Supporting the Development of Electrophysiologically Interactive Computer Systems

Jennifer Allanson

Submitted for the degree of Doctor of Philosophy

Computing Department
Lancaster University
UK

June 2000

Abstract

Supporting the Development of Electrophysiologically Interactive Computer Systems

Jennifer Allanson

Submitted for the degree of Doctor of Philosophy
Computing Department, Lancaster University, UK
June 2000

New interactive computing applications are continually being developed in a bid to support people's changing work and recreational activities. As research focuses on one particular class of interactive systems, high level models of interaction are formulated and requirements emerge that reflect shared features or common functionality among those systems. The existence of models of interaction and shared functional requirements mean that support tools can be created which ease the subsequent development of these systems. Support tools most often take the form of architectures or frameworks that describe how a system should be structured. The type of tool that interactive systems developers are most familiar with is a library of reusable code that can be used for prototyping and building interactive applications and their interfaces.

Within this thesis a new class of interactive system is identified, based on shared requirements for detection, processing and presentation of human physiological information. We have named these systems *electro*physiologically interactive computer systems (EPICS) and describe in this thesis both the physiological and technological details behind their operation. A review is presented of existing research and development into this exciting new area of human-computer interaction, the aim being to establish the common requirements. These have enabled us to develop a suite of software components to support the creation of future EPIC systems. It is envisaged that the work presented in this thesis will serve as a jumping off point for others interested in exploring the potential of incorporating physiological information into the human-machine relationship.

Acknowledgements

This work would not exist were it not for the encouragement and support of Ian Sommerville and the Computing Department at Lancaster University. Thanks are due to my supervisor John Mariani for letting me do my thing and to both he and Tom Rodden for encouraging and constructive comments on this thesis and the publications associated with it.

I’ve been lucky enough to undertake this work in an environment that is alive with good humour and a passion for research. Thanks are due to too many exceptional colleagues to list, but special thanks must go to Simon Lock and Peter Phillips.

I have also received the best support in the world from my family. Love especially to my sister Jacqui and to Chris, Holly and Adam. Deep and everlasting respect to the person who always told me I was capable of doing this - my husband, David.

For Mum and Dad

Table of Contents

List of Figures

List of Tables

Chapter 1

Introduction

1.1 The Lack of Support for New Modes of Interaction

The development of traditional windows-style interactive applications is supported by a plethora of software tools consisting of common interaction techniques and supporting common input devices. Unfortunately, existing interactive system development support tools have been created with electromechanical device-based interaction in mind and cannot be easily extended to accommodate other, modally disparate models of interaction. As new models of human-machine interaction emerge, so new tools are required to support the development of systems adhering to these models. One emerging paradigm in human-machine interaction is based on the electronic detection and interpretation of human physiological information. What these *electro*physiologically interactive applications share is a prerequisite to detect, process and present human physiological[1] information within both traditional windows-style interfaces and more novel, non-windows style interfaces.

Electrophysiological human-machine interaction requires, in the first instance, access to specialised sensing devices that detect human physiological information. Although physiological sensing devices that serve as peripherals to personal computers are available these devices have often been designed with specific medical monitoring applications in mind. This is reflected in application software that is task-specific and closed to further development. At present, each new application conceived with a view to exploring some aspect of physiologically-enabled human-computer interaction needs to be built entirely from scratch, despite the existence of many common signal pre-processing and presentation requirements across such applications.

This thesis seeks to address the current lack of support for developing electrophysiologically interactive computer systems. In order to achieve this, it identifies and describes a new model of human-machine interaction that is based on the sensing and preprocessing of human physiological information. The role of the thesis is to introduce this new class of interactive systems and thereafter to describe a software toolkit designed to support the future development of these systems. The motivation behind this work is to encourage a wider exploration of the potential utilization of human physiological information within the human-machine relationship.

[1] The terms physiological and electrophysiological are used interchangeably throughout this thesis

In this chapter, an introduction to this new paradigm of human-machine interaction is presented. A statement of the objectives and scope of this thesis, as well as a description of the novel contributions of this work follow. The chapter concludes with an overview of the structure of this thesis.

