Prototype of an Event Driven Physical Interface for Web Conferencing Systems

By

Yuntao Li

A thesis submitted to

the Faculty of Graduate Studies and Research

in partial fulfillment of

the requirements for the degree of

Master of Computer Science

Under the auspices of

Ottawa-Carleton Institute for Computer Science

School of Computer Science

Carleton University

Ottawa, Ontario, Canada

May 2012

Library and Archives Canada

Published Heritage Branch

395 Wellington Street
Ottawa ON K1A 0N4
Canada

Bibliothèque et Archives Canada

Direction du Patrimoine de l'édition

395, rue Wellington
Ottawa ON K1A 0N4
Canada

Your file Votre référence
ISBN: 978-0-494-93499-9

Our file Notre référence
ISBN: 978-0-494-93499-9

THE UNDERSIGNED RECOMMEND TO THE FACULTY OF GRADUATE

STUDIES AND RESEARCH ACCEPTANCE OF THE THESIS

Prototype of an Event Driven Physical Interface for Web Conferencing Systems

submitted by Yuntao Li

in partial fulfillment of the requirements for

the degree of Master of computer Science

Chair, School of Computer Science

Thesis Supervisor

Carleton University

May 2012

Abstract

Web conferencing systems are widely used in education and business, because they help reduce cost, increase productivity, and remove geographical barriers. However, current graphical user interfaces used by web conferencing systems impose limitations. One challenge that web conference participants face is awareness. For example, it is difficult for a presenter to become aware when another participant has a question. This thesis proposes the use of physical interfaces to complement the existing graphical interfaces used by web conferencing systems.

Physical interfaces promise to enable the implementation of a wider range of customer requirements in web conferencing systems. There is little existing work on physical interfaces for web applications, or web conferencing systems, in particular. This thesis develops an event-driven architecture through which an existing graphical user interface can be extended with a physical interface. The architecture is implemented in the BigBlueButton open source web conferencing system. Several examples of specific physical interfaces using the architecture are demonstrated.

Acknowledgements

I wish to thank my supervisor Dr. Michael Weiss for his dedication and time. His encouragement carried me through this project and his invaluable suggestions helped to shape my thesis. I feel that his feedback contributed greatly to the direction of my work and that this experience has helped to consolidate my research skills.

Furthermore, I wish to thank my wife for her loving support and helping me polish my thesis writing. I also would like to express my gratitude to the developers on the BigBlueButton team, including Fred Dixon, Richard Alam, and Denis Zgonjanin for their comments, suggestions, resources and tool provision. Thanks to all the people who gave me help in Phidgets Forums and mate.asfusion Forums. Their contributions and support have made this thesis possible.

The academic study presented here could not have been completed without my father-in-law, Professor Wu, Jun. His selfless encouragement and continued support enabled me to keep going during the tough periods of my research process. I will embalm him and his spirit in my life.

Finally, I also would like to express my deepest gratitude to my parents for their endless love, understanding, and encouragement as well as for always being there when I needed them most. Therefore, it is to the most important supporters in my life - my parents - which I dedicate this work to.

Table of Contents

List of Figures

List of Table

List of Acronyms

AMF	Action Message Format
AJAX	Asynchronous JavaScript and XML
API	Application Programming Interface
ARIA	Accessible Rich Internet Applications
BBB	BigBlueButton
CHI	Computer-Human Interaction
CLI	Command-Line Interfaces
CSS	Cascading Style Sheets
DDA	Data Driven Application
EDA	Event Driven Architecture
EJB	Enterprise Java Beans
FMS	Flash Media Server
GUI	Graphical User Interface
GUID	Globally Unique IDentifier
HTML	Hypertext Markup language
HTTP	Hypertext Transmission Protocol
ID	Identification
IDE	Integrated Development Environment
I/O	Input / Output
LED	Light Emitting Diode
MODEM	Modulator-Demodulator
MVC	Model-View-Controller
OOP	Object-Oriented Programming
PUI	Physical User Interface
REST	Representational State Transfer
RF	Radio Frequency
RFID	Radio Frequency Identification
RIA	Rich Internet Applications
RPC	Remote Procedure Call
RTMP	Real Time Messaging Protocol
SDK	Software Development Kit
SE	Software Engineering
SOA	Service Oriented Architecture
SOAP	Simple Object Access Protocol
TUI	Tangible User Interface
UI	User Interface
UML	Unified Modeling Language
URL	Uniform Resource Locator
USB	Universal Serial Bus
WAI	Web Accessibility Initiative

WLAN	Wireless Local Area Network
WSA	Web Services Architecture
WSDL	Web Service Definition Language
XHR	XML-HTTP-Request
XML	Extended Markup Language

Chapter 1 Introduction

Web conferencing systems are widely used in education and business, because they help reduce cost, increase productivity, and remove geographical barriers. Web conferencing systems are an alternative to more expensive solutions such as video conferencing rooms that require specifically designed equipment, space, and staff trained in using the equipment. All that is required to participate in a web conference is a modern web browser, and a webcam. Inexpensive web conference hosting is also available, and it is equally affordable for larger organizations to operate their own web conferencing servers. Besides commercial solutions, there are now also a number of open source web conferencing systems that have found wide adoption.

Although web conferencing systems have many advantages over specifically designed solutions, their restriction to standard hardware and browsers also imposes limitations. One of the most significant restrictions is that current web conferencing systems exclusively use graphical user interfaces. A key challenge that web conference participants face with a graphical-only user interface is awareness. For example, it is difficult for a presenter to become aware when another participant has a question, as the presentation takes over most of the limited screen real estate.

This thesis proposes to complement the existing graphical interfaces used in web conferencing systems by using physical interfaces. A physical interface allows objects in the user's environment to become part of the interface with the system. Physical interfaces promise to enable the implementation of a wider range of customer requirements in web conferencing systems. For instance, an object on the user's desk could notify them of an important status change in the conference (such as a new participant joining the conference). However, there is little existing work on physical interfaces for web applications, or web conferencing systems, in particular. The thesis aims to make a contribution towards closing this gap.

1.1 Concepts

In this section, some commonly used concepts are briefly defined. More detailed descriptions will be provided in the background chapter.

User Interface (UI): A user interface defines the point where humans and machines interact. It is used to output information and respond commands. There are many types of user interfaces in computer systems, including graphical user interfaces (GUI), voice user interfaces (VUI), and tangible user interfaces (TUI).

Tangible User Interface (TUI): A tangible user interface or Physical Interface (PI) constitutes an alternative vision for user interfaces that integrates computing back into the real world (Ishii and Ullmer 1997). A TUI maps between the state of a physical device and its digital representation. It allows users to interact with physical objects in their environment, and use them to control, and get responses from machines.

Event-Driven Architecture (EDA): An event-driven architecture promotes the production, detection, consumption of, and reaction to events (Chandy, 2006). Events correspond to important changes in state. The components of an EDA are organized in a loosely coupled manner. This includes publishers of events and subscribers to certain types of events. An EDA is related to a service-oriented architecture (SOA), as the invocation of a service can be triggered by an event. Services may also create events, which are then distributed to interested subscribers (Michelson, 2006).

Model-View-Controller (MVC): MVC is a framework that underlies the design of most current desktop and web applications. It consists of three components: model, view, and controller. The major benefit of using MVC architecture is that it decouples the user interface (views) from the underlying data (model) and application logic (controller).

The model contains the data and business rules of the application. There can be

multiple views of the same model that render the model to users in different ways.

The view displays the state of application and generates output to the users.

The controller links the user's activity, the model and the views. It receives user requests, invokes the model to handle the request, and relays the output to the corresponding view.

1.2 Motivation

Dourish and Bellotti (1992) have presented an early definition of awareness in computer supported cooperative work (CSCW) systems. Awareness is knowledge of the activities of other participants. In this thesis research, we will examine how physical interfaces can provide increased awareness for the participants in a web conference.

Many recent innovations in computing have been about ways to simplify our lives and enhance our capabilities. Barrett and Maglio (1998) describe a system that helps people browse and share digital media such as photos, videos, and music. Elliot et al. (2007) provide a solution to send reminders. Consolvo and Towle (2005) explore the use of ambient displays to help people stay in contact with their friends and family. Under the umbrella of pervasive computing, research has been conducted on the "smart" home that continuously monitors the user's health (Babulak, 2006).

Research on physical interfaces has its roots in the vision of ubiquitous computing, first articulated by Weiser (1991). Weiser envisions a future where computing is "embedded" in the environment, and provides users with helpful information where needed (Weiser and Brown, 1997). According to Dourish (2001), technology should be working within, rather than apart from, everyday practices. By that he means that technology should become part of the user's workflow, rather than requiring the user to structure their workflow around technology.

Traditional computing systems only provide a limited set of interaction devices – typically, monitor, keyboard, and mouse. On the other hand, there are many different external physical devices – sensor, actuators, and displays – in the user's environment

that could be used as interfaces. For example, users could control and receive responses from computing systems through servo motors or small embedded displays.

Connecting physical devices with computing systems requires frameworks for interfacing with these devices, and for integrating those interfaces with applications. For instance, a smart home system that consists of a range of sensors and actuators installed in different rooms in the home requires a well-designed communication mechanism for integrating those sensors and actors.

Although existing research has already studied the hardware-level interface to integrate local physical devices into software systems (Villar and Gellersen, 2007; Klemmer et al., 2004) integrating physical components into web applications, and doing so in a simple and maintainable manner, is still an open area of research. Physical devices promise to address some of the limitations of the graphical-only user interfaces used in existing web conferencing systems, in particular, in the area of awareness. Offering a solution to the awareness problem, and addressing it in a technically sensible manner will be a goal for this thesis.

1.3 Objective

To address the problems raised by graphical-only interfaces, the objective of this thesis will be to build a physical interface to a web application while limiting required changes to the existing code. The web conferencing system is chosen as an example of a web application. The thesis will deliver the physical interface module for a web conferencing system that facilitates the construction of physical user interfaces by reducing the complexity of hardware access, device exploration, network communication, and synchronization. The module will raise the level of abstraction at which developers build and extend their envisioned physical interface components, while avoiding having to deal with low-level implementation issues like device access and network synchronization.

The prototype to be built (also referred to as "physical device module" throughout

the thesis) will be based on the Phidgets toolkit for constructing physical interfaces (Greenberg and Fitchett, 2001; Phidgets Inc., 2008). Phidgets is short for "Physical Widgets", which represent the hardware equivalent of widgets in a graphical user interface. The prototype will integrate currently available Phidgets and make them easily accessible from within a web application.

To demonstrate the physical interface module, it will be used to implement different physical interfaces for an open source web conferencing system.

1.4 Thesis Contributions

This thesis will add an abstraction layer to an existing integration layer for physical interfaces, such as the one provided by the Phidgets toolkit (Phidgets Inc., 2008), that maps events in the physical interface to events in the web application. The thesis makes the following contributions to knowledge:

1. Description as constructive approach in the building prototype to prove possibility of integrating physical interface with existing graphical user interface.
2. Event-driven architecture for integrating physical interfaces which does not require changes to the existing GUI of a web application.
3. Demonstration of how this event-driven interface can be used to improve the awareness of remote users in a web conferencing system.

1.5 Thesis Outline

The thesis is organized as follows.

Chapter 2 summarizes the related literature on physical user interfaces, web conferencing systems, and event-based architectures.

Chapter 3 introduces and describes the design of the architecture of an Event Driven Physical Interface for integrating physical interfaces with web applications.

Chapter 4 demonstrates the use of the Event Driven Physical Interface and the implementation of a physical interface module for the BigBlueButton open source web conferencing system.

Chapter 5 evaluates the Event Driven Physical Interface, identifying its benefits and limitations. Finally, Chapter 6 concludes the thesis with a summary of the thesis contributions and a discussion of future work.

Chapter 2 Background

This chapter provides a background on physical user interfaces, web conferencing systems, and event-based architectures.

2.1 Web Conferencing Systems

A web conferencing system allows a group of users (also referred to as conference participants) in different locations to communicate with one another in the synchronous manner (Bafoutsou and Mentzas, 2002). To join a web conference, each user only needs to run a conferencing client in their web browser. Web conferencing has been predated by web-based chat tools and instant messaging (IM) software (Bafoutsou and Mentzas, 2002). In comparison to asynchronous communication software systems such as email, a web conferencing system is a real-time communication system that incorporates voice, video, and data (for example, a presentation) streams to provide a more convenient and user-friendly communication experience. A typical web conferencing system offers features such as Voice over IP (VOIP) to support voice conversations, video streaming, and desktop sharing.

Many features of web conferencing systems aim to reproduce elements of face-to-face meetings. For example, a presenter can share a document with all participants. Participants can communicate with each other through voice, video, or chat. Control of the web conference can also be passed from one participant to another. Today, web conferencing systems are used for business meetings, in customer support, and for long distance education as a way of reducing cost and time investment.

Given the merits of web conferencing systems, many notable vendors have already developed products, such as Citrix Online, GoToMeeting, InterCall and WebEx. Many products have features that go beyond basic functionality such as online presentations, real-time video demonstrations, and chat. For instance, some products offer the ability

to record meetings and others allow presenters to share their desktop of presenters as to annotate presentations with a virtual pen.

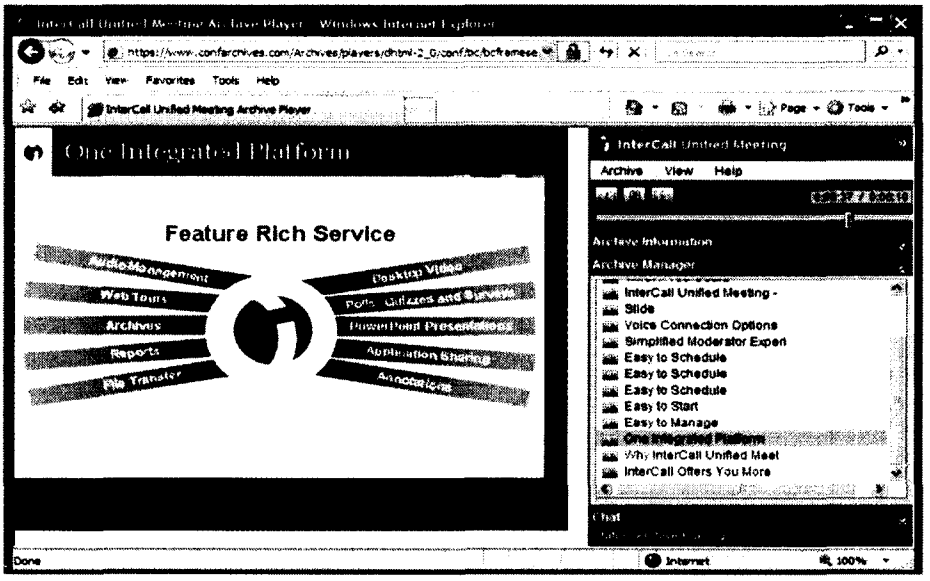


Figure 1 Web Conference – An Integrated Platform (InterCall 2010)

In the existing products, all newly-added features are displayed in an enumerated management menu. The button of mixture and sub-category folder can bring up an integrated menu which contains a task list with icons and cascading feature menus. As an example, InterCall web Conference system (Figure 1, Intercall, 2008) represents its new functions within one rolled-up category because of size restriction of the screen. Based on the menu display scenario, handling multiple features simultaneously becomes a problem to the attendees that a new triggered event might be ignored because its status cannot be displayed on screen while its mother submenu is folded. To avoid the weakness, a physical tangible interface is introduced as a potential solution because it is able to control an external device to generate a physical action when other attendees are executing a corresponding request (Preece 2002). The technologies could make up the deficiency of conference management display. Therefore, the physical interface is the key point of the web conferencing system extension research.

