [10]. The Viktoria Institute's Future Applications Lab, and the Interactive Institute's PLAY Studi from Sweden developed the Sonic City, a form of interactive music instrument using the city ambient as an interface. Sonic City enables users to create a real-time personalized music by walking through and interacting with urban environments [16].

2.2 Common issues

While the described systems have been used in different domains, each of them addresses several common issues, including:

- sensing and affecting the user and environment,
- modeling the context, by analyzing data available from the sensors,
- higher-level reasoning and integrated interpretation of sensory and contextual data, and
- integration mechanisms, which enables efficient communication between devices, modeling modules, and reasoning modules.

Sensing and Affecting the User and Environment

One of the most important elements of an ambient intelligence system is maintaining the communication with sensors, and, if necessary, performing operations such as sensor calibration, high-pass filtering and mean value calculations, or applying more sophisticated signal processing. In most of the systems, sensor processing modules are an integral part of the application. Hardware includes RFID sensors, cameras, microphones, temperature sensors, accelerometers, as well as bio-sensors, such as heart rate or galvanic skin response (GSR) sensors. Sensor processing modules vary from simple, providing digitally converted data without applying complex processing, to very complex ones that use sophisticated signal processing algorithms to derive values, such as a trajectory of the user or the user state. Devices often include simple actuators such as lights, fans or vibrators, which can be used to affect the user. Sensors also communicate through both wired, (e.g. RS-232, USB, or Ethernet) or wireless interfaces (e.g. Bluetooth).

Modeling Users, Devices and the Environment

In most cases, ambient systems include modules that track the context in which they are used, keeping relevant models about users, devices, and the environment. The techniques to create and update these models vary depending on the environment in which the system operates. Systems used in public spaces, focus on modeling environment and devices, while systems used in more personal environments include more detailed user models. The core activity of these modules is to track the changes in the environment, and store relevant data about ongoing interactions. For example, Perring et al. [19] developed a system where mobile devices could be used with other devices in the room. To enable seamless integration between the services of each device, the system has to maintain service description that a device provides, and track the temporary relations between devices.

Reasoning

In addition to interpreting data from individual sensors, ambient intelligence solutions often include modules that use data from all the sensors, together with information from other sources to facilitate higher-level reasoning. These reasoning modules usually form the "intelligent" part of ambient intelligence. Techniques used vary from simple analysis to more complex artifical intelligence techniques. For example, Wakkary et al. use a rule-based reasoning engine to interpret the sensor data, identifying the level of body state completion, which was then used to control the narrative flow of the experience [24]. In some cases, solutions did not require complex reasoning. For example, some approaches focused more on providing simple metaphors that can be easily manipulated by developers, rather than using complex modules that do this automatically. Kameas et al, for example, illustrated that even users can become “creators” of ambient applications if they are given simple metaphors and usable (but not necessarily intelligent) manipulation tools [14].

Integration Mechanism

In ambient intelligence settings, it is important to provide efficient communication mechanisms to connect all the components. Most existing systems have solved the integration problem by directly connecting sensor data to modules that use it. Some ambient intelligence systems have tried to provide more generic and reusable integration mechanism that can enable plug-in integration of components. For example, the EventHype system, used in the Interactive Workspaces project, addresses the issue of integration through the design of a centralized event exchange system for workspace devices [13].

3. REQUIREMENTS FOR AN AMBIENT MULTIMEDIA PLATFORM

The bottleneck for more efficient development of interactive ambient multimedia environments is integration. In most existing systems, integration is hardcoded for a particular scenario, making individual parts of the system hard to reuse. Our aim is to establish a flexible integration platform that facilitates the needs of the common process modules. In this section, we identify basic requirements for a flexible platform for rapid tailoring of novel interactive multimedia applications in ambient intelligence environments. Our aim is not to create a comprehensive list of requirements, but to identify those that allow us to go beyond simply connecting sensors in an *ad-hoc* manner.

3.1 Supporting a Wide Range of Scenarios

Scenarios range from single user environments in an office or home environment, to multi-user interaction in outdoor and urban environments. Each situation requires different sensors, user profiles, device profiles and environmental context. Components are typically not easily extensible or reusable for scenarios other than the one for which they were designed. *The platform has to support the varied needs of applications in a wide range of scenarios, without restricting unforeseen uses of sensors or other components.*

3.2 Supporting Different Sensing Modules

Input can be obtained from a wide range of sensing modules, connected over different communication interfaces. Some sensing modules send low-level data frequently, others send high-level process data infrequently. Sensing modules may also use different communication models. For example, some sensing modules work in push mode, automatically sending data, others work in client-server mode, sending data upon request. The infrastructure should support both models, and allow flexible communication of data among modules. In most existing solutions, sensing modules are an integral part, crafted for a particular scenario. This inhibits reuse of existing sensor modules in other applications. *The platform should enable the integration of sensing modules using different interfaces and communication models. Introduction of new sensing modules should require only minor changes in other components.*