1.2 Limitations of Electromechanical Human-machine Interaction

For more than fifty years people have interacted with computers through mechanical input devices external to both themselves and the computer (Palfreman 1991). From switches and punched cards through to the ubiquitous arrangement of keyboard and mouse, mechanical interaction devices remain the most common means of supporting human-to-machine communication in interactive computer systems.

Mechanical control, where a user physically manipulates an electromechanical device to initiate a computer operation, requires the periodic dedication of one or both hands. Unfortunately, many people work in environments where their hands are already fully occupied with other physical tasks. Examples include surgeons, fitters and maintenance engineers, aircraft flight crew and drivers of heavy goods, passenger and private vehicles. In all of these situations, access to information or non-essential operation of electronic devices would perhaps be better served by alternative, hands-free access control.

A further limitation of existing mechanical models of interaction is that they exclude access to those individuals for whom normal physical control is either difficult or impossible. The needs of physically disabled users are rarely considered during the design of new computer systems (Bergman 1995), and yet these individuals constitute the user group most likely to reap the benefits of computer-based interactive technologies. Add to this group an ageing populace, with accompanying restrictions on physical abilities, and it is clear that supplemental methods of human-machine interaction to those currently available are needed.

1.3 Electrophysiological Human-machine Interaction

Investigations into hands-free alternatives to electromechanical control have to date focussed on voice- and gesture-recognition technologies. An alternative method of hands-free of human-computer interaction currently under investigation is based on the detection of consciously controllable human physiological information. This physiological information can be processed electrically (hence we talk about *electro*physiological signals) and thereafter transformed into computer control signals and commands.

Using specialised electronic sensing equipment it is possible to both detect and make available in a digital format information pertaining to a wide range of human physiology. Physiological sensing equipment can be used in order to train individuals to gain conscious control over a range of physiological parameters including heart rate, muscle tension and brain activity. The process of making physiological information available to a subject who is being trained to control some aspect of his or her own physiology is known as *biofeedback*. After training, physiological signals can be applied to hands-free human-machine control. The most exciting results of biofeedback-based hands-free control research so far focus on the utilization of electrical brain activity to directly control a cursor on a computer screen (Wolpaw 1990) and to input alphanumeric characters using a soft keyboard (Kubler 1999).

1.4 Monitoring System Operator State

The prospects for incorporation of physiological information into future forms of human-machine interaction go beyond the potential for hands-free control, however. People play an integral role in the operation of many complex and potentially dangerous systems. Flight crew, nuclear installation personnel, operators of heavy machinery, even car drivers are all placed in situations where a complex system is reliant upon them for instruction which enables that system to maintain safe operation. But what happens if the operator experiences a physiological condition that is detrimental to their continued safe operation of that system? Examples of such conditions range from a loss of the operator’s concentration all the way through to their total loss of consciousness.

Many operator-reliant systems incorporate auto pilot functionality that can allow them to maintain a stable condition in the absence of the human operator. However, how is the system to know, unless explicitly told, that it needs to take over its own operation? If it were possible to provide a system with information regarding the physiological state of its human operator then that system would be in a better position to take over its own control if something happened to its operator. By using the same physiological signal detection technologies that are used in biofeedback training it is possible to provide information which allows a system to monitor the changing physiological condition of it's human operator.

The kinds of technologies required to explore both electrophysiological hands-free human-machine interaction and complex system operator monitoring are commercially available, retailing for around the price of a standard PC. They take the form of dedicated electronic sensing devices that can detect human physiological information and make that information available to a computer for processing and presentation. At the present time each developer wishing to build an interactive application which incorporates human electrophysiological information needs to build that application from scratch as no existing development tools provide support for these physiological sensing devices.