Besides implementing new functionality, multiple physical components are added

to the system in some products to provide higher quality services. For example, webEX (WebEX, 2009) integrates Cisco hardware to record the conversation in a meeting (Figure 2), and SmartRoom (AIRE spaces 2009) from MIT is able to provide oral control through specific hardware devices. The complexity of the specific designed equipment also limits compatibility and extensibility of the conference system. The research described in this thesis provides an attempt to expand the functionality of web conferencing systems using an Event Driven Physical Interface that is simpler to program and maintain.

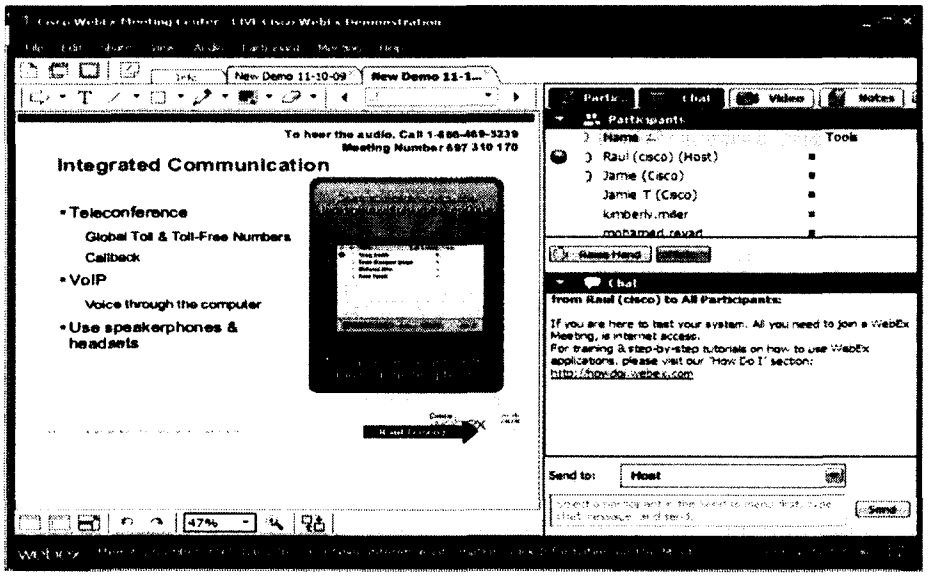
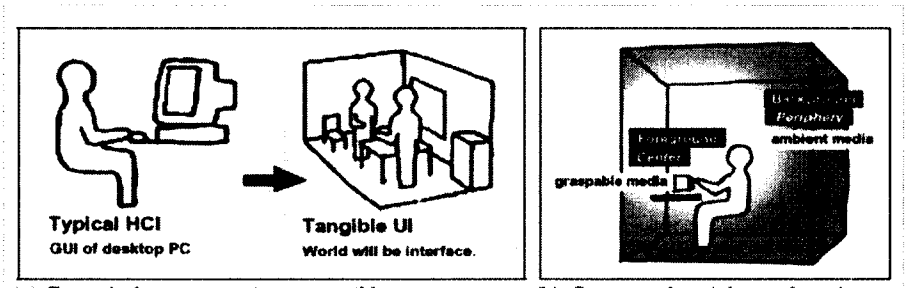


Figure 2 WebEX Conferencing System (WebEX 2009)

2.2 Tangible User Interfaces

The research of ubiquitous computing, which is the foundation of the tangible user interface, has been completed by Weiser (1993). It presents a method of enhancing computer use by turning the physical environment into a part of the digital world. Nevertheless, Weiser (1993) did not provide a method for building ubiquitous computing systems. To fill this gap, MIT's Tangible Media Group introduced the concept of tangible user interfaces. Tangible user interfaces bridge the gap between

digital and physical environments by making digital information (bits) "tangible" (Ishii and Ullmer, 1997). Other research also extends the interaction interface, such as metaDESK, transBOARD (Ullmer and Ishii, 1997), and mediaBlocks (Ullmer and Ishii, 2000). In the studies, users can access and control digital information by manipulating physical objects.



(a) From desktop computing to tangible user interfaces.

(b) Centre and periphery of user's attention.

Figure 3 Principles of tangible user interface (Ishii and Ullmer, 1997)

Compared to the three traditional I/O user interfaces in a personal computer system (keyboard, mouse and monitor), tangible interfaces can implement input and output functions on multiple physical objects that are able to submit/receive standardized digital signal transformed from objective movement. They can support input source objects such as wooden block, glass or metal, if they submit understandable signals and respond to any object that can be triggered by digital signals. The challenge of implementing tangible user interfaces is setting up a communication channel between physical objects and the core system that plays the input and/or output role during the transmission of digital information.

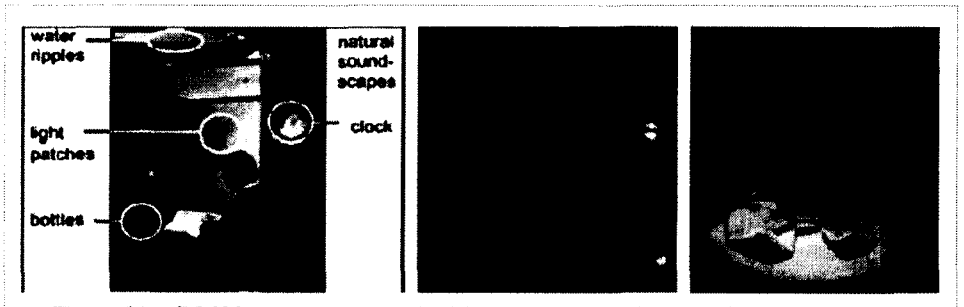
After defining tangible interfaces, a new model, illustrated in Figure 3, was introduced by Ishii and Ullmer (1997). The model is able to simulate human actions through using tangible interface technologies to collect, analyze, transform and respond data flow triggered by user behavior. In the sealed testing environment, graspable objects, such as cards or books, was set up on the surfaces of physical objects including walls, desktops and ceilings. Any movement of graspable objects triggered single flow as input of the model. At the output side, ambient media, such as sound, light and water

movement could respond to the input activity.

Buxton (1995) contributed a human-centric model in his study to offer the richer vocabulary of expression for input device. The systems can support seamless transitions running between the background and foreground interactions to expand the sorts of input device.

Furthermore, the ambientROOM project (Ishii and Ullmer, 1997), illustrated in Figure 4, aid in the understanding of the ambient media concept and application demos. The physical objects, which includes light, shadows, sound and water movement, can be used as input devices (Figure 4(a) and 4(b)) because environment changes of user perception can provide event notifications. In Figure 4(a), the movement of a clock hand is captured to trigger the output. In Figure 4(c), a bottle is used as a tangible handler for digital information management.

After Ishii and Ullmer invoked a tangible interface schema since 1997, the interactions with tangible bits have been deployed to support various physical devices by rapid developments of related research. This thesis study, Event Driven Physical Interface, draws much inspiration from previous research results.



(a) The ambientROOM overview. (b) Water lamp projections as ambient notification. (c) Bottles as physical handles to control music and light.

Figure 4 The ambientROOM project of the Tangible Media Group

2.3 Embodied Interaction

Embodied interaction using tangible computing was accomplished by Dourish (2001).

The work tends to draw both on the technical and social domain and to explain the relationship between them. According to Dourish (2001), "embodiment is the property of our engagement with the world that allows us to make it meaningful". The research opened a new window for simulating daily life activities in digital system.

Dourish (2001) established the common foundation between tangible and social computing which allows Human-Computer Interface (HCI) research to extend into the real world from the screen for testing. According to the assumption, different roles and materials in daily life environment could be utilized as physical objects. Embodied interaction does not only state physical presence of the artifacts, but also carries information and values of events and specific circumstances. Dourish (2001) describes embodied interaction as the "coextensive relationship between people using technology and connecting their social practices". The important effect of embodied technology is that the system can be aware of its context by adapting and reacting to changes in its context. Based on Dourish's research, substantive interactive application research is dedicated to find a solution on how to integrate digital technologies into the real world.

Dourish's concept and theories was designed to find a technical integration solution for social practices and activities which might be differently comprehended by developers and designers. However, no specific definition or guidelines were defined by Dourish for clarifying how to create systems to support embodied interaction. Therefore, embodied interaction is a perspective on the relationship between human people and systems instead of a technology or a set of rules. The method of developing, exploring and instantiating the implementation structure is still open investigation to all researchers.

Although short of a practical model, accomplishments in the Dourish study about the design principles of embodied interaction system still performed critical character in further research. For example, the design principles outline important concepts and stress the consideration of users when integrating the technology with social practices.

In this Event Driven Physical Interface research, module design and establishment will focus on simplifying the mechanism on simulating daily life activities into digital

system through embodied technology. The model also attempts to help developers to find an easier solution for embodied interaction through implementing a physical and tangible user interface instead of traditional electrical technologies.

2.4 Tangible Digital Information and Media

The research from the Computer-Human Interface community predicted that the next generation user interfaces could be moved off from the screen. These interfaces employ novel input techniques such as a tangible interface, camera-based interaction, access to vast information repositories in sensor networks, and information presentation to a wide range of devices (Olsen and Klemmer 2005). In this category, two examples of tangible user interface prototypes will be introduced for making digital information graspable and instantiated. These systems are directly inherited from Ishii's vision of the Tangible Bits (Ishii and Ullmer, 1997).

The marble answering machine designed by Durrell Bishop (Crampton Smith, 1995) is one example of a tangible user interface. Different from traditional phone answering machines, this system contains the marbles, and a specific container, instead of normal buttons or a screen on the phone. Figure 5 illustrates how this system works. When a new message is left on the machine, the machine drops a marble into the small container on the panel. The sequence of marble will associate with the order the messages. While the operator found marble in the container, when a new voice mail is recorded, it could be played by putting the marble into an indentation. The indentation could select corresponding messages to a different marble.

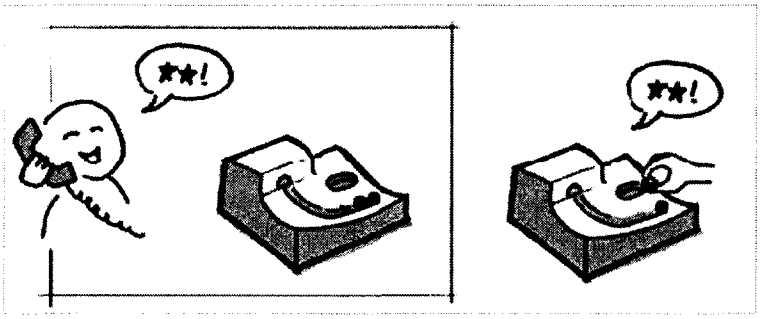


Figure 5 The marble answering machine by Durrell Bishop

Furthermore, the number of the last caller is able to be dialed back automatically if the corresponding marble was placed near an augmented machine. By means of implementing above two bidirectional functions, the marble answering machine demonstrates the mechanisms of controlling digital information from physical presence and maintaining communication through a tangible user interface.

The mediaBlocks system (Ullmer and Ishii, 2000) is another example of the integration of digital information and physical elements. In the testing environment, small wooden blocks carry and handle different digital information (shown in the left side of Figure 6). The physical object can contain multiple media, such as text, photos or video.

The mediaBlocks system adopts two tactics about tangible user interface to execute the interaction between human activities and physical device. Primarily, digital information can be carried and transferred between wood blocks. Additionally, users can modify and organize the digital contents through arranging the blocks in a sequence rack to identify the physical objects as the illustrated part on the right side of Figure 6. The mediaBlock system explores possible techniques to connect digital media information with the real world.

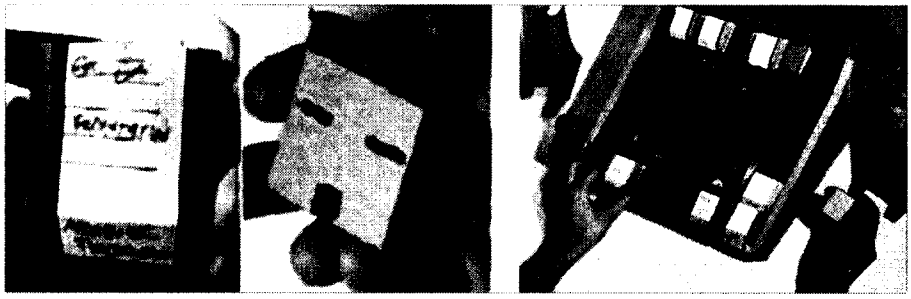


Figure 6 Tangible digital information with mediaBlocks (Ullmer, Ishii 2000)

2.5 Event-Based Systems

Event driven mechanisms have been studied since the late 1990s. Event based systems encapsulate services and data as autonomous and heterogeneous components. Those components are not designed to work in a traditional request/reply mode, but can interact via a dedicated notification service. Two components were defined in the mechanism: message and events. Carzaniga (2001) documented a routing algorithm model to support restricted event patterns. Gryphon is an information flow based approach to break down message transmission (IBM 2002). Its goal was to support monitoring the context of business activity but unfortunately it had a very complex event processing logic. Most researchers subsequently focused on avoiding broadcasts and on fault tolerance.

Unlike the request/response module in traditional software system architecture, event based systems use an event dispatcher and an event capturer to manage the system functions. In traditional systems, responding is synchronized with listening to requests. An event capturer does not require running the dispatcher in the event based system. Three critical concepts were described in event based system: event source, listener and response reaction for dealing with event dispatching.

As a valuable property, the event based system can be developed with multiple programming languages such as .Net, Java, and ActionScript. Fortunately, lots of web applications and tangible user interface systems still choose and implement event driven architecture, no matter what language it is programmed.

Most physical devices can be maintained by the event driven following the scenario in the event based system to receive and respond the digital information. Obviously, establishing the physical interface in the web conference system by using the event driven architecture becomes considerable and possible since the architecture can also be deployed into the web system. There are two benefits if the event based system mechanism is chosen as the design foundation of thesis research.

1. The event based system has well-distributed properties. The driven events could be collected and responded by multiple consumers respectively using different business scenarios through the central controller. Using the architecture, the physical user interface could handle multiple physical devices in the web conference system.