3.3 Modeling and Reasoning Support

In most cases, the environment has to be aware of many parameters, such as the device used, users present, and/or their current physiological state. The range and complexity of approaches to modeling and analyzing these parameters varies significantly. Reasoning engines may also need access to data from other sources, such as user and device profiles stored in external sources (such as, files, databases, or web repositories). *The platform should enable the integration of modeling and reasoning mechanism of differing complexities. It also has to be able to import data from external sources, in addition to sensor data.*

3.4 Dynamic Integration

Configuration of connections among sensing modules and other components is a basic requirement for any ambient system. Current systems and tools, however, require the developer to specify these connections manually. This is sufficient where the system is static and connections are not overly complex. Where there are many sensing modules inputting data to different processing modules, and where the configuration is likely to change over time, for example when a user becomes more expert in using the system, then support for specifying connections amongst modules is required. Optimal interaction configuration can also differ from user to user. *The platform has to be able to dynamically reconnect sensing modules with modules depending on the current state of the system.*

4. IMPLEMENTING AN AMBIENT MULTIMEDIA PLATFORM

Based on the requirements, we have implemented a generic platform that enables multimedia developers to experiment with ambient intelligence interaction techniques. In this section, we give an overview of the platform, and describe its implementation details.

4.1 System Overview

Our platform supports building interactive multimedia applications using a component-based approach [15]. Sensing, modeling, reasoning, and application modules are treated as independent components that can run on different machines in a distributed environment. *Support for a wide range of scenarios* is achieved through providing different connections among these components. We use a loosely-coupled model to connect the components through a centralized communicator system (Figure 1). The communicator decouples the components both spatially and temporally. This indirect communication provides extensibility by allowing usage of sensing and other modules in different configurations. Furthermore, it allows modules to intercept and translate all communication events.

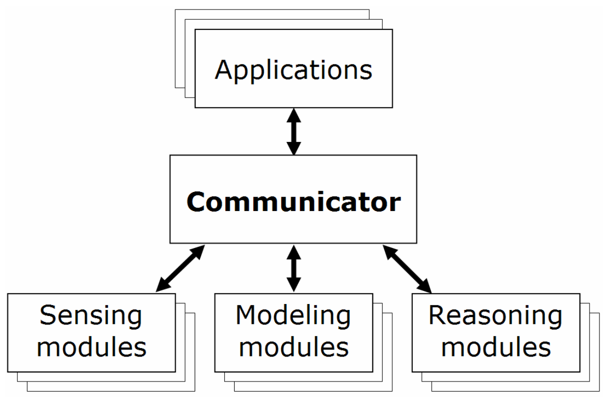

Figure 1. Connecting components through centralized communicator system.

Components in our system communicate by exchanging events via the communicator. We have provided several interfaces and communication models to *support integration of different sensing, modeling and reasoning modules.* For greater flexibility, we abstracted the events send by components into the concept of a *variable*. One variable can be updated or derived by different events. For example, a variable that represents a number of people in the room can be updated by a RFID sensing module or by a camera sensing module. Applications can register for this variable, not being concerned with the details of the sensing module used to instantiate it. Modules could also derive new variables by processing existing variables.

One of the key differences between our communicator and other event exchange systems is the *variable derivation* part. This enables instantiation of new variables from variables updated by sensing modules and applications. New variables are derived by using a set of *variable transformations*. Therefore, applications do not have to directly use low-level sensor variables, but can add transformations that adapt these variables for their needs. In this way, the communicator not only exchanges events, but also provides a means for the creation of higher-level semantics from low-level events. An important goal of our transformation framework is to enable variable transformations in several steps. For example, one transformation can start from low-level sensor data, and transform them into intermediate variables. These intermediate variables can then be used by other transformations that transform them into application specific variables. Usage of intermediate variables can also simplify integration of new sensing modules, as we can reuse transformation from intermediate to application specific variables, and add only transformations of sensor-specific variables to intermediate variables.

Figure 2 illustrates variable derivation using a camera-based playback control. A face detector sends coordinates of detected faces. The communicator then runs variable transformations that derive several other variables: one describing the number of detected faces, and, for each face, a variable containing the coordinates for that face. The number of faces directly correlates to the number of people in the room, while the coordinates of the faces facilitate the derivation of the average height of faces, what roughly correlates to the distance of people from the camera. The derived distance variable can then be used to change the variables used to control presentation of the content, for example the volume intensity or font size. Although we could directly obtain playback controls from a sensor variable, deriving variables in several layers is a more flexible approach. For example, playback control could also be derived from a speech recognizer that updates the same variable. In this way, the player can be controlled, through speech or movement, but it does not have to be reconfigured, as all the sensing modules update the same variables, but using different transformations.

The communicator also provides interfaces for dynamically loading and changing the custom defined transformations, providing support for *dynamic integration of components*. Our platform, therefore, provides a dynamic middleware that applications can adapt to support different interaction scenarios. This is also the main difference between our system and existing component systems, which usually require the developer to specify connections among components manually [5].