1.5 The Need for Development Support

The benefits of using software tools to develop interactive applications are threefold:

Firstly interaction techniques provided by software toolkits provide an abstraction over computer-input peripherals. This enables the building of applications that will run with any suitable configuration of *virtual* interaction devices. So for example, a windows-style application which relies on keyboard and pointing device-based interaction can be used with most flavours of keyboard (i.e. QUERTY, Dvorak, Chord, etc.) and any one of a number of similar pointing devices (mouse, trackball, trackpad, joystick etc.).

The second benefit is that software toolkits save development time by providing commonly used code fragments that can be incorporated into the final system implementation. These same reusable code fragments are also often used to create prototype systems, which can be presented to potential users for evaluation during system development.

The third benefit of software development tools is that they promote standardization by providing recognised ways of presenting information and carrying out tasks. Standardization leads to the creation of systems with features and functionality familiar to users of other, similar systems. This in turn increases the potential usability of newly created interactive systems.

In order to exploit the potential for incorporating electrophysiological signals into the human-machine relationship, access is required to software tools containing suitable interaction techniques that are responsive to human electrophysiological signals. In order to support development of applications such as those described above development tools are required which provide:

- ❑ A means to quickly prototype systems that incorporate various combinations of physiological signals for evaluation.

- ❑ A configuration that enables developers to easily add to the existing signal processing and presentation software capabilities and thus create new interaction techniques that meet the requirements of their particular application.

- ❑ The flexibility to exploit as full a range of media components as possible, for both monitoring and training of human physiological functionality.

One major consideration in the creation of such software tools centres around the fact that electrophysiological human-machine interaction is, as we will show, a multidisciplinary field involving input from psychologists, psychophysiologists, electronic engineers and software engineers. So an important criteria for any development tools is that non-programmers must be able to use them in order to prototype electrophysiologically responsive interfaces. The final criteria which is fulfilled by existing interface development tools and which is equally desirable in an electrophysiological systems toolkit is the notion of device independence. This implies:

- ❑ The ability to deploy the interaction techniques of the toolkit over any one of many similar electronic physiological sensing devices.

1.6 Aims and Objectives

The main aim of the work presented here has been to develop interaction techniques to support the creation of interactive systems that incorporate electrophysiological information and to do this in such a manner as to provide an abstraction over multiple devices that sense human physiology.

This can be broken down into a number of objectives:

- ❑ Identify the requirements for building interactive systems that incorporate physiological sensing technologies.
- ❑ Identify a suitable model of interaction for systems based on sensing.
- ❑ Based on the satisfaction of the first two objectives, design and implement a device-independent software toolkit suitable for building interactive systems that incorporate human physiological information.

In support of this endeavour, this thesis draws together work from a diverse range of disciplines whose common interest is the incorporation of electrophysiological information into the human-machine relationship. It provides this review in order to highlight shared requirements in terms of detection and presentation of physiological information. Identifying common functionality among existing systems is only one aspect of the work required in order to produce suitable development tools. More importantly a high level conceptual model of interaction needs to be identified which will be reflected not only in the toolkit produced as part of this work, but also in other toolkits that may emerge to support the future development of this new class of interactive application.

1.7 Scope of this Work

As mentioned above, electrophysiological human-machine interaction is a multidisciplinary field. In order to set the toolkit produced as part of this work in context it is helpful to look at a range of related concerns. Firstly, some discussion of human physiology is necessary in order for the reader to both appreciate the sources of detectable physiological information and follow discussions of electrophysiologically enabled interactive computer systems. Detectable human physiology is presented in Chapter 2 and this chapter serves as a reference for the *electrophysiologically interactive computer systems* (EPICS) review presented in Chapter 3.

To support discussion of electrophysiological signal-enabled hands-free control systems, *biofeedback training* is described as being the method by which an individual learns to gain conscious control of some aspect of their physiological functionality. Biofeedback training is discussed within the context of its current most popular application, which is as a clinical treatment for a range of psychophysiological disorders. This discussion serves to support the argument for both flexibility in design of development support tools and the need to create development tools that are accessible to and useable by non-programmers.

Finally, it should be noted that