2. The event based architecture has extremely loose coupling because the event can be treated as a transparent signal of physical device for the physical user interface. The event is a factor in the information transmission process. It does not maintain the device reactions directly. Since the similarity between tangible user interface and event based system, the mechanisms could be combined, inosculated and implemented in the web conference system. The combination might accomplish better expansibility and modifiability.

2.6 Prototyping Techniques

As the descriptions in previous sections, prototyping technology can be used for designing interactive system. Hence, in the initial physical interface design of thesis research, the technology is implemented in the web conference system environment. Two general categories of the technique are low-fidelity (e.g., paper-based) and high-fidelity prototypes (e.g., software development, GUI builders) (Preece et al., 2002; Rudd et al., 1996). The definition of low-fidelity prototypes was given by Preece. It can be used to evaluate designs and developments in early design period for a project. Although various techniques can create low-fidelity prototypes, the prototypes achieve simpler, cheaper and quicker solutions because of their characteristics. On the other

hand, high-fidelity prototypes are good at extending and improving the existing system.

Considering the implementation and extendibility of the tangible user interface, this research adopts two modes of high-fidelity prototypes: the horizontal prototype strategies and vertical prototype strategies. The horizontal prototype is utilized to maintain the features for a global system. Moreover, the vertical prototype only focuses on fewer features which are implemented to manage more detailed requirements than with a horizontal prototype.

High-fidelity prototypes are useful for the testing of the user's interactions with the system (Rudd et al., 1996). They help to identify problems with the current prototype design and support the developer with decisions between implementation alternatives. As perspective of Liu and Khooshabeh (2003), it can be designed to reveal weaknesses of ubiquitous computing systems. Therefore, high-fidelity prototypes are good complements to the design sketches and storyboards of the early design phase. The drawbacks of high-fidelity prototyping strategies are also considered in the design of the Event Driven Physical Interface.

2.7 Chapter Summary

This chapter first introduced the history of web conferencing systems, which is the testing research environment of the physical interface. The necessity and importance for implementing a physical interface for web conferencing systems is then analyzed. The background information regarding tangible user interface, embodied interaction and event based systems was then presented. That information explains the technical and theory foundation of the Event Driven Physical Interface described in this thesis. Finally, the comparison of low-fidelity and high-fidelity prototyping techniques highlighted the aspects during the designing of the physical interface of the web conferencing system.

Chapter 3 Related Work

This thesis proposes an approach to build tangible interface of web conferencing system based on an Event Driven Physical Interface. To better understand the current situation of the physical interface with web technologies and choosing a sufficient, efficient, deploying and also robust module for web conference, this chapter expounds the event based technologies, MVC frameworks and related research about the web conferencing system using tangible interface.

3.1 Interface of Web Conferencing Systems

The extension research of the web conference interface is always a hot topic. In the previous achievements, the web conference system has adopted the graphical user interface. Due to the widespread of web-based technologies and acceleration of network speed, both synchronous and asynchronous communication scenario are able to be implemented in the web conference system. Asynchronous communications can maximize performance and minimize system overhead when the network cannot guarantee connectivity or availability of the target. Therefore more improvements and enhancements of quality and maintenance are anticipated.

Using current technologies, external peripheral devices such as monitors, web cameras, and microphones are able to connect to the web conference system through respective interfaces but without specific designed software support. Obviously conference system functions are limited by the number and the sort of interfaces. Therefore, a physical interface technology can be merged to a simpler architecture. The easily deployed characteristics of physical interfaces are desired for improving the web conference system.

Furthermore, to create a web conference system with higher quality, former introduced industrial research and educational institutions devote to the projects of

combining software applications and physical interfaces. Cisco designed the WebEx conferencing system to provide the cooperation solution for Cisco hardware and programmable interface. MIT designed a Smart Room using artificial intelligence technologies to schedule and manage the web conferencing system. The above systems demonstrate the possibility of integration of software and physical interface, but they all used specific hardware or a complex setting. Obviously, over-dependence on specific designed hardware obstructs to the developing and implementing of other physical toolkits in the web conferencing system although it might simplify and uniform the programming process.

Considering the limitations of the previous two solutions, the physical interface designed in this thesis provides a solution on managing physical components to expand more functions. The tangible interface is introduced to Event Driven Interface project to resolve the issues because of its prominent compatibility and extendibility. The new function that will be elaborately expounded in following chapters provides more cooperation management and stability.

In addition to implementing the above two technologies in project investigation, the project adopts horizontal prototype mechanism to design the software structure. With global management property of horizontal prototype mechanism, most functions of the web conference system could be administrated between external physical component and software source code.

3.2 Phidgets

To make the developing of the physical interface more reachable, Greenberg and Fitchett (2001) introduced a set of physical toolkits that are called Phidgets. Those toolkits simplify the process of programming and establishing a testing environment because all sensors and actuators it produced possess standard application interfaces.

Some examples of the Phidgets toolkits family developed in an HCI course taught by Greenberg at the University of Calgary are illuminated in Figure 7. They include

"tangible picture frames" (Figure 7(a)), "physical devices interactive" (Figure 7(b)), "meeting reminder machine" (Figure 7(c)), "ambient lights" (Figure 7(d)), "tangible music player" (Figure 7(e)), and a "digitally augmented storybook" (Figure 7(f)). Since the properties they have, students could concentrate on physical design of the information appliance, as well as the testing of various implementation strategies (Greenberg, 2007).

In the Phidgets toolkit scenario, the hardware control is abstracted into the software logic flow. The physical components plug into the USB port and can be programmed through a simple application programming interface (API). Multiple programming languages are supported by Phidgets toolkits (including C, Java, and ActionScript, Visual Basic).

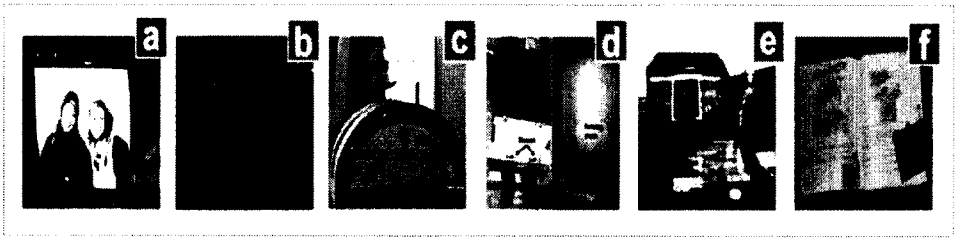


Figure 7 Prototyping examples of the Phidgets toolkit (Greenberg, 2005 2007)

Phidgets have the advantage of being easy to program and sensor/actuator integration. They can be implemented as source objects to trigger an action or receivers to respond to an activity. Furthermore, to optimize the web conference system, the project attempts to feedback a single resource event in distributed system, instead of only responding on the computer, the physical component connects.

The Event Driven Physical Interface project selects Phidgets toolkit as resource/response objects in the web conference system for the following reasons. It is possible to transplant and implement the project result in different web conference systems or equipment. Secondly, Phidgets toolkits could be accessed and controlled by a web application or by a computer embedded programming method. Therefore, an ordinary consumer without much technical knowledge is able to control the physical

devices. However, the scenario to implement Phidgets devices in network environment is a challenge to the project because former research only accomplished on a single computer connecting the Phidget toolkit directly.

3.3 Tangible Media Systems

The definition of media is a method or way of expressing something (Cambridge Online Dictionary, 2010). This means a tangible media system would be used to provide an expression method in which people can interact with. Web conferencing systems are taking advantage of multiple technologies, such as media publishing, voice over internet phone, media compression, high-speed Internet network, and so forth. The research regarding Tangible User Interface (TUI) on the web conferencing system focuses on helping users to grasp and manipulate the conference activities by handling the digital information within physical devices and the computer world.

Ishii founded the Tangible Media group in October 1995 to design and developed a seamless interface among human beings, digital information and physical objects. In later research from Ishii and his team, the simple system evolved into a suite of low-cost, low-technical notification devices for public and personal use. Ryokai (2004) from MIT's Lab built an I/O Brush (Figure 8) to help children explore colors, textures and movements.

The I/O Brush includes two basic functions that paintbrush does: picking up attributes from the real world and painting with these attributes. The demo of the electrical drawing tool provides awareness events via the multiple notification devices in physical space that can sense the computer, interact and communicate together.

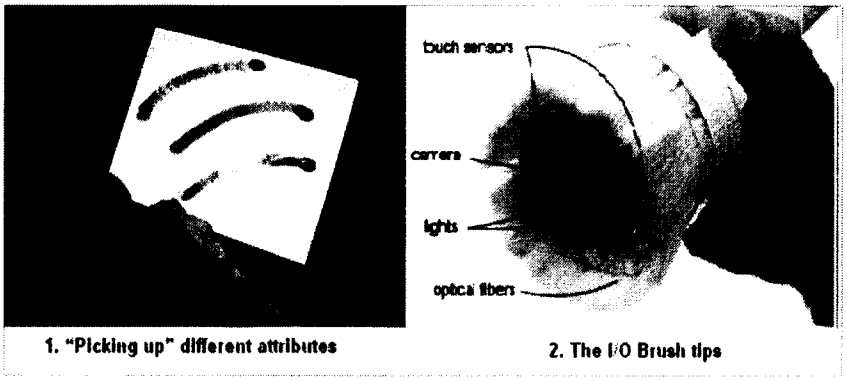


Figure 8 I/O Brush: Drawing with Everyday Objects as Ink (Ryokai et al. 2004)

As shown by these research results, the technology of tangible objects means that everything in our environment can be a medium for communication. The kind of interaction is only used to seek in the digital world. This thesis research tries to implement and design a system or physical interface to make the web conferencing system more efficient and friendly. The physical devices can be used to take accurate quantitative data that characterizes the traditional medium. For example, the remote servo can adjust the volume or change presentation slides; the combined sensors can detect the movement and action of the conference speaker.

Furthermore, networked tangible media allow people to target an individual rather than audiences. This thesis research creates a way to use medium for the web conference system that can gain interactive behaviors from the physical world instead of static web browser. The flat screens are no longer the only destination in the web conference system with equipped Event Driven Physical interface. Thus, the tangible technology literally integrates media, not just the digital message.

In this thesis research, multiple meeting activities are abstracted by a graphical user interface with icons, images and buttons on the web conferencing system. If it can be done on the web, it can likely be done in the physical world. This thesis research project processes how to make things digitally first and then transform them into tangible media.

3.4 Event Based Architecture

Two software architecture strategies, SOA (Service-Oriented) and EAD (Event Based) have been discussed in recent years (Bass 2006). Service-Oriented Architecture (SOA) principles are used for designing, developing and integrating software applications. The services in SOA systems are a set of well-defined businesses functionalities, which can be treated as reusable software components for different purposes. Each service executes one action, such as accessing a user account, processing the payment transaction, or displaying the order list. The service is able to use metadata or the data that drives them to pass or parse messages.

EDA uses a commonly accessible messaging backbone and it can offer a way of loose coupling then SOA because a central controller is not necessary (Hoff Van 2006). The requester might or might not associate with the responder within the EDA system because the event is used to generate, transmit and consume for accomplishing the application functions. The application of the EDA approach contains a service provider and a service consumer. The provider sends the event message to the broker or agent bus and it may not monitor the response from the other component within system. Then the regarding consumer can catch up with the event from the agent bus to make action. There may not be a relationship between the provider and the consumer in the EDA system.

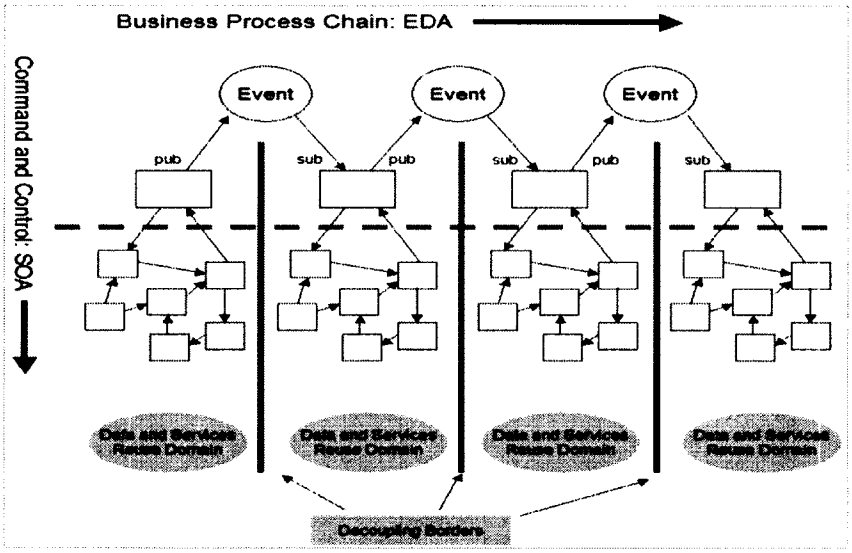


Figure 9 SOA versus EDA (Hoof 2006)

In this thesis, Event Driven concept will be used to build a interface prototype. As mentioned before, the web conferencing system has multiple function modules by adopting different technologies. To make the physical interface of the web conferencing system take advantage of loose coupled, the message delivery component through the physical interface should be separated from existing business logic. Compared with the one on one relationship in SOA (shown on Table 1), one event can be sent to multiple consumers in the EDA system. It allows the event bus to register and manage events among loosely coupled software modules and application, as is illustrate by Figure 9. This advantage will help implement and extend the physical interface of the web conferencing system without changing existing codes.

The conferencing system allows functions and facilitates more responsiveness, because event driven system has been normally designed for unpredictable and asynchronous environments. Furthermore, the Phidgets components, which this thesis project has adopted, can be handled and managed by events. Those advantage effects make the design and implement of the Event Driven physical interface to be possible. For establishing horizontal prototype dimension of this thesis research, the requirements of the physical interface, such as functions, user actions can be defined as different event processing styles. For building the vertical prototype of the interface,

more detailed components such as event metadata, event flow and so forth, should be considered and designed.

Architectural Characteristic	SOA	SOA2.0	EDA
App Interaction	Loosely Coupled	Coupled	Strongly Decoupled
App Flow Control	Synchronous Service Invocation	Synch/Asynch	Asynchronous Event Trigger
Process Coordination	Scheduler Required	(un)Scheduled	No Scheduler
Process trigger	Consumer	Consumer/Producer	Producer
Process Management	Orchestration	Orches, Pub/Sub	Publish/Subscribe
Comm Models	One-on-One	1:1, 1:M, M:M	1:1, 1:M, M:M
Business Goal	Reduce Costs, Time	Cost, Time, Visibility	ReduceCost, Visibility
Technical Goal	Component Reuse	Distributed Compute	Faster Sense/Respond

Table 1 SOA, SOA2 and EDA Defined and Illustrated (Bass 2006)

Although some web conferencing systems are built under SOA, the mechanism is different within the EDA architecture. However, SOA pattern can be used to web conferencing system with Event Driven Physical Interface because the services in SOA system can be treated as the trigger for events in the EDA system. To extend the Event Driven Physical interface into SOA system will be a very useful for ongoing research topics in the future.