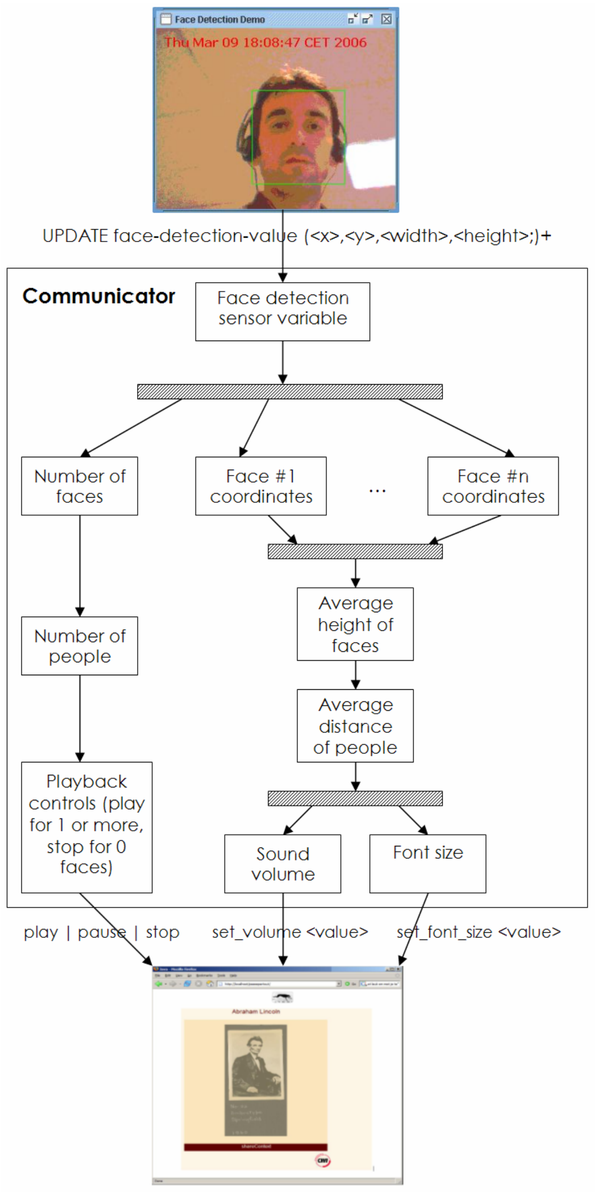

Figure 2. Derivation of variables in a camera based control of multimedia playback.

4.2 Implementation Details

The communicator forms the core of our platform. Its main task is to enable the flexible connection of all components with applications. It is implemented in Java, using standard Java libraries. The communicator and its external modules can be downloaded from our Web site[1].

The communicator keeps a pool of variables, which can be updated by sensing modules or applications, and distributed to clients upon request or automatically after a state change. Applications and sensing modules can update or read variables using simple UDP and TCP communication interfaces, or they can

[1] http://amico.sourceforge.net/

use higher-level communication modules realized on top of the communicator, e.g. generic communication adapters such as the XML-RPC adapter. For example, a speech recognizer can send an event with a recognized speech phrase by sending a UDP package with content "`UPDATE speech-phrase-variable value`". The communicator also provides a user interface that enables users to see which variables are currently used and to associate a privacy control tag with them.

Components receive values of variables in several communication models, including:

- *Push model*: a component registers for specific variables, providing a command template where parts of the command are replaced with the concrete variable values. When a trigger value is changed, these templates are evaluated and returned.
- *Pull model*: clients can connect to our platform, and request a list of values. The component could also send a template, which will be populated with variables present in the communicator and returned.
- *Direct connection*: sensing modules communicate with the clients avoiding the processing part of our platform. This communication model enables the efficient communication of large streams of data to clients.

The transformations used to derive new variables are described in the Extensible Stylesheet Transformation (XSLT) language. XSLT is a standardized, commonly used transformation language, familiar to many developers and supported by many existing tools. Transformations also allow sensing modules, as well as applications, to be simpler and reusable in different situations, as a part of processing can be present in the transformations. By merging the transformation and routing logic in the communicator, we can keep client libraries simple and easy to port as well as reduce the administration needed on each component. In addition, developers could use other components to derive variables, integrating them with the communicator through our interfaces.

The communicator is able to load variables dynamically from files or Web pages. In this way, the communicator can load variables from user or device profiles, making them accessible to other modules, for example, to a reasoning engine that has to derive a user state based on threshold values from a user profile. We use a simple XML format for describing variables, and the communicator can load these variables from a given URL. This URL can reference a local file, or a dynamic Web page that generates the file from a database.

Users can share a communicator instance on some common machine, but they can also run their own communicator. Communicators could also be interconnected, so that one or more communicators could receive events derived by some other communicator.

The communicator also includes components that can provide help in debugging and rapid prototyping of the applications. Scenarios are often based on still undeveloped components. A partial implementation of the system cannot show the full potential, while a complete implementation might be impractical [7]. Therefore, we have also developed tools that can enable developers to model some undeveloped or unstable component by simulating them manually or automatically.

5. APPLICATION ADAPTERS

The communicator provides flexible communication mechanisms that can be directly used by different applications and sensing modules. However, in order to ease the integration of existing and new multimedia applications, we have developed additional components that further simplify this process. We will firstly introduce some of the generic application adapters, and then illustrate adapters that can be used to integrate Web-based multimedia applications.