3.5 Distributed MVC Framework

Although MVC is known as a framework, it is essentially architecture (Sagar 2004). It is used by n almost all modern graphics user interfaces (GUI). Events can request actions instead of controlling the flow. MVC describes large scale structure; event driven programming presents how flow is controlled. Since multiple clients access the web conferencing system remotely, it has necessary to study the ability and usage of using distributed MVC framework.

The physical interface of web conferencing systems can adopt the MVC technologies, which allow them to adapt, scale and evolve continuously; so the digital information can be shared within the virtual worlds. The hardware devices should be observed and controlled through multiple client machines. The access is mediated with a distributed Model-View-Controller pattern (Greenberg and Roseman, 1999).

Distributed physical user interface should have the availability to manage the remote and shared objects as shown in Figure 10. The shared model is used to handle the multiple views and controllers working together. The controllers (illustrated in Figure 10(a)) simulates as hardware devices, which record model changes, such as a sensor is touched or the position of the servo motor is switched. The hardware and its current status can be represented through the shared data model (Figure 10(b)). The multiple views with this abstract model can be created within the client side software application as well as the hardware model (Figure 10(c)). The controller (Figure 10(d)) can be either software application or physical devices. So the status of hardware changes will be accordingly applied to the model entries (Figure 10(e)).

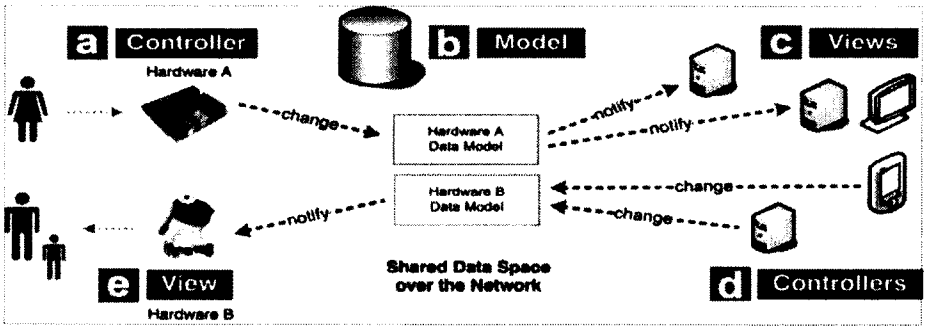


Figure 10 Distributed Model-View-Controller pattern
(Greenberg and Roseman, 1999)

In summary, the distributed Model-View-Controller pattern has the ability to represent the shared hardware with the current status or properties in the shared model. Those characteristics will be helpful and useful for the physical interface of the web conferencing system. The web conference can be attended from different places enabling the physical components of the clients participating in the web conference

separately and remotely during the web meeting. Multiple web conferencing clients' side applications or system can be operated as views and controllers at the same time.

3.6 Dynamic Updates Interface

In the traditional web application architecture, the presentation style and related data change must happen with the entire web page refresh because of the simple HTML programming. Asynchronous JavaScript and XML (AJAX) techniques improve the interactive or dynamic interface on web pages so that the web application can provide automatic updates abilities. However, it becomes more difficult and complex to implement source code into the EDA or SOA systems. Accessible Rich Internet Applications (ARIA) is a technical specification for the initiative of web service. It gives the basic concepts related with the accessibility of dynamic user interface components by using Ajax, HTML, JavaScript and other technologies. The web conferencing system also has this characterization. The applications of the web conference may consist of several components; each of them can be independently associated with different services to simulate the functions of the meeting, such as presentation board, message chatting and desk sharing. For the physical interface of the web conferencing system, status of physical devices has to be the dynamic updated as well.

Firstly, the physical devices should be initialized and recognized automatically by the physical interface. This means the physical interface has to include the programmed libraries or drivers of physical components. Application users just need to plug the physical devices into the computer which connects with the web conference system, and then the devices can take effect without additional settings.

Secondly, the status of physical devices should be updated dynamically since the application users can have different actions at the same time. For example, the network failures may cause the system to lose the connection for part of the users. The statuses of those users' activities need to be synchronized to keep users from having the same

view. This object can be achieved by programming the specific functions or by designing the global parameters.

Furthermore, the dynamic updated interface not only manages single message flow within the entire web conference system, but also shares the information among the attached physical devices. The Phidgets devices can be interacted by event messages and also allow the communication through web service, so one single status change of the physical device can be noticed to all relevance hardware and software applications. Those advantages make it is possible to establishing the physical interface for the web conferencing system.

3.7 Overview of Related Work

Several existing work related with web conferencing system, event driven protocol and tangible user interface have been studied. The physical interface of this thesis project adopts the studied technical and researches and aims to improve capability of the web conference system. The comparison of the exiting work and the improvement made by this thesis research is shown on Table 2. The more technical details will be discussed in the next chapter.

	Existing Work	This Thesis Research
Traditional Web conferencing system	Most systems use GUI and traditional I/O device	Import new physical devices by applying new physical interface
Web conference with Physical device extension	Devices are specific and expressive. Unpublished Tech.	Easy to implement and extent structure
Phidget tool Kits	Signal computer use. Seldom integrate with Web application	First case to use Phidget into Web conferencing system
Tangible Media Systems	Most research based on non-network system.	Implement tangible device into Web conferencing system
MVC Framework	Based on Web application development	Integrate physical device library within the physical interface

Table 2 Comparison Related Work with the Research Achievement

3.8 Chapter Summary

This chapter summarized related work for building an Event Driven Physical Interface for a web conferencing system. This physical interface will improve the capability and awareness of the web conferencing system. Phidgets toolkits were adopted from Greenberg and Fitchett's research to avoid the physical difficulties of programming and management. The SOA and EDA architecture have been compared for better understanding the loosely coupling, and then the event driven concept can be used for designing the physical interface of the web conferencing system. Furthermore, MVC framework and shared object availabilities were presented. The distributed and dynamic updated features of the web conferencing system were noted for building the physical interface prototype as well. Finally, the reasons for choosing Phidget and MVC framework combination theory have been described in detail.

Chapter 4 Event-Driven Approaches

This chapter focuses on the concept of adding a physical interface to a web conference system. The difficulties that interface design confront and the technical approach that research project selects are described in detail. In addition, it also analyzes how the new approach can support the integration of the physical object and the conferencing system.

4.1 Awareness in Web Conference System

Like many Internet applications, web conferencing systems have advantages in terms of convenience and economy. Using the system, participants in different locations can communicate in one virtual Internet environment. However, a current web conferencing meeting system still needs to be improved to optimize its functions and qualities. Some crucial problems considered in project solution are listed as follows:

In real face-to-face meetings, people sit at a table and communicate with each other with relative ease. One participant keeps silent when another one is talking. Attendees could receive and react to all activities and events from other users or environments instantly. Nevertheless, in a web conferencing system, the communication is limited by the processing capability of information inputs and outputs. For example, synchronizing graphical information and participants actions is an important function and needs to be improved. The issue might be caused by both hardware and software system design weaknesses.

Moreover, adding different physical components to a web conferencing system is considered by the researcher to simulate real meeting environments. Unlike traditional conferences, the web conference system is still lacking some management functions. For instance, the function of sharing information instantly from one physical device with all participants automatically is a challenge with current technology.

To resolve he information sharing and synchronization awareness, the project designs and implements a new architecture, the Event Driven Physical Interface. With the new interface, a web based conference meeting could support more external physical devices and maintain digital information from and to the device. Information sharing could also be optimized.

4.2 Design Issues in Event Driven System

Considering the characteristics of the web conference system, multiple physical devices need to be supported through different type interfaces. As QoS (quality of service) requirements, the web system pursues the global management mode controlling all physical components and digital processing software together. The Event Driven interface, a combination of tangible physical interface and event based system, is designed and implemented to resolve the issues above. The reasons and process of designing Event Driven Interface are analyzed in the following:

1. Complex functions of the conference system require multiple events with different physical user interfaces. For instance, some events take the data management job or stream control task. The activities which could be implemented by different events are listed in Table 3. The table contains human actions and corresponding system, GUI and TUI reactions which can be presented by using graphical user interface and tangible user interface. Appendix B will give more usage of physical devices for the web conferencing system in detail.

The Event based system has a loosely coupled structure and is able to be deployed rapidly and effectively. More events could be supported using an event based system. Although an Event based system is good at implementing applications in unpredictable and asynchronous environments, global management of all events is challenging because each type of physical device in the conference system has its own event definitions.

Functions in Web conferencing	Reaction	GUI	TUI
Monitoring the administrating feedback, doing survey	Notification/ Feedback	Icons Pop-Up Windows	Flash light Servo Motor
Turning slides, zooming window size or adjusting microphone volume	Presentation control	Icons, graphical control panel	Touch, Linear Sensors
Attendees automatically login in conferencing system	System Access	Inputting UserID and password	RFID Reader and RFIDs
Notify speaker if he/she moves out of the monitor ranges	Tracing movement	N/A	Servo Motor and Sensors

Table 3 Abstract GUI and TUI for reaction in Web conferencing system

2. Like the objects in software application, the statuses of physical devices also need to be taken care of. This task becomes more difficult because synchronous process needs to maintain various hardware/software resources. Increasing the number of resources complicates the physical interface structure and decelerates the processing speed of the web conference system. In the distributed system, this issue probably becomes worse because geographic compartmenting obstructs management sufficiency and efficiency.

There are two steps that can simplify the solution of previous problems; firstly, establish a uniform status management schema to trigger and capture the status of multiple physical interfaces. It means that the status has to be tracked when any action is executed. Secondly, status updating should be synchronized instantly when any activity takes place. Also, reducing network traffic could promote the system quality by registering and activating required physical devices only.

3. The service provider and requester are mediator components in the web applications with SOA architecture instead of the event sender and consumer in the EDA system. Event driven physical interfaces may work for web conferencing systems with SOA architecture because most business logic and process can be simulated as the event trigger and responder. With Web 2.0 technologies and Mashup concept development, web applications with EDA architecture can be regarded as displaying

some SOA characteristics (Hinchcliffe 2005).

In web 2.0 applications, all the service demand and supply integrate into the complex services infrastructure on the service delivery platforms. Since the services transitions in SOA and EDA system would be achievable, this thesis research will pay more attention to establishing the connection between physical components and event-driven based web conferencing systems.

4. The events as illustrated on Table 3 can abstract the activities of end user in web applications. Taxonomy of events will help to build an effective physical interface for web conference system. The status change event can be used to send end user's notification through graphical user interface and tangible user interface. The event triggered from physical devices makes user's environment to become part of the conference when the event can be consumed by web application. So the events on both directions between web conferences and physical devices enhance the usability of web conferencing system.

Furthermore, integration the physical devices' library within Event Driven Physical Interface eliminates the user installation requirement. Participants of web conference system can take the advantage as long as the low level physical complex is hidden behind the physical interface. The developers will be aware of the opportunities of physical interface because the new features can be implemented without changing existing application code.

4.3 Design Process with Physical Interface

The design process is analyzed in this section including how the physical interface is created and why relevant technologies are chosen. Some milestones will be addressed in this section to classify the difficulties in design which were introduced before.

Actually implementing tangible user interfaces using Phidgets has been demonstrated in much previous research of individual and web applications. However, two necessary adaptations are required for supporting the web conference systems.

First of all, specific libraries of Phidget products need to be installed in the Event Driven system in advance. Regarding the conference system, multiple Phidget products are running to accomplish all functions. Moreover, each sort of Phidget needs to run on its specific web service platform. Hence, combining the previous two requests and creating a new integration schema is a solution. Nevertheless, new schema by combining the above two requirements has two drawbacks.

1. It is hard to find a unique interface solution for all web services because of rapidly developing web service technology and incompatible manufactory server.

2. Even if a perfect interface is designed to adapt all web conferencing systems, the performance of the web conferencing system might be limited by unstable and costly execution.

In an Event Driven Interface project, as the first step of design, the libraries and web service platforms are installed and plugged into the system. Considering Phidget products and the web conference system utilize different web service platform, research also integrates two platforms to one. With the above optimizations, ordinary users could connect their physical devices to the web system without any programming or application installation. The web conference system becomes more user-oriented and friendly.

After the web conferencing system can communicate with physical components through physical connections, the second step of interface design is to clarify the event flow and integrate events. Interface maintains bi-directional information flow. One comes from user side, which contains human and device activities. The other is the response to all request events from the web conferencing system. To manage and integrate events for physical devices, our new schema defines an event class standard for managing event flow and reducing system redundancy. The new schema aims at preserving system performance by conserving the existing software architecture with the least amount of modification.

The initial model of event driven physical interface can be achieved by finishing the two steps stated above. Developers are able to organize the application components and the interactions through Model-View-Controller (MVC) framework.

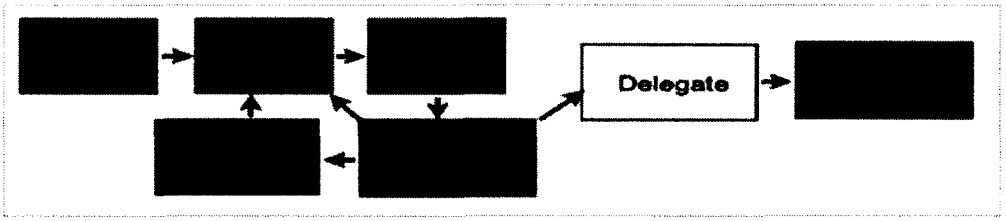


Figure 11 Mate and Pure MVC Framework (Hillerson 2008)

There are several kinds of MVC style framework, such as Pure MVC and Mate MVC. The difference between two MVC frameworks has been described by Hillerson (2008) as shown on Figure 11. PureMVC enforces a strict programming environment but development work becomes challenging with the abundance of complex source code, which makes the application structure unmanageable. The central components of the Mate MVC are the EventMap and EventHandlers. The EventMap allows application to listen to all the events whenever it registers into the system. EventHandlers are defined in the EventMap in order to control the given event. Mate also employs Dependency Injection instead of the Service/Model Locator pattern in PureMAC. The benefit of loose coupling of the Mate MVC allows the new event dispatch/capture to be deployed with seldom changing existing source code.

4.4 Using P.I. to Connect Web Conference

The target of this section is to analyze the link between physical devices and the web conferencing system. Without using tangible user interface, the relationship between software and hardware was too complex to satisfy conference requirements because it cannot identify and search virtual information from/to physical objects. The tangible user interface can make up for those defects, since it is able to extract virtual data from physical objects and aggregates it to readable and understandable information to conference attendance.

The Event Driven Physical Interface can be merged with the web conferencing system as a functional module, rather than as an external extension. This module is in charge of maintaining and administrating all physical device connections to the web conferencing system. Physical devices are represented as objects in the module: tool objects and data objects. A physical object can be in one or even both of these classes, depending of its perceived affordances (Couderc and Banatre 2003). For example, the RFID of the Phidgets component can be considered as data (the unique ID has been tagged) and as a tool (for way finding the physical object). Each tagged object can contain metadata, independent of its class. Notice the similarity in different classes of web resources: a data object can be mapped to a web fragment; a metadata object can be mapped to a type of physical resource. A tool object can be related on a web service or function of the web conference system, such as: using RFID to login automatically.

Extracting digital information from embedded physical interfaces in the web conferencing system is an important key in project solution. There are many different kinds of information (e.g attending conference, asking a question, staring a desktop sharing, etc) that need to be processed. After successfully extracting the information, the method of presenting it to users is another essential factor that needs be considered.

Event Driven Physical Interface allows conference users to attach the information to different physical devices/ objects. The objects can be identified and annotated with snippets of information. In addition, the information can be seen as the virtual representation of the physical object. Hence, it is easy to establish the associations between a physical object and an extensible web based application framework.

4.5 Overview of Event Driven Interface

The Event Driven Physical Interface of this thesis research is the first interface prototype, which integrated Phidgets devices and a web conferencing system together. The studied approach demonstrates a different possibility of connection physical user interface with existing graphical user interface.

The end user of web conference systems will be aware of the capabilities of Event Driven Physical Interface because the physical devices can make user's environment to become part of the interface of the conference. For example, the traditional graphical user interface highlights hands icon to describe the participant has question to ask during the conference. The servo motor device provides the position or speed feedback of signal. It can be read and set individually to address the status change in web conference. The notification feature will be improved if the hands-on graph icon is replaced by servo motor device. Same as user login function, the login process can be completed by scanning Radio-frequency identification (RFID) instead of inputting username and password each time. In other words, by using physical interface module the lighter quality services will be provided to the end user of web conference.

Many web conferencing systems offers physical interface to organize the physical devices used in conference. As introduced in background chapter, Cisco binds their hardware devices with physical interface solution for video conferencing system. The complicated customized development can not be avoided because of unpublished physical interface structure and specific programming language.

This thesis research exams a relatively simple solution to deploy physical interface for web application. The developer will be interested in the opportunities of using physical devices in web conference system. The event driven interface of the thesis research reduces the communication topologic between physical device and web application. The decoupling MVC framework decreases existing coding changes when the new physical device is added into web application. Furthermore web application participants do not need to install any plug-in software to use physical device because the web services library from physical devices can be integrated into physical interface itself.

4.6 Chapter Summary

In this chapter, several challenges which have to be resolved by the Event Driven Interface design were outlined. At first, the notion of awareness in a web conferencing system was analyzed and compared to existing solutions in other web conferencing system. Then the combination of the tangible physical interface and event based system was chosen as a solution, and the benefits of using Phidget technical and Mate MVC architecture were discussed. The event driven interface is reviewed to clarify boundary to existing physical interface of web applications.

Chapter 5 Case Study

This chapter will review the case study of a real interface built within an open-source online education system. The mechanism of the Event Driven Physical Interface will be evaluated by applying and deploying a physical interface for an open source web conferencing system. The methodology will be deployed and three functions using the Event Driven Physical Interface will be evaluated.

5.1 Background

BigBlueButton, former known as BlindSide project, is an open-source web based conferencing system which supports multi-media communications (Figure 12). The initial version of the system was created in the Department of Systems and Computer Engineering at Carleton University, and is now maintained by Blindside Networks. . It is currently being used for long distance education at a number of universities.

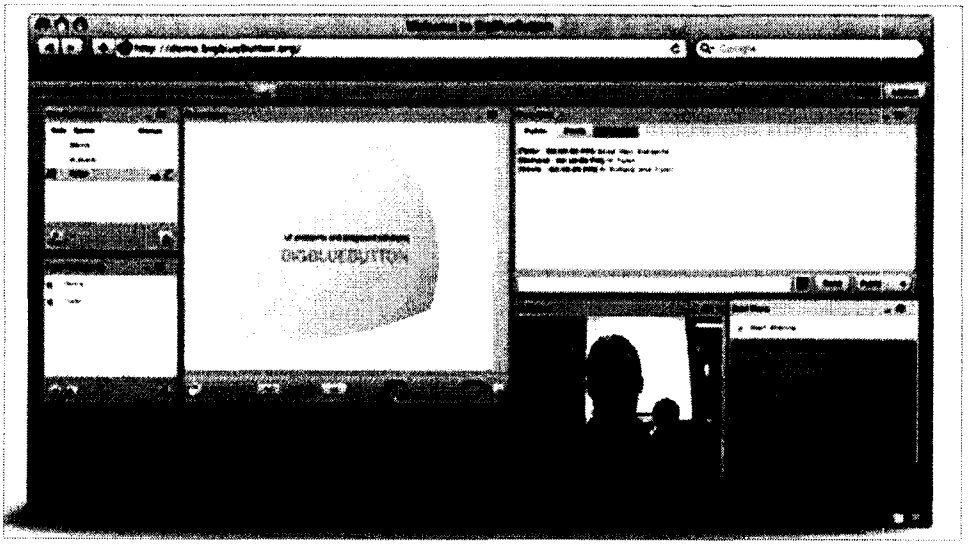


Figure 12 BigBlueButton Web Conference

The project's goal is to build a web conferencing and webcasting system by using open source web technologies. Using the BigBlueButton web conferencing solution, the attendees can access the conference through web browsers within the Internet connection. There is no application installation required on the client side.

After entering username and password, users join a specific virtual meeting room. Simultaneously, the audio system will be executed to enable the connecting local phone or embedded software microphone. Other modules of BigBlueButton web conferencing are running and cooperating in the conference environment to achieve different functions. One of these is the presentation module, which allows a user to give a live presentation with synchronized graphic display, and the user viewer module which has a 'raise hand function' to notify other participants that one attendee is asking a question.

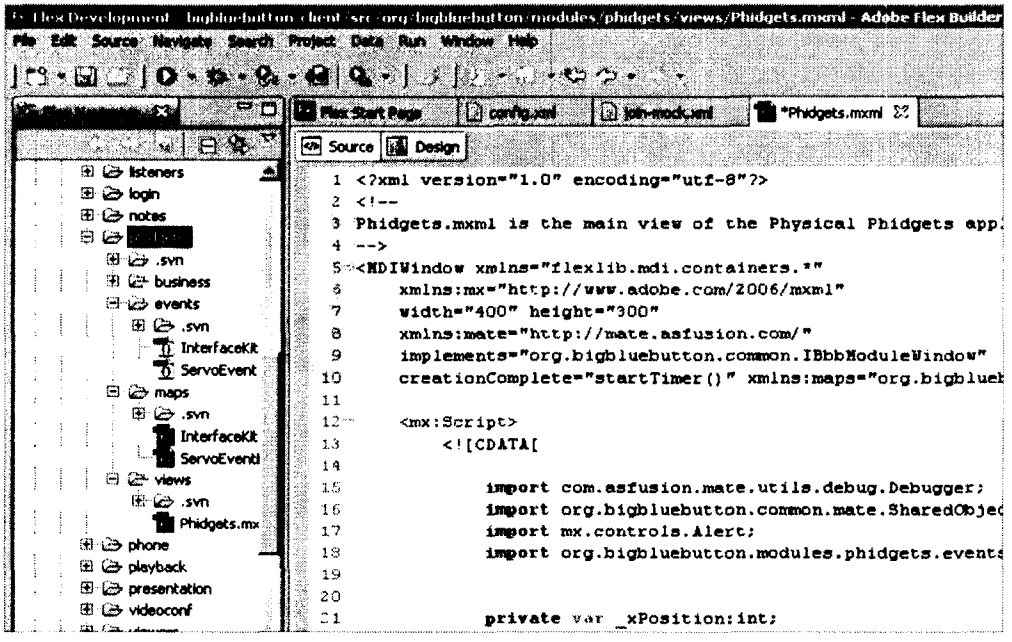


Figure 13 The project developed tool: Flex Builder 3

Since the entire BigBlueButton web conferencing system was developed in Adobe Flex programming language and environment, the physical interface module of BigBlueButton system is established based on Flex technology. Adobe Flex is an integration technology which can create Rich Internet Applications (RIA) using the

Macromedia Flash platform. In late 2007, Adobe released a new version of Flex (Adobe Flex 3, as shown on Figure 13). Flex architecture and framework have three basic components: MXML, ActionScript and a Class Library which match along with the three layers of the model-view-controller architecture.

The Macromedia eXtensible Markup Language (MXML) is specificly used to create and implement Flex applications. This markup language can manage the macros layout of application, controller components and effects. Furthermore, it can also be used to construct the unviewed components, such as xml data, web service or data source components. The class library consists of different components of Flex application layer, and it also contains elements that support HTTP service calls, image loading, WSDL calls, and the parsing of XML streams and RSS feeds.

To execute and deploy the Flex applications, the source code should first be compiled into a SWF file. There are two solutions to deploying the compiled files. Using the server compiled way, all the components of Flex applications have to be placed on the Flex runtime server until the SWF files generated, and then they are sent to the client side. The second choice is creating the SWF files within the Flex Builder, and then put them on the pre-existing server.

5.2 Related Technologies

Both Macromedia Flex Server and Red5 server can support Flex/Flash applications. BigBlueButton is using the Red5 as its data and media server. The Real Time Messaging Protocol (RTMP) is used to manage the communications between the server and clients of Flex based applications. The remote client can trigger an action on the server side and the server can also execute the procedures for multiple clients. The most important concept in RTMP is that of a Share Object.

RTMP is programmed by ActionScript and the normal ActionScript object can communicate message information between server and clients using NetConnection class of Action Message Format (AMF).

Additionally, Shared Objects in RTMP are used to synchronize the methods status and data structures of multiple clients in a conference meeting. The clients could be the external physical device or the features on GUI. Shared Objects record the status information of Flex applications on both server and client sides. It includes two parts: local shared object (LSO) or remote shared object (RSO, shown as Figure 14). The local shared object has a similar running scenario as the cookie in the web browser. Remote shared object is a special component on responding data change and synchronizing the status among multiple clients. For example, Client A produces the event; a shared object in the server broadcasts it to Client B and Client C. Shared Objects implement bi-directional communications. The persistent shared objects can remember the information on the disk and reflect it into the following connections. The transient objects are not saved and the status will lose after all the clients disconnect.

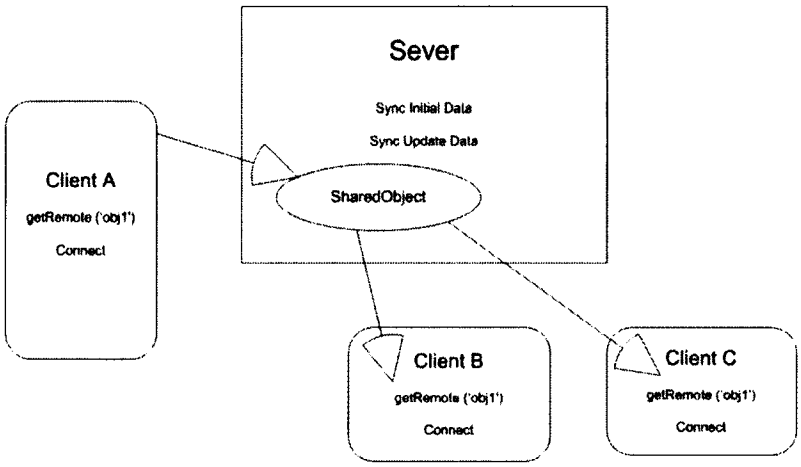


Figure 14 Remote Shared Objects (RSO)

The shared object scenario is implemented in the BigBluButton web conference system. Conference clients are able to call the scope of shared objects after it is defined in scripts on the server. During the message processing procedure, shared object records all the necessary information among providers and consumers. It works like a router in IP network, which maintains the object status and notifies subscribers when change happens on the server or the client side.

Module Nam	Description
Viewer	Display the participation list
Presentation	Controlled by presenter, can load and change the slides
Video	Show the video through the Web camera
Chat	Group chat function, like instant message within a conference
Voice	Listen and speak during a conference
Login	User login with different role, be replaced by viewer module
ShareNotes	Participates can share note, a demo of using Mate MVC
Deskshare	Share desktop image within a conference

Table 4 The Modules of BigBlueButton Web Conferencing System

There are two ways that a developer can organize the functions in Flex applications. One method is used in a dynamically loadable SWF file, which has an IFlexMoudleFactory class factory. It is not required to link class implementations with main application to load source codes of Flex application. Another solution relates to modules concepts. Flex application allows module to simulate Runtime Shared Libraries (RSLs). The module can be loaded without recompiled application, so it is more flexible than using runtime shared libraries.

The Flex applications can be constructed by multiple modules. Similar to the BigBlueButton web conferencing system, the different conference functions are split into several pieces or modules, such as voice module, presentation module, chatting module and so on. Modules in BigBlueButton (as illustrated by Table 4) are programmed in SWF files that can be loaded and unloaded dynamically. The shell, also named main application, is responsible for loading other modules when it is required. All requested modules are to be loaded into the system while the application is started. The application is able to unload the module for releasing up memory and resources when it is no longer called by the application.

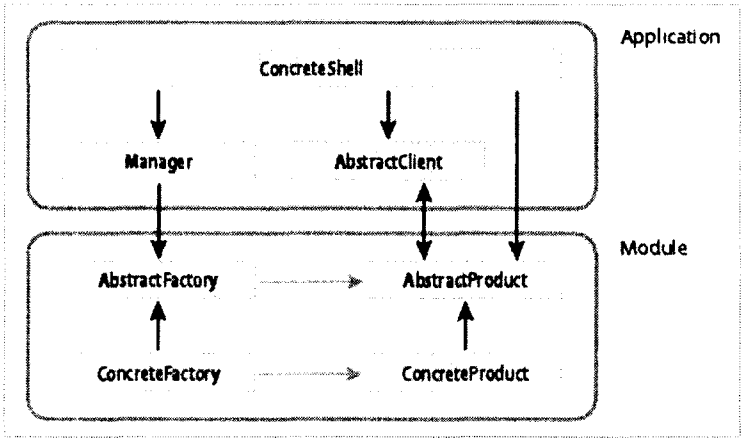


Figure 15 The relationship between shell and modules in BBB application

The BigBlueButton web conferencing system is using class factory to implement the connections between application and modules. Figure 15 shows the relationship between the shell and the interfaces of different modules. Modules split one application into several pieces. Shell is main application, which can load or unload modules dynamically. ModulerManager is indexed by the module URL and can be treated as a map to determine which modules to load for a given session. Shared interface is defined to reduce dependencies between the shell and the modules.