5.1 Generic Applications Modules

In order to improve the basic functionality of the communicator with respect to structure and applicability, we have developed several modules that provide extended interfaces. Those interfaces improve the interoperability of the communicator with other components and existing applications. The first set of these modules consists of generic communication interfaces, including:

- a XML-RPC interface for accessing communicator variables, mapping the variables to XML-RPC function calls, and updating the variables with the results of the calls.
- an OpenSound Control (OSC) interface, which is similar to the XML-RPC interface only that it uses the OSC protocol.
- a sensing adapters package, such as RS-232 or USB adapters.

We have used these generic adapters to integrate existing multimedia applications and sensing modules. For example, we use the OSC interface to control the VeeJay system [23], to create visual performances and interactive video installations that manipulate video in realtime using different devices, such as the cushions (Section 6.4).

We are also working on creating some application specific adapters for other multimedia environments such as Macromedia Director, by using its Xtras C++ extension mechanism [17].

5.2 Web Applications Modules

The World Wide Web is one of today's main channels for distribution of online multimedia content. Broadcast companies already distribute their content over the Internet, primarily using a Web interface. Also, there are many adaptive multimedia applications which are developed as Web applications and services. For example, some automatic presentation generation systems produce Synchronized Multimedia Interchange Language (SMIL) content based on parameters they receive with calls [22]. Therefore, large amount of multimedia content from different Web servers is already available for consumption and manipulation. On the other hand, Web browsers offer a flexible architecture for playing multimedia content, by supporting many standards and formats. Although the Web infrastructure does not provide a direct means for integration of novel interaction modalities, mechanism and interfaces that exist there can be used to extend basic functionalities of Web applications. For example, the functionality of Web browsers can be extended with mechanisms such as scripting and applets. On the other hand, multimedia applications, such as RealPlayer, extend their functionality by embedding Web browsers. Scripting in Web servers and clients also enables more flexible rapid prototyping. We have, therefore, used these Web mechanisms to integrate existing web-based multimedia applications with our system to enable interaction with multimedia content through new sensing devices. To make this possible, we have developed two components, namely:

- A template-based content changer that facilitates flexible multimedia content selection;
- A scripting applet that controls the playback and interaction with multimedia content by mapping variables from the communicator to calls of script functions inside a Web browser.

Template-Based Content Changer

Currently, online multimedia content providers usually offer their content in a fixed Web HTML interface. These interfaces, however, limit the flexibility of content usage.

Instead of providing fixed HTML pages with embedded links to multimedia content, we have developed a more flexible content-changer system that can enable playback of multimedia on the Web. The basic idea is to enable the dynamic building and control of player settings based on the values present in the communicator. Our content-changer component loads two lists of links. The first list contains links to multimedia content, such as a link to a movie, or a link to a VRML scene. A link could refer to static content on some server, or to a dynamic Web page that generates multimedia content on-the-fly. Every link from this list is also associated with a list of players that can be used to open the linked content, optionally with the target window in which it should be played. The second list contains links to dynamic Web pages that embed players for each of the supported multimedia content types. Each of these pages requires a link to content that it will play. We have developed several of these dynamic pages, using PHP, JSP and ASP technologies, with embedded players for different content types (for example, RealPlayer, Windows Media Player, VRML players). Our player pages also embed a scripting applet, described in the following section, that enables the control of audio-visual playback by using a sensing module, such as a speech recognizer or a camera-based face detector. When a user selects the content from the list, the content changer automatically selects an appropriate player Web page, and generates the link to the selected content.

Links to content and player pages are described in the form of *templates* that can be populated with variables present in our communicator. This means that parts of URLs can be replaced with actual variables from the communicator, enabling run-time selection of content and adaptation of players. For example, parameters to dynamic Web pages can be replaced with actual values, such as user state, or preferred colors. Active player Web pages can receive parameters about the dimensions of the screen, and adapt the dimension of the player window.

The content changer is implemented as a Java application that communicates with the communicator in several ways (Figure 3). Firstly, the content changer registers for the commands, such as reload of current content, or selection of previous and next content item from the list, so it can be controlled with any component that updates this variable. The content changer also updates variables based on the currently selected content and player, and can reload the content when some of the variables change. For example, dynamically generated content can be sensitive to changes in the user state, and can be automatically adapted with the described technology. Finally, the content changer can update the variables with one or more URLs of currently selected content. This variable can be used with any application that wants to present this content. A simple *content change applet* that registers for this variable, and reloads the content in the Web browser each time new content is selected, facilitates automation of the process. The content changer is configured with links to player and content templates, which can be files, or an active Web page that generates these lists based on user queries for multimedia content.

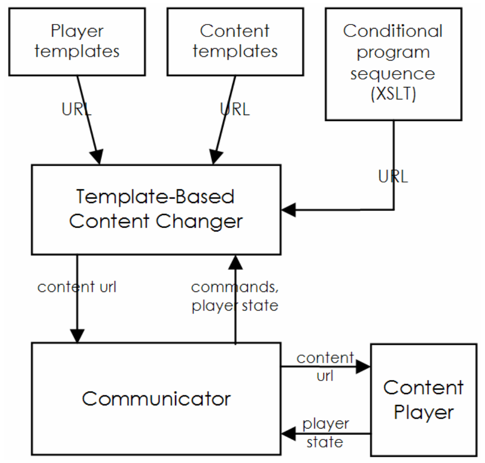

Figure 3. Template-based content changer and its interaction with the communicator.