The relationship structure in the BigBlueButton applications uses shared interface definitions. The illustrated structure achieves more stable communication solution and makes it so that the application could be built in more abstract layers. It also simplifies the relationships among modules and makes the architecture of BigBlueButton easy to extend and implement. The Event Driven Physical Interface will implement a class factory as the standard shared interface.

5.3 Programming Process

Based on the Flex technologies introduced in the former section, BigBlueButton contains several modules whose SWF files can be loaded and unloaded dynamically. An application is able to implement multiple modules which may or may not have

pertinence between each other. By using modules, the services based interface can be established. XML data are used in BigBlueButton to determine which modules to load for a given session. After the module is loaded, a handle is created inside the module to manage the class factory. Also, an instance of a class is generated to implement the portal interface. Therefore, recompilation is required when the interface has been changed.

BigBlueButton is an event driven system. That means all the reactions are defined and built as the events. Model-View-Controller (MVC) is used to develop the software pattern of the BigBlueButton system. Events, triggered by input from the web conference system, are collected by the Controller. After that, the Controller transfers the events to a Model. Finally a Model displays the events in the View component. Due to a particular piece of input that can be used for multiple types of output, the integration and extend event in MVC framework can support the higher-level input events. The structure of the modules event is generated in the same way and gives the similar abstraction solution which allows the developer to grape the objects, events, and even the behaviors of the semantic web activities and responses.

Moreover, all business logic in BigBlueButton is simulated by event driven architecture. Modules can connect and communicate through triggering or catching events. One action generates a corresponding event, and then receives the response after the event is processed by the conference system. Many events are produced by internal function which might not relate to external user operations. Furthermore, the event handling process does not involve how the event is generated but the event information could be captured and accessed by the event object as working theory as a variable in programming language. The architecture allows the customer event to be created and dispatched within the components of the BigBlueButton system and extends the functions of the web based conference system.

There are a couple of MVC frameworks that can be used to build the Flex applications. The first widespread one used in the BigBlueButton system is Cairngorm MVC framework. It was then changed to PureMVC framework for a while. The Mate MVC was implemented by the conference system recently since it is a declarative

programming framework. Based on the benefits of Mate framework addressed in the former chapter, the codes can be simply compiled to ActionScripts and they are not necessary to be maintained as the object to object calls (BigBlueButton 2009).

Although the Mate framework is able to support most common tasks for Flex application such as processing the HTTP services and SQL queries, it still has limitations (BigBlueButton 2009). The development team of the BigBlueButton extended the Mate MVC framework to allow the Remote Shared Object to be maintained among the components of the BigBlueButton system. This extension works like the bridge connecting BigBlueButon modules and passes the information within the entire system.

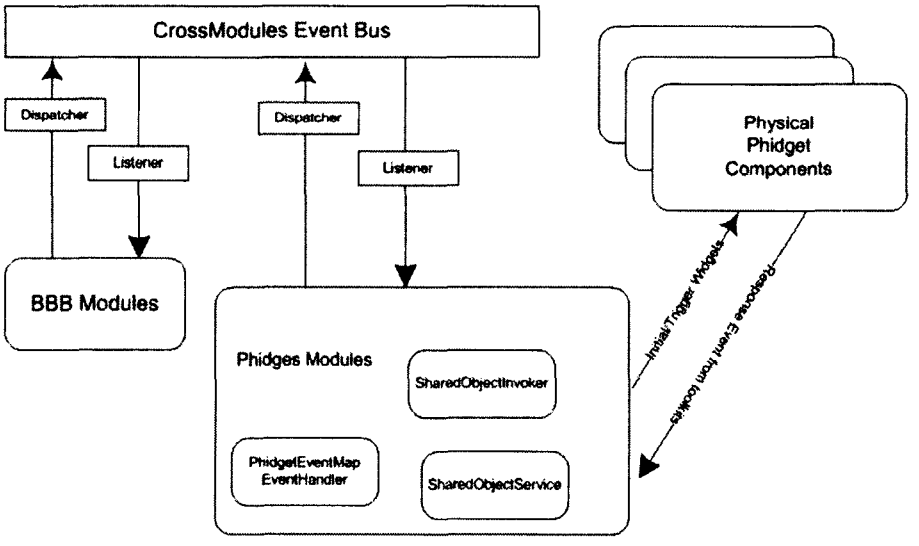


Figure 16 Architecture of Phidgets Module using Mate

The architecture of the BigBlueButton physical module is shown in Figure 16. The events of Phidgets devices can be generated and captured by the modules in the web conferencing system. Due to the fact that it is an inside component of Flex application, those events have the capability to react other modules in the BigBlueButton system. In addition, the Phidgets module can be treated as the control panel of the physical devices to manage hardware reactions and movements. More mechanisms of the Phidget device

will be explained in following section.

5.4 Event Driven Physical Interface

Constructing and programming physical hardware is usually onerous because it requires a great deal of professional knowledge for implementing conference functions. To accomplish it, several Phidget widgets are chosen and analyzed for different relationships to achieve thesis project target.

Primarily, PhidgetInterfaceKit provides a physical panel that multiple external devices could plug in. It controls the physical connection and virtual input/output from/to the devices. Generally speaking, one panel can control eight digital output devices (e.g. LEDs and solenoids); retrieve the state of eight digital input devices (e.g. push buttons and switches) and inspect the state of four analog sensors (e.g. potentiometers, heat, force, capacitive plates and light sensors, as shown in Table 5) simultaneously with the commands from the programmer.

PhidgetRFID is a class of Phidget devices which contain an RFID tag and an RFID tag reader (as shown in Table 6). The RFID tag carries a unique RFID ID that could be identified by the RFID reader through scanning the message sent by the RFID tag antenna.

PhidgetServo comprises of one or more servo motors (as shown in Table 7), and the motor position is easily configured through software API.

To apply Event Driven Physical Interface in the web conference system, the event object shares information of the event including event name, the object dispatched at the event, the context of the event and other detailed information for an understanding occurrence process. In addition, the programming code is decoupled because the Mate framework is integrated in Events. The purpose of the Event Driven Physical Interface is using its characteristics and abilities to connect physical devices in the web conference system.

The implementation of the new API is limited in a physical module by itself. That

means the events can be created and consumed by entities in another BigBlueButton module without changing any existing modules. For example, the graphical raise hand feature can dispatch handsDown event and handsUp event as shown on view module part of Figure 17. The physical raise hand function need only be triggered by the raise hand events, which were dispatched with the graphical raise hand feature untouched.

Phidgets module, the physical module of BigBlueButton, represents the prototype of the Event Driven Physical Interface of BigBlueButton. The module is responsible for generating and handling the physical device events. The input layer is in charge of the sensor input, interpreting input, and generating the Events. Phidgets module has the benefits as it does limited changes on source code of existing modules and it has fast deploying physical functions.

With the properties of Phidgets module, the developers only need focus on selecting input types such as Servo or Vision. They do not need to discover the input devices attached to the computer manually or establish a connection between device and conference system, or generate events from the input. All the lower level work, such as connection and registration of the physical devices, are completed by the physical interface. The physical interface module should have the availability to control the hardware version and designed functions

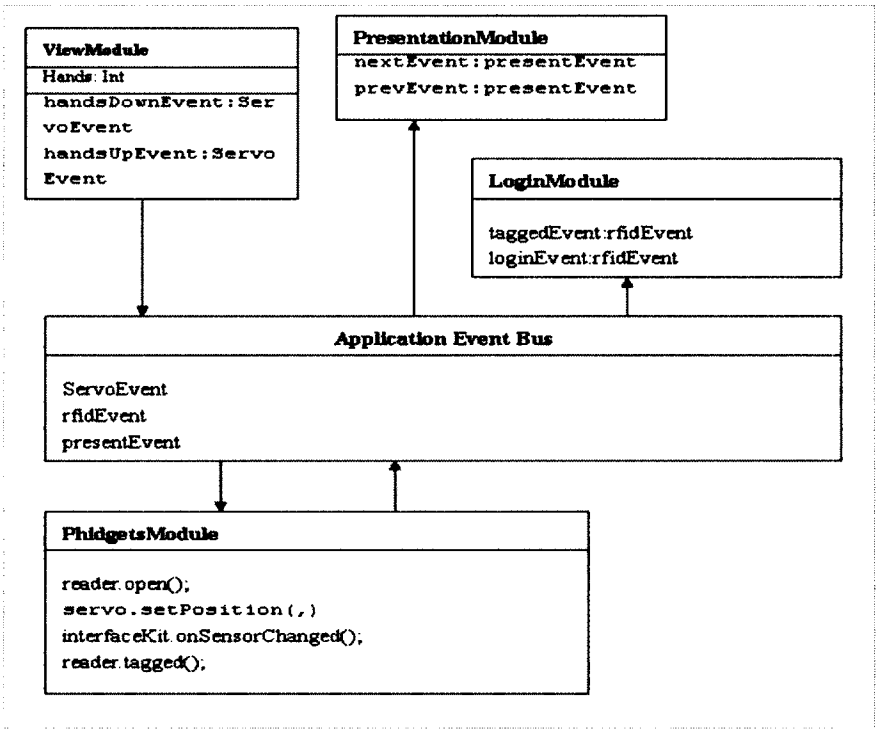


Figure 17 Event Flow of Phidget Module

Once the developer has selected an input source, the module generates the events representing inserting, updating, and removal of objects from a sensor view. Event types are consistent across all technologies. Providing high-level events substantially lowers the application development threshold and facilitates technology portability. While all technologies fire the same events, different technologies provide different types of information about the physical objects they sense.

Besides setting up the communication tunnel, the Event Driven Physical interface can be used to discover the status of features in the web conference system. For example, Servo provides the moving activity and position information. User behaviors and application functions could be synchronized through monitoring the position of physical devices. When an event is generated and inserted into the Phidgets module, a coordination process is triggered at the same time when the event consumer activates the designed response.

Developers can adopt this additional process to reset the incorrect status of Servo device or to disconnect the expired device from the web conference system. It is

necessary that implementation of distributed computing method manages multiple devices connecting from distributed located points. Hence the web conference system provides a standard management platform to all application users no matter what the physical devices are being used.

5.5 Evaluation of Interfaces

Three specific features in the Phidget Module of the BigBlueButton web conferencing system are created to illustrate the Event Driven Physical Interface. The interface is defined as a set of named operations that can be invoked by the client. It is able to emerge internal functions and component features of the web conferencing system to external devices. Considered as a module, the Event Driven Physical Interface could not only communicate with other modules but also coordinate information transmission among multiple modules.

The first case showed here uses the Phidget Servo Motor (Table 7) to enhance the question notification requirement in the web conferencing system. Hands-up function (Figure 18) is used to monitor the behaviours of participators which replace the notification feature in GUI of BigBlueButton but it does not modify existing GUI. The HandUp event is generated by Viewer Module when the attendee pushes the hands icon on screen and applies to be noticed. After the Event Bus of BigBlueButton listened and processed the HandUp event, the corresponding EventHander controller of Phidgets Module is to be triggered and respond to it.

It is possible that multiple attendees press the hands icon to active the unique Motor position of the physical component at the same time and because of this a HandsUp counter has been designed to administrate this many-to-one mapping relationship. In the new mechanism, the Servo Motor will be returned to hands down position if and only if all questions have been answered and the counter is cleared. Consequently, the presenter can concentrate on his or her speech without missing the questions.

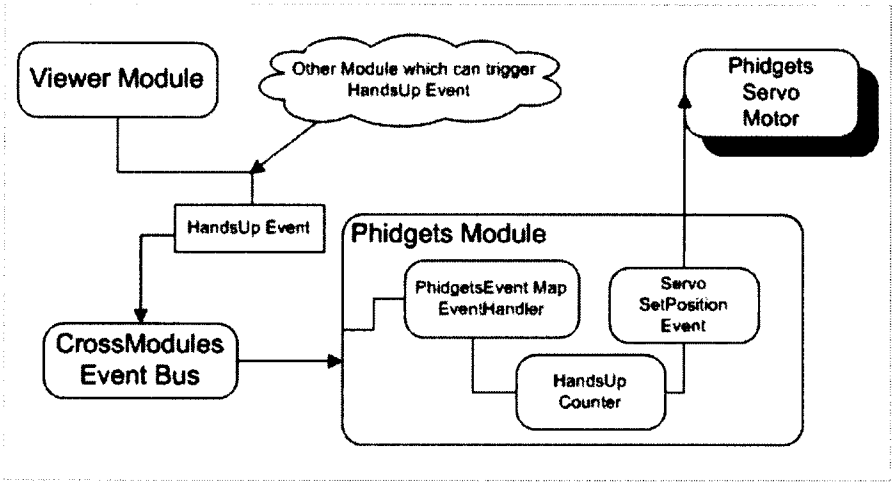


Figure 18 Hands-up Function using Phidgets Servo Motor

Compared with the first case, two sensors and Phidget Interface Kit (Table 5) are utilized for slides-change function by representing the event flow from physical tool into the BigBlueButton web conferencing system.

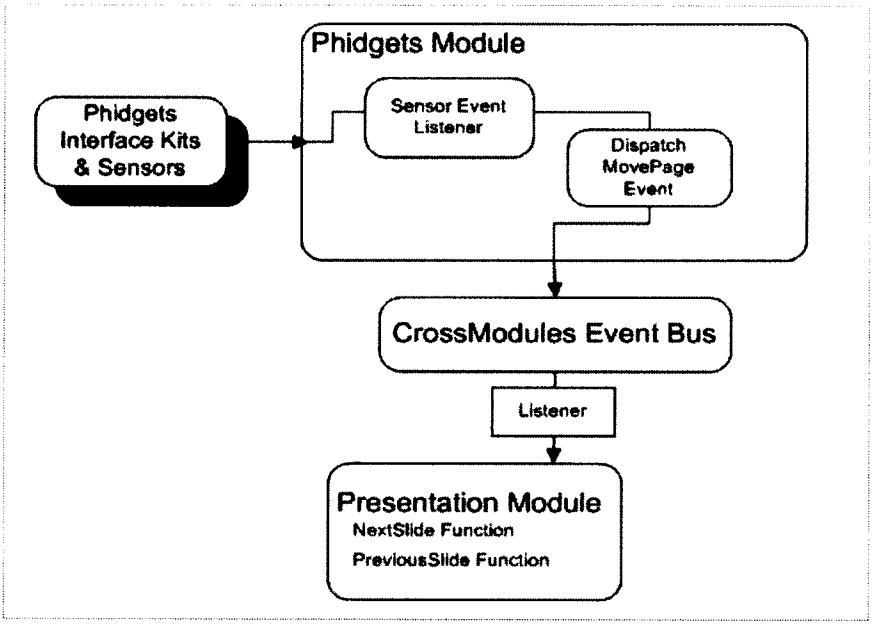


Figure 19 Playing the Sensors to Present

The force sensor and push sensor are selected to control the NextSlide function and PreviousSlide function. After loading the Phidgets Module, the slides-change function

keeps listening to the sensor value change events from Phidget Tool Kit. Then the MovePage event in Phidgets Module is trigged and delivered to the event bus. Finally the Presentation Module responds the slide change actions through the listener of the module (Figure 19).

The last case of Phidgets Module is Sweep-to-Login function (Figure 20) using RFID tools (Table 6), which gives a good example of how events are handled among different modules. It shows that the Phidgets Module does not only have the ability to respond or trigger events, but that it can also communicate with the web conferencing system through Event Driven Physical Interface. The Login Module generates the initialization event to start the listener of the RFID reader and wait for the response user information from Event Bus. This method shows that the Phidgets widgets can be loaded only if it requests and saves the resources and memory of the conference system. Moreover, the Phidgets Module is able to collect and send the account information regarding the unique ID caught by RFID reader.

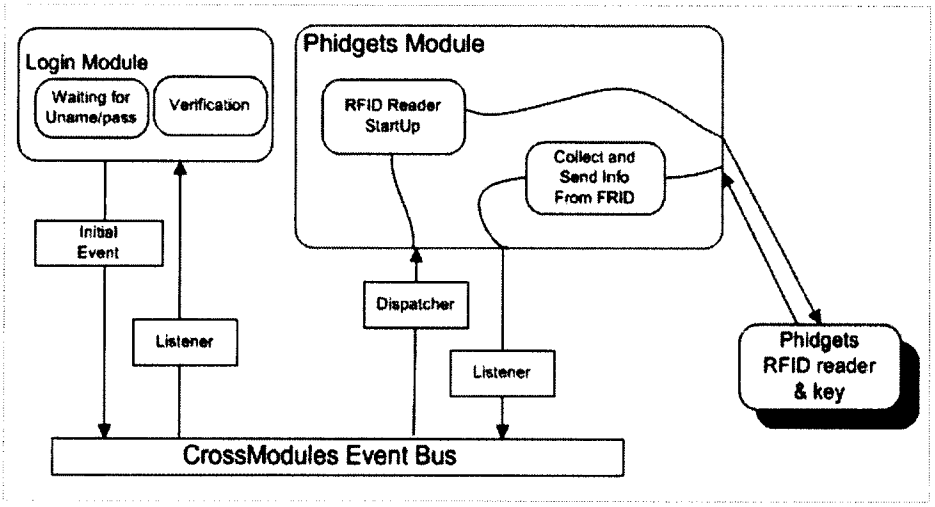


Figure 20 Fast and Security Login through Two-Way Physical Interface

Based on the above Phidgets Module implementation processes, two benefits of Event Driven Physical Interface can be easily drawn. Primarily, it is easy to implement. There are only three steps to activate the Phidgets Module: setting up the physical components, loading the Phidget Module and playing with the Phidget devices.

Secondly, for modifying and extending the function of the BigBlueButton conferencing system, the developers only need to consider dispatching and/or catching up the events through the Event Bus. All implementations do not involve modifying existing source code of BigBlueButton web conference system because the Event Driven Physical interfaces only work as a communication tunnel of different events instead of managing event execution process.

5.6 Case Study Summary

This chapter presented a case study of the Event Driven Physical Interface in the BigBlueButton conferencing system environment. First of all, it introduced the system background and technical details of the BigBlueButton web conferencing system. Secondly, the developing process and the generic description of the physical interface module prototype have been presented. Furthermore, Event Driven Physical Interface was designed using Mate theory. Finally, this chapter also gave three examples, hands-up, slides-change and Sweep-to-Login functions about implementing Phidgets devices in the web conference system with physical interface to elaborate the capability of the prototype of the Event Driven Physical Interface.

Chapter 6 Conclusions and Future Work

In the previous chapters, the methodology, design, implementation, and case study of Event Driven Physical Interface for the web conference system were elaborated. In chapter six, the application of the Event Driven Physical Interface to a web conferencing system will be reviewed. Finally, the chapter will indicate areas for future work.

6.1 Summary of Contributions

The main objective of this research is to validate an approach to add physical interface to the graphical user interface of web application. To achieve the objective, the concept of the Event Driven Physical Interface is introduced and a BigBlueBlue module for physical interface is proposed to accomplish the remote access to Phidgets hardware. The implementation of the physical module and the performances of related functions in the case study contribute the following conclusions:

- To analyze that Event Based physical interface can be deployed into the web conferencing system without changing the existing source code.
- To design a flexible Physical Module to implement the integration of Phidget hardware and existing web conferencing system.
- To verify that the features based on Event Driven Physical Interface and the Physical Module achieve the essential requests of the web conferencing system. So the end user can control the hardware remotely.
- To demonstrate how the Event Based physical interface can be used to improve the awareness of remote users in a web conferencing system.
- To give the suggestions to further investigation of the physical interface in the web conferencing system by exhibiting the availabilities of Phidgets components and the functions in the web conference.

- To anticipate the future works on the web conferencing system by providing the possibilities to improve the web conferencing system performance and to implement more functions.

6.2 Limitations and Future Work

Although the research of the new integration prototype of the physical interface and current web conferencing system introduced in the thesis accomplish some developments and improvements on physical user interfaces, some unsolved issues still exist and can be investigated in further study. This section overviews the limitations and proposed future research areas for extending the performances and functions of the web application or web conferencing system.

Limitations

The interface prototype of this thesis research is built as a module of BigBlueButton web conference system. The capabilities of Event Driven Physical Interface are based on features and the architecture of BigBlueButton web conference system. However, the studied approach provides a framework, which can be integrated as event driven physical interface for other web applications or web conference systems. For example: Vidyo (Vidyo Inc. 2005), OpenMettings (Google 2011), and so on. The decoupling structure reduces the coding changes on existing web application.

Furthermore, the Phidget module is established within Adobe Flex programming language and environment because BigBlueButton web conference system chosen the technology. Since Phidget API allows systems to access the phidget devices in a high level manner. The attached devices can subscribe the events and access the state of the phidgets through a variety of programming languages, include Java, .Net, Python Visual Basic, etc. The Event Driven Physical interface can be extended into another web application, which programmed by different programming languages. Nevertheless the prototype of this research was specific to Flex technology, the

approach of building physical interface may be concerned as an effective framework to establish connection between web application and physical devices.

Further Evaluations

The presented case studies were an initial evaluation of the physical event driven API of the web conferencing system and only three functions appliances built on that. Even though these case studies illustrated the communication capabilities of Event Driven Physical Interface and easy implementation advantage of MVC architecture, it is still important to optimize the development process in further detail.

First of all, The EventMaps technology in Mate MVC framework has difficulties to track data flow within application developing. Additionally, Mate MVC is short of the standard management strategy to administrate the events since all tested functions are designed respectively. To resolve the issues, the better organized relationship of event flows is required in physical interface module of the web conferencing system. New strategy is required to catalog all events distributed in different function to avoid redundancy flow and simplify event design process for new extension.

Secondly, the capacity of the fault-tolerance control for the Event Driven Physical Interface needs to be improved. In the current conference system, if a user triggered an event but the event is dropped or truncated by a network issue, the system is not able to detect his status change immediately and reset it for synchronization. More investigations are required on fault-tolerance control and alarm strategy to optimize web conference performance.

Furthermore, implementing extending source code of the web conference system for further functions is a challenge to new researchers because of the limitation from parameter settings in Event Driven Physical interface. For instance, the raise hand function has a hands-up counter to record number of attendees who want to ask a question. The counter is arranged in specific module of web conference system to reduce programming complex. Globe variable or another mechanism to discover the attendees may be a better solution to achieve this aim and integrate the current requirements.

Module Extensions

The MVC module architecture and developer library of the Phidgets module already support some of diverse hardware devices. Most of the developing tasks will focus on demonstration of user activities in the web conference. The physical complex managements are hidden behind the Event Driven Physical Interface. Furthermore this kind of structure enables designer and developer to extend the web conferencing system easily and rapidly.

To deploy the new physical function, the former source code of graphical user interface and specific modules in the web conferencing system are not required to change as long as the event itself is not modified. Furthermore, the existing events regarding the physical interface can be collected to improve and optimize graphical user interfaces. The current events are designed to work in the tangible user interface and the web conference system for the specific function. An event management container will help the Event Driven interface to handle event flow and improve the software logic of the web conference system.

There are more opportunities that can be found to excavate the requirement of a web conferencing system by using other Phidgets components. To learn and analyze the attendees' behaviors in the web conference system is also important. The embedded system and tangible user interfaces technologies provide lots of opportunities to simulate people's activities. The useful physical devices can improve user awareness of using the web conference system based on the clear understanding of the requirements of functions. Therefore, integrating existing functions of the web conference system with useful physical components is another major further work in the future.

Reference

Abowd, G. D. 1999. Software Engineering Issues for Ubiquitous Computing. In Proceedings of the 21st International Conference on Software Engineering - ICSE 1999 (Los Angeles, CA, USA). IEEE Computer Society Press, Los Alamitos, CA, USA, 75-84.

Aaron, A. and Davis, R. 2004. Speech and Sketching for Multimodal Design. In Proceedings of the 9th International Conference on Intelligent User Interfaces, pp.214--216. 2004.

AIRE spaces. 2009. http://aire.csail.mit.edu/projects.shtml, website last reviewed on Dec 9th, 2009

Babulak, E. 2006. Automated Smart House 2015 via Ubiquitous Computing. In Proceedings of the International Conference on Interactive Mobile and Computer Aided learning ¨C IMCL 2006 (Amman, Jordan).

Bafoutsous, G., and Mentzas, G. 2002. Review and functional classification of collaborative systems. International Journal of Information Management. 22, 281-305.

Ballagas, R., Memon, F., Reiners, R., and Borchers, J. 2007. iStuff Mobile: Rapidly Prototyping New Mobile Phone Interfaces for Ubiquitous Computing. In Proceedings of the 25th ACM Conference on Human Factors in Computing Systems ¨CCHI 2007 (San Jose, CA, USA). ACM Press, New York, NY, USA

Banavar, G., Kaplan, M., Shaw, K., Strom, R., Sturman, D., and Tao, W. 1999. Information flow based event distribution middleware. In Proceeding of ICDCS'99 Middleware Workshop.

Barolli, L., and Koyama, A. 2005. A Web-based Distance Learning System Using Cooperative Agents. Chapter in Encyclopedia of Online. Learning and Technology. (2005) 430-439

Barrett, R. and Maglio, P. P. 1998. Information Things: How to Attach Information to the Real World. In Proceedings of the 11th Annual ACM Symposium on User Interface Software and Technology-UIST 1998 (San Francisco, CA, USA). ACM Press, New York, NY, USA, 81-88

Bass, T. 2006. SOA, SOA2.0 and EDA Defined and Illustrated. http://www.slideshare.net/ TimBassCEP/soa-soa-20-and-eda-defined-and-illustrated-presentation

BigBlueButton. 2009. http://code.google.com/p/bigbluebutton/ Website last visited on October 20, 2009

Burge, J. and Brown, D. C. 2006. Rationale-based support for software maintenance. Rationale Management in Software Engineering.

Buxton, W. A. S. 2007. Sketching User Experience : Getting the Design Right and the Right Design. Morgan Kaufmann Publishers, San Francisco, CA, USA.

Carzaniga, A., Rosenblum, D.S., and Wolf, A.L. 2001: Design and Evaluation of a Wide-Area Event Notification Service. ACM Trans. on Comp. Sys. 19 (2001) 332-383.

Chen, H., and Perich, F. 2004. Intelligent Agents Meet Semantic Web in a Smart Meeting Room. Proceedings of the Third International Joint Conference. Jan 2004

Chen, Y.-X., and Butz, A. 2009. MusicSim: Integrating Audio Analysis and User Feedback in an Interactive Music Browsing UI 14th international conference on Intelligent user interfaces, IUI 2009, Sanibel Island, Florida, USA, February 8-11, 2009.

Consolvo, S., and Towle, J. 2005. Evaluating an Ambient Display for the Home. In Extended Abstracts of the 23th ACM Conference on Human Factors in Computer Systems ¨C CHI 2005 (Portland, Oregon, USA). ACM Press, New York, NY, USA 1304-1307

Couderc P., and Banatre, M. 2003. Spreading the Web. In M. Conti, S. Giordano, E.Gregori, and S. Olariu, editors, PWC, volume 2775 of Lecture Notes in Computer Science, Pages 375-384, Springer, 2003.

Dourish, P. 2001. Where the Action Is: The Foundations of Embodied Interaction. The MIT Press

Dourish, P., and Bellotti, V. 1992. Awareness and Coordination in Shared Workspaces. In Proceedings of ACM Conference on Computer Supported Cooperative Work, Toronto, Ontario, ACM Press, 1992.

Ducheneaut, N., Smith, T. F., Begole, J. B., Newman, M. W., and Beckmann, C. 2006. The Orbital Browser: Composing Ubicomp Services Using Only Rotation and Selection. In Extended Abstracts of the 24th ACM Conference on Human Factors in Computing Systems - CHI 2006 (Montreal, Quebec, Canada). ACM Press, New York, NY, USA, 321-326.

Ellot, K., Watson, M., Neustaedter, C., and Greenberg, S. 2007. Location Dependent Information Appliances for the Home. In Proceedings Graphics Interface ¨CGI 2007 (Montreal, Quebec, Canada)

Eugster, P. T. and Guerraoui, R. 2001. Content-Based Publish/Subscribe with Strucutural Reflection. In Proc. of the 6th USENIX Conf. on Object-Oriented Technologies and Systems (COOTS01), Jan.

Greenberg, S. and Fitchett, C. 2001. Phidgets: Easy Development of Physical Interfaces Through Physical Widgets. In Proceedings of the 14th Annual ACM Symposium on User Interface Software and Technology ¨C UIST 2001 (Orlando, Florida, USA). ACM Press, New York, NY, USA, 209-218

Greenberg, S. and Roseman, M. 1999. Groupware Toolkits for Synchronous Work. In Computer-Supported Cooperative Work (Trends in Software 7), M. Beaudouin-Lafon, Ed. Vol. 7. Wiley & Sons, Chapter 6, 135-168.

Greenberg, S. 2007. Toolkits and Interface Creativity. Multimedia Tools and Applications 32, 2, 139-159.

Gryphon. 2002. Publish/Subscribe Over Public Networks. IBM T J Watson Research Center Whitepaper (2002).

Gunderson, J, Schwerdtfeger, R. 2006 Roadmap for Accessible Rich Internet Applications (WAI-ARIA Roadmap), http://www.w3.org/TR/aria-roadmap/

Hillerson, T. 2008. Mate, the Pure MXML Framework. http://www.insideria.com/2008/12/frameworkquest-2008-part-5-mat.html

Hoof, Van, J. 2006 How EDA extends SOA and Why it is important. Http://soa-eda.blogsopt.com

Hudson, S. E. and Mankoff, J. 2006. Rapid Construction of Functioning Physical Interfaces from Cardboard, Thumbtacks, Tin Foil and Masking Tape. In Proceedings of the 19th Annual ACM Symposium on User Interface Software and Technology - UIST 2006 (Montreux, Switzerland). ACM Press, New York, NY, USA, 289-298.