Scripting Applet

In order to control multimedia content in Web browsers using interaction modalities not present in the user interface of the browser, we have developed a flexible mechanism for mapping of communicator variables to Web browser controls. Browsers usually provide several ways in which they can be extended. For example, it is possible to develop new plug-in modules, or toolbar extension, and in that way affect the content loaded in the Web browser. However, this approach requires development of components for each browser and platform, and requires that users install them in order to be used. Instead of that, we provide a simpler, but equally functional way of controlling the content loaded in the Web browser using *a scripting applet*. The scripting applet transforms notifications sent from the communicator into calls of scripting functions. The scripting applet is completely reconfigurable, and developers can use it to control any scriptable element in the HTML page.

```
<script>
   function p_start() { document.movieclip.DoPlay()    }
   function p_pause() { document.movieclip.DoPause()  }
   function p_stop() { document.movieclip.DoStop()    }
   function p_set_volume(volume) {
      document.movieclip.SetVolume(volume)
   }
   function p_set_bgColor(color) {
      document.bgColor=color
   }
</script>
......
<applet code="ScriptingApplet.class"
               width="0" width="0" MAYSCRIPT>
  <param name="port" value="3320">
  <param name="command-template-1"
     value="ADD TEMPLATE DIFF player-action
              p_<%=player-action%>">
  <param name="command-template-2"
     value="ADD TEMPLATE DIFF bg-color
              p_set_bgColor <%=bg-color%>">
</applet>
```

Figure 4. Parameters for the ScriptingApplet used to control the playback of the RealPlayer plugin.

Figure 4 illustrate parameters of the scripting applet used to control the playback of the RealPlayer plugin. The applet registers for change of variables named 'player-action' ('start', 'pause', 'stop', or 'set_volume <value>'), and variable 'bg-color'. Whenever these variables change, the applet receives a string such as 'p_start', and it calls the script function with the same name.

We have used our applet to control interaction in Mozilla, Firefox, and Internet Explorer Web browsers, including:

- Controlling standard elements of HTML page through DOM interfaces, changing content, such as text in some paragraph, style elements, such as background color, or the layout, like moving image controls by camera.
- Controlling playback in multimedia movie and music plugins, such as RealPlayer, Windows Media Player, and QuickTime players,
- Accessing and manipulating VRML scenes, using VRML External Authoring Interface (EAI) scripting API exposed by VRML players.

6. PROTOTYPE APPLICATIONS

The proposed framework has been applied to and tested in different applications, several of which we describe in this section. The adaptable communication model and tools that we developed enable us to combine many components by simply reconfiguring the system. We start with straight-forward examples, which illustrate the use of off-the-shelf sensor modules for building interactive multimedia applications. We then describe a more complex example where the communicator is used in a complex multi-user and multi-sensory environment.

6.1 Camera-Based Playback Control

In this prototype, we have evaluated how camera based face detection, and other real-time computer vision techniques, can be used to control the playback of multimedia content. The basic scenario is that the system starts a playback when there is at least one person looking at the screen, it pauses playback when there is no one there, and continues again when someone else appears.

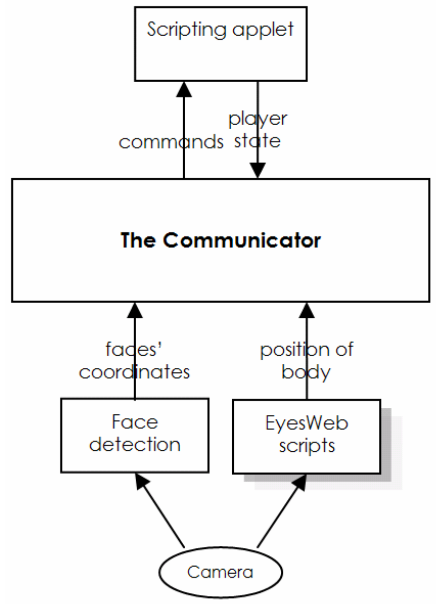

Figure 5. Basic configuration of components, and data flow in our camera-based playback control example.

For the face detection component we have used OpenCV, an open source computer vision library [18]. The data from the face detector, i.e. coordinates of detected faces, are transformed by the communicator transformations into variables that control the ScriptingApplet to influence the playback of multimedia content in players such as RealPlayer and Windows Media Player. The derivation of variables in the camera-based control of multimedia playback is described earlier in Figure 2. The basic configuration and data flow are described in Figure 5.

In a similar way we also enabled that parts of the screen, such as background color, can be changed based on the number of the detected faces. The face-detecting sensing module can also recognize other object profiles, such as frontal face, profile, upper body, lower body, whole body, and it can be configured with the minimum size of an object. Minimal size of an object is an important parameter if we want to detect people or objects within a limited distance.