InterCall. 2010. InterCall Unified Meeting solution. Http://www.intercall.ca The website last review in April 2010.

Ishii, H., and Ulimer, B. 1997. Tangible Bits: Towards Seamless Interfaces Between People, Bits and Atoms. In Proceedings of the ACM Conference on Human Factors in Computing Systems - CHI 1997 (Atlanta, Georgia, USA). ACM Press, New York, NY, USA, 234-241.

Johanson, B., and Fox, A. 2002. The Event Heap: A Coordination Infrastructure for Interactive Workspaces. In Proceedings of the Fourth IEEE Workshop on Mobile Computing Systems and Applications. IEEE Computer Society, Washington, DC, USA, 83-93.

Chandy, K. M. 2006 Event-Driven Applications: Costs, Benefits and Design Approaches. Presented at the Gartner Applicaton Integration and Web Services Summit, San Diego, CA, June 2006

Kasal, A., Liu, J., Miller, J., Nath, S., Rouhana, D., Santanche, A., and Zhao, F. 2007. SenseWeb Project. http://research.microsoft.com/nec/senseweb/. Website. Website last visited on September 18, 2009.

Kim, S. W., Kim, M. C., Park, S. H., Jin, Y. K., and Choi, W. S. 2004. Gate Reminder: A Design Case of a Smart Reminder. In Proceedings of the 5th ACM Conference on Designing Interactive Systems - DIS 2004 (Cambridge, Massachusetts, USA). ACM Press, New York, NY, USA, 81-90.

Klemmer, S. R., Li, J., Lin, J., and Landay, J. A. 2004. Papier-Mache: Toolkit Support for Tangible Input. In Proceedings of the ACM Conference on Human Factors in Computing Systems - CHI 2004 (Vienna, Austria). ACM Press, New York, NY, USA, 399-406.

Laliwalla, Z. 2006 Event-driven dynamic web services composition and automation of business processes. In International Conference on Services Computing, 2006.

Lee, J. 1997. Design rationale systems: Understanding the issues. IEEE Expert, Vol. 12, No. 3:78-85.

Lee, J. C., Avarhami, D., Hudson, S. E., Forlizzi, J., Dietz, P. H., and Letgh, D. 2004. The Calder Toolkit: Wired and Wireless Components for Rapidly Prototyping Interactive Devices. In Proceedings of the 5th ACM Conference on Designing Interactive Systems - DIS 2004 (Cambridge, Massachusetts, USA). ACM Press, New York, NY, USA, 167-175.

Li, Y., Hong, , J. I., and Landay, J. A. 2007. Design Challenges and Principles for Wizard of Oz Testing of Location-Enhanced Applications. IEEE Pervasive Computing 6, 2, 70-75.

Liu, L. and Khosshabeh, P. 2003. Paper or Interactive?: A Study of Prototyping Techniques for Ubiquitous Computing Environments. In Extended Abstracts of the 21st ACM Conference on Human Factors in Computing Systems - CHI 2003 (Fort Lauderdale, Florida, USA). ACM Press, New York, NY, USA, 1030-1031.

Marquardt, N. 2008. Developer Toolkit and Utilities for Rapidly Prototyping Distributed Physical User Interface. Media Systems Science, Bauhaus University Weimar.

Michelson, B. 2006. Event-Driven Architecture Overview. Feb 2. Patricia Seybold Group.

Myers, B. A., Hudson, S. E., and Pausch, R. 2000. Past, Present, and Future of User Interface Software Tools. ACM Transactions on Computer-Human Interaction 7, 1, 3-28.

Nicolai Marquardt 2008.Developer Toolkit and Utilities for Rapidly Prototyping Distributed Physical user Interfaces

Nielsen, J. 1993. Usability Engineering. Morgan Kaufmann Publishers.

Norman, D. A. 1988. The Psychology of Everyday Things. Basic Books, New York, NY, USA.

Norman, D. A. 1999. The Invisible Computer: Why Good Products Can Fail, the Personal Computer Is So Complex, and Information Appliances Are The Solution. MIT Press.

O'Reilly, T. 2005. What Is Web 2.0. O'Reilly Network. Web-site last visited on 2009-08-06.

OpenMeetings 2012. http://en.wikipedia.org/wiki/Openmeetings 2012

Phidgets Inc. 2009. Phidgets Product Website. Http://www.phidgets.com. Web-site last visited on October 2, 2009.

Pietzuch, P.R., Bacon, J.M.: Hermes 2002: A Distributed Event-Based Middleware Architecture. In: Proc. of the 1st Int. Workshop on Distributed Event-Based Systems (DEBS'02),Vienna, Austria (2002) 611-618.

Pinson, L. & Wiener,R. 1998 An Introduction to Object-Oriented Programming and Smalltalk, Addison-Wesley, 1988, 336-358.

Preece, J., Rogers, Y., and Sharp, H. 2002. Interaction Design. John Wiley & Sons, Inc., New York, NY, USA.

Red5. 2009 Open Source Flash Server http://osflash.org/red5. Website last visited on September 20, 2009

Regan, B. 2005 Macromedia White Paper, Best Practices for Accessible Flash Design. http://www.adobe.com/resources/accessibility/best_practices/best_practices_acc_flash.pdf Webiste last visited on Augest 20, 2009

Roseman, M. 1993. Design of a Real-Time Groupware Toolkit. M.S. thesis, University of Calgary, Department of Computer Science.

Rouached M., Perrin O., and Godart C. 2006. Towards formal verification of web service composition. In Proceedings of the International Conference on Business Process Management, 2006.

Rudd, J., Stern, K., and Isensee, S. 1996. Low vs. High-Fidelity Prototyping Debate. ACM interactions 3, 1, 76-85.

Santanche, A., Nath, S., Liu, J., Priyantha, B., and Zhao, F. 2006. SenseWeb: Browsing the Physical World in Real Time. In Proceedings of the Fifth ACM/IEEE International Conference on Information Processing in Sensor Networks - IPSN 2006. Nashville, TN.

Vasekar, S. and Rimov, M. 2004. Expresso Developer's Guide. http://www.jcorporate.com/expresso/doc/edg/edg_WhatIsMVC.html: Jcorporate.

Security Space Tech. 2007. Security Space technology Penetration reports. http://www.securityspace.com/s_survey/data/man.200703/techpen.html. Website last visited on May 20, 2009

Shneiderman, B. 1996. The Eyes Have It: A Task by Data Type Taxonomy for Information Visualizations. In Proceedings of IEEE Symposium on Visual Languages. 336-343.

Sohn, T. and Dey, A. 2003. iCAP: An Informal Tool for Interactive Prototyping of Context-Aware Applications. In Extended Abstracts of the 21st ACM Conference on Human Factors in Computing Systems - CHI 2003 (Fort Lauderdale, Florida, USA). ACM Press, New York, NY, USA, 974-975.

Sunsted, T. 1998 MVC meets Swing, JavaWorld, April 1998, Online: http:// www.javaworld.com/javaworld/jw-04-1998/jw-04-howto.html Website last visited on June 16 2009

Ullmer, B., and Ishii, H. 1997 The metaDESK: Models and Prototypes for Tangible User Interfaces. In Proceedings of UIST ¡¯97 pp. 223-232

Ullmer, B. and Ishii, H. 2000. Emerging Frameworks for Tangible User Interfaces. IBM Systems Journal 39, 3-4, 915-931.

Vaucelle C., Ishii H. 2005 Interfacing Video Capture, Editing and Publication in a Tangible Environment. MIT Media Laboratory, Tangible Media Group

Vidyo Inc. 2005 http://en.wikipedia.org/wiki/Vidyo 2012

Villar, N. and GellersenELLERSEN, H. 2007. A Malleable Control Structure for Softwired User Interfaces. In Proceedings of the 1st International Conference on Tangible and Embedded Interaction - TEI 2007 (Baton Rouge, LA, USA). ACM Press, New York, NY, USA, 49-56.

WebEX. 2009, http://www.webex.com/partners/webex-onnect.html Website late reviewed on Dec 9th, 2009

Weiser, M. 1991. The Computer for the 21st Century. Scientific American 265, 3 (September), 66-75

Weiser, M. and Brown, J. S. 1997 The coming age of Calm Technology. Beyond Calculation: The Next Fifty Years 1, 75-85

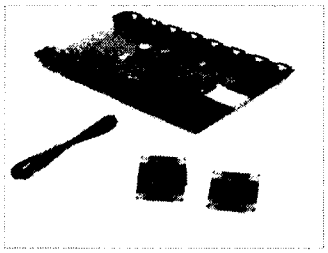
Want, R., Fishkin, K. P., Gujar, A., and Harrison, B. L. 1999. Bridging Physical and Virtual Worlds with Electronic Tags. In Proceedings of the ACM Conference on Human Factors in Computing Systems - CHI 1999 (Pittsburgh, Pennsylvania, USA). ACM Press, New York, NY, USA, 370-377.

Appendix A: Implemented Hardware Devices

For the events generated from Phidgets widgets, the devices present event flow from physical widgets to software application with the proposed architecture, discussed in previous chapters. The first input device (Table 5) is Interface Tool Kits with force sensor and push sensor. These tools demonstrate slides control functions. This example illustrates that the architecture allows building devices other than the traditional keyboard and mouse.

InterfaceKit and Sensors Hardware

Implementation of the Phidget Interface Kit: this generic I/O board has multiple digital inputs and outputs, as well as analog inputs (for sensors).



Functions:
The force/touch sensor is used to switch next/previous slide.

API Event:

```
43  private function onSensorChange(evt:PhidgetDataEvent):void{
44      switch(evt.Index){
45          case 2:
46          if (evt.Data == 0) {
47              // gotoNextSlide();
48              var nextEvent:presentEvent = new presentEvent(presentEvent.NEXT_SLIDE);
49              var goNext:Dispatcher = new Dispatcher();
50              goNext.dispatchEvent(nextEvent);
51          }
52          break;
53          case 5:
54          if (evt.Data == 0) {
55              // gotoPreviousSlide();
56              var prevEvent:presentEvent = new presentEvent(presentEvent.PREVIOUS_SLIDE);
57              var goLast:Dispat org.bigbluebutton.modules.presentation.events.presentEvent
58              goLast.dispatchEvent(prevEvent);
59          }
60          break;
61      }
62  }
```

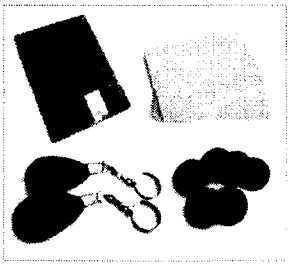
Table 5 InterfaceKit and Sensors Hardware and API

The second input device (Table 6) is RFID reader and tags. It illustrates the action

from generate operator and can be passed into web conferencing with event driven architecture, which specifies that an event is to be generated only when the corresponding variable reaches a threshold value.

RFID Reader Hardware

Implementation of the Phidget RFID Reader: the reader can identify RFID tags up to a distance of 10-15 cm.
All RFID tags have a unique 16 digit hexadecimal identification. The reader has no collision detection; therefore, only one RFID tag can be identified at the same time.



Functions:
Automatically login function without input username/password.

API Event:

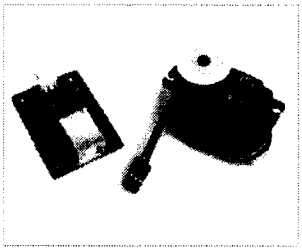
```
149     private function onTag(evt:PhidgetDataEvent):void{
150         trace(evt);
151         rfid.LED = true;
152         switch(evt.Data){
153             case "010775a8cc":
154                 var rfidLoginEvent:RFIDEvent = new RFIDEvent(RFID.LOGIN);
155                 var rfidLoginEvent:Dispatcher = new Dispatcher();
156                 rfidLoginEvent.dispatchEvent(rfidLoginEvent);
157             break;
158         }
159     }
160
161     private function onTagLoss(evt:PhidgetDataEvent):void{
162             trace(evt);
163             rfid.LED = false;
164             rfid.close();
165     }
```

Table 6 RFID Reader Hardware and API

On the other hand, the Phidgets module can also handle web conference output to the physical device. It illustrates the Phigets widgets (e.g., Servo) can respond to the event that came from the web conferencing system. The physical devices (Table 7) can be used to emphasizes and monitor the events, which happened in the web conference system.

Servo Hardware

Implementation of the Phidget Servo: the position of the servo motors can be set to any value between 0 and 180 degrees. Two versions of the servo controller exists: a single servo controller, and a controller for up to four servo motors.



Functions:

To demonstrate the hands up/down events from web conference.

API Event:

```
254
255             if (status == 'raiseHand') {
256                 if (value == true) {
257                     hands = hands + 1;
258                 }   else {
259                     hands = hands - 1;
260                 }
261
262                 if (hands == 0) {
263                     var handsDownEvent:ServoEvent = new ServoEvent(ServoEvent.HANDS_DOWN);
264                     var handsDown:Dispatcher = new Dispatcher();
265                     handsDown.dispatchEvent(handsDownEvent);
266
267                 } else if (hands > 0) {
268                     var handsUpEvent:ServoEvent = new ServoEvent(ServoEvent.HANDS_UP);
269                     var handsUp:Dispatcher = new Dispatcher();
270                     handsUp.dispatchEvent(handsUpEvent);
271                 }
272             }
273
274         }
275
```

Table 7 Servo Hardware and API

Appendix B: Discuss Uses of Phidgets

Introduction

The following contexts for using Phidgets toolkits are the result of brainstorming and discussion of project group members as well as contributions from the community of BigBlueButton. (Wiki of BigBlueButton 2009)

Details

Colored LED Light

1. To give a feedback mechanism for remote users to indicate their level of confusion
2. To notify moderator or in-class users in a hybrid class room (with both local and remote students) as remote users join or drop.
3. To provide visibility to in-class users of the remote users, and help integrate both groups.
4. To notify an administrator of the status of a BigBlueButton server.

Servo Motor

1. Raise Hand function when user has the question to ask.
2. To show the users availability in web conference
3. To notify an administrator of the status of a BigBlueButton server.

Sensors

1. To change page for presenter during the presentation.
2. To give the notice when end user has the question to ask.
3. To mute/unmunt the participants by just a push and donot need to click every participant one by one.
4. To track movement of a presenter in a room.

RFID and reader

1. User login function, including logout automatically if the user is away from computer
2. To help the vote during the discussion in conferencing because the user ID is unique.
3. To detect the shadow of the presenter using the radio range of RFID reader, for example, if the presenter move out of whiteboard and RFID lost the tagged, then give him/her a notice

Linear Touch Sensor:

1. To zoom in/out the presentation window. Also can be used to move and focus the context in slider in screen, like what the mouse does.
2. To adjust the volume of speaker and microphone.