We also used other external sensing modules that can process camera input, such as EyesWeb [8]. We have adapted several examples earlier developed within the EyesWeb platform. As we use simple network protocols for updating the variables in the communicator, in most cases the modification consisted of converting detected variables to a string, concatenating it with a prefix to meet the requirements of our protocol, and using EyesWeb's NetSender component to send this new string value to our communicator. For instance, we used an EyesWeb example that tracks motion of human body, with our scripting applet to control an animated character in a simple HTML game.

Described ways of interaction can be particularly useful in active public interfaces, which already have started to use techniques to visually sense humans [25].

6.2 Speech-Based Content Selection and Playback

In this application we have used a speech recognizer (Sphinx-4, an open source speech recognition platform [21]) to control the content-changer component and parameters of multimedia players using our scripting applet (Figure 6).

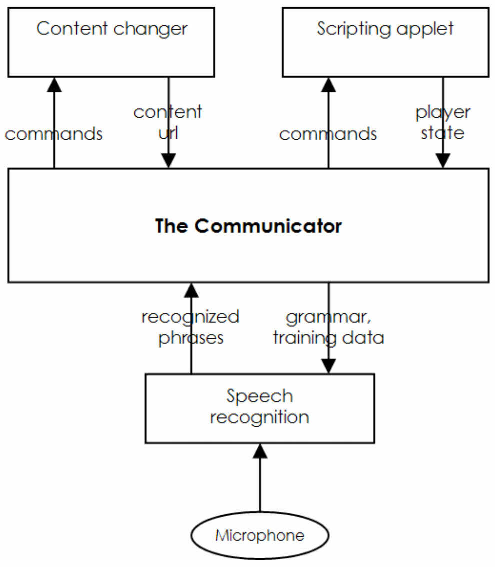

Figure 6. Basic configuration of components, and data flow in our TTS news reader with speech control example.

When the speech engine recognizes a phrase, it updates the registered variable in the communicator. This event also triggers a transformation that maps the speech variable into command variables for the scripting applet and/or content changer (for example phrases 'reload', 'previous' and 'next' are used to control the content changer, while phrases 'play', 'stop', 'quiet' and 'loader' are used to control the playback and volume in multimedia players such as RealPlayer).

We also enabled the dynamic reconfiguration of the speech recognizer with data from the communicator. To increase the overall performance of the player adaptation, the speech recognizer is configured to recognize a limited set of phrases. The speech recognizer receives the grammar of phrases that can be recognized, and then reconfigures to recognize these phrases. The speech recognizer can also receive a link to files with training data of the user loaded from the user profile.

6.3 TTS News Reader with Speech Control

This example combines an RSS reader, with a text-to-speech (TTS) engine, and the speech recognizer (Figure 7). The RSS reader component is based on ROME open-source syndication library [20], while the TTS engine uses FreeTTS open source library [9]. We used the same speech recognizer as in the previous example.

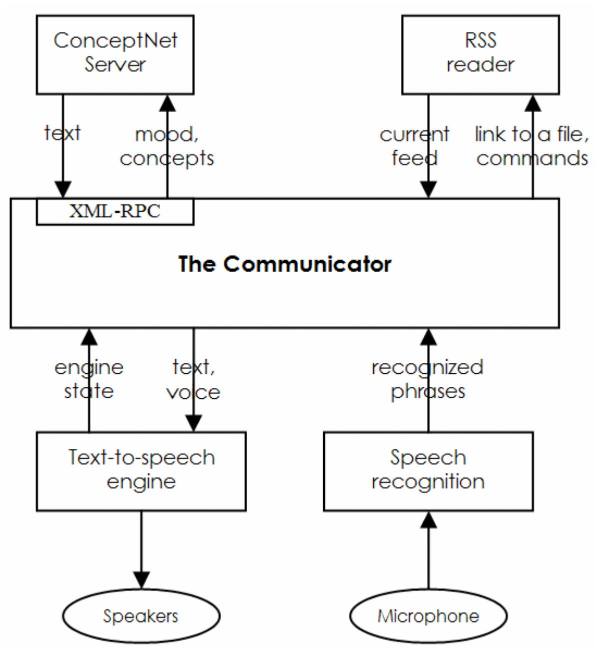

Figure 7. Basic configuration of components, and data flow in our TTS news reader with speech control example.

The RSS reader is configured so that it loads RSS feed in the communicator variable as soon as the TTS engine is ready (after it has started, or when it has finished reading previous text). The TTS engine is configured to read the text from the same variable that the RSS reader updates. In this way, both components are configured to trigger each other. We have also used the ConceptNet service to extract basic concepts and to guess the mood of RSS feeds [12]. ConceptNet is a freely available commonsense knowledgebase and natural-language-processing toolkit, which can be run as XML-RPC server. Data derived in the ConceptNet server is put in several communicator variables, and used by other modules, for example by our search engine pages which are triggered by these values, so that a user can further explore context of the news using engines such as Google or Wikipedia. Finally, the RSS reader can receive commands from the speech recognizer, so that a user can request news again, skip to news, and jump to some news item.

6.4 MOVE.ME: Content and Environment Adaptation Based on Biometric Data

The *move.me* prototype is ongoing research in an interactive environment in which several users can manipulate the audio-visual content presented on screens through interaction with sensor-enhanced objects, in our case cushions. The sensors embedded in these objects include pressure, galvanic skin response (GSR), movement patterns, and presence (RFID reader). Moreover, each object can be equipped with output devices, such as fans, vibrators or light-emissive fibers. The cushions within the environment are provided with different sets of sensors and output devices. A simple pressure-based cushion looks like the one portrayed in Figure 8. In order to support interaction with these types of devices in a home entertainment environment, we have been developing a system that detects the user's current excitement state and compares the retrieved sensor data with established excitement thresholds in the user model. Based on the outcome of this comparison the system adapts the presentation of the content the user is currently involved in or it alters the environment if required. For example, the system might change presentation attributes such as font size or volume due to decreasing excitement values.

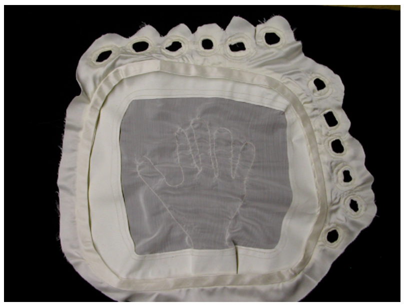

Figure 8: A pressure-sensor cushion, where the pressure pad is located in the center grey area.

To support the interaction with such sensor-enhanced devices we have developed several components on top of the communicator, including:

- *The Device manager*, is a complex sensing module which instantiates the device drivers for one or more cushions and runs the interpretation engine for each of the sensors. This module updates the variables in the communicator based on the values interpreted from the cushion's sensors, and receives the variables transferring them into the sensors' commands, such as activating the cushion's light or vibrator to communicate with the user.
- *The Context manager* updates the current status of users and devices present in the environment. For example it tracks

the spatial and interaction relation between cushions and users, based on RFID tags and change of sensor data.

- *The Adaptation engine* collects information from the context and device managers to draw inferences on the current state changes of a user. Based on established thresholds it determines the appropriate adaptation method and provides the communicator with instructions about the source to be adapted and the means of adaptation.

Each of the managers generates data structures that might be potentially relevant for other system components. The basic configuration and the data flow is described in Figure 9. This basic configuration has been used in several settings, with additional modules. For example, in one of the settings we use data from these managers to control the dynamic of interactive video in the VeeJay system [23].

The communicator provides the flexibility and easy reconfigurability to facilitate such a data exchange. It proved useful during the preparation phase of several workshops where we evaluate the usability of sensor-sets for a cushion in different environment settings. We apply the same fast development cycles in the preparation of the final installation, which needs to be set up by autumn of 2006.

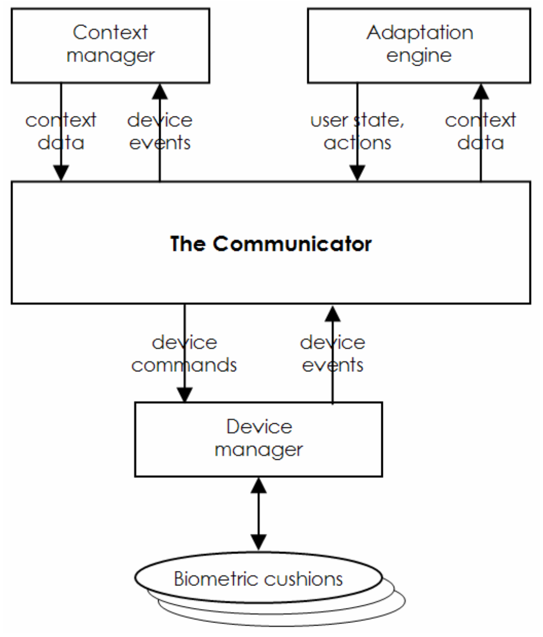

Figure 9. **Basic configuration of components in content and environment adaptation based on biometric data.**

7. CONCLUSIONS AND FUTURE WORK

In this paper, we have presented a generic platform that facilitates more flexible development of interactive multimedia in sensor-enhanced ambient environments. We have outlined basic requirements and described a platform we have developed based on these requirements. We have illustrated the flexibility of our platform through a number of prototypes of interactive multimedia applications, which explore the usage of camera-based techniques, speech, and biometric data to control and adapt multimedia content and environment.

Our platform supports building interactive multimedia applications using a component-based approach. We have used a loosely-coupled model to connect the components through a centralized communicator system. This indirect communication provides extensibility by allowing dynamic integration of components. The ability of our platform to dynamically reconfigure connections among sensing modules and other components proved useful, especially where developers needed to rapidly reconfigure interactive systems according to their needs. For example, our platform is used as a part of an artistic work that goes beyond commonly used interaction modalities, and has enabled artists to test their visions within the tight schedule provided for the installations. Usage of application adapters also enabled us to use existing multimedia content creation and playback systems without changing them, or with implementation of simple wrappers.

In future work we would like to explore how end-users could exploit the flexibility of our platform, and configure the system to use interaction modalities they prefer. The biggest challenge, therefore, is to develop tools that can automatically configure the system based on high-level description of interaction. The platform still lacks tools that can enable end-users to configure the system easily. Currently, the platform requires administration of configuration files, which, even with simple tools, is usually beyond the skills of ordinary users. This mechanism has to take into account the description of devices, supported interaction modalities, transformations, and application interfaces automatically. We would also like to explore the potential of our platform in education, to enable students to easily connect and experiment with interaction components without intensive programming efforts.

Our mechanism for embedding novel interaction modalities in Web browsers currently cannot be directly used with existing Web pages, as it requires Web pages to include our scripting applet. To solve this problem, we are also working on extending Web proxy servers that could embed such scripting code in existing Web pages on-the-fly. Proxies could also adapt the multimedia Web content based on the settings of the communicator, for example, by reducing the size or quality of content to suit the user device settings.

8. REFERENCES

[1] Aarts E., "Ambient Intelligence Drives Open Innovation", *interactions*, July-August 2005, pp. 66-68.

[2] Bannon L et al., "Hybrid design creates innovative museum experiences", *Comm. of the ACM*, Vol. 48, No. 3, March 2005, pp. 62-65.

[3] Benford S. et al., "Bridging the physical and digital in pervasive gaming", *Comm. of the ACM*, Vol. 48, No. 3, March 2005, pp. 54-57.

[4] Blattner M.M. and Glinter E.P., "Multimodal Integration", *IEEE Multimedia*, Winter 1996, pp. 14-24.

[5] Cabello-Miguel T., Fernández-Barracel O., and García-Panyella O., "iGlue.v3: An Electronics Metaphor for Multimedia Technologies Integration", *Proceedings of the*

12th annual ACM international conference on Multimedia, New York, NY, USA, October 2004 , pp. 664-651.

[6] Camurri A. et al., "Communicating Expressiveness and Affect in Multimodal Interactive Systems", *IEEE Multimedia*, January-March 2005, pp. 43-53.

[7] Davies N. et al., "Rapid Prototyping for Ubiquitous Computing: Introduction", *IEEE Pervasive Computing*, Vol. 4, No. 4, pp. 15-17, Oct-Dec, 2005.

[8] EyesWeb Platform Web Site, http://www.eyesweb.org/, Last visited: April 11, 2006.

[9] FreeTTS Project Web site, http://freetts.sourceforge.net/, Last visited: April 11, 2006.

[10] Healey J., Picard R., and Dabek F., "A New Affect-Perceiving Interface and Its Application to Personalized MUSIC Selection", *Perceptual User Interfaces (PUI) Workshop*, San Francisco, CA, November 4-6, 1998.

[11] His S., and Fait H., "RFID enhances visitors' museum experience at the Exploratorium", *Comm. of the ACM*, September 2005, Vol. 48, No. 9, pp. 60-65.

[12] Hugo Liu, Push Singh, "ConceptNet: A Practical Commonsense Reasoning Toolkit", *BT Technology Journal*, Vol. 22, No. 4, October 2004, pp. 211-226.

[13] Johanson B., Winograd T., and Fox T., "Interactive Workspaces", *Computer*, Vol. 36, No. 4, April 2003, pp. 99-101.

[14] Kameas A., and Mavrommati I., "Extrovert gadgets", *Comm. of the ACM*, Sept 2005, Vol. 48, No. 9, page 69.

[15] Kozaczynski W., Booch G., "Component-Based Software Engineering," *IEEE Software*, Vol. 15, No. 5, Sept-Oct, 1998, pp. 34-36.

[16] Lalya Gaye, Lars Erik Holmquist and Ramia Mazé, "Sonic City: The Urban Environment as a Musical Interface", *Proceedings of NIME 2003 New Interfaces for Musical Expression*, McGill University, Montreal, Canada, May 2003, pp. 109-115.

[17] Macromedia Director Xtra Extensions Web site, http://www.macromedia.com/software/director/productinfo/xtras/, Last visited: April 11, 2006.

[18] OpenCV Project Web site, http://www.intel.com/technology/computing/opencv/, Last visited: April 11, 2006.

[19] Pering T., Ballagas R., and Want R., "Spontaneous marriages of mobile devices and interactive spaces", *Comm. of the ACM*, September 2005, Vol. 48, No. 9, pp. 53-59.

[20] ROME Project Web site, https://rome.dev.java.net/, Last visited: April 11, 2006.

[21] Sphinx-4 Web site, http://cmusphinx.sourceforge.net/sphinx4/, Last visited: April 11, 2006.

[22] Stefano Bocconi et al, "Vox populi: a tool for automatically generating video documentaries", *Proceedings of the sixteenth ACM Conference on Hypertext and Hypermedia 2005*, September 2005, pp. 292-294.

[23] VeeJay Web site, http://veejay.dyne.org/, Last visited: April 11, 2006.

[24] Wakkary R. et al., "An ambient intelligence platform for physical play", *Proceedings of the 13th annual ACM international conference on Multimedia*, Singapore, 2005, pp. 764-773.

[25] Waters K. et al., "Visual Sensing of Humans for Active Public Interfaces", In R. Cipolla and A. Pentland (Eds.), *Computer Vision for Human-Machine Interaction*, Cambridge University Press, 1998, pp. 83-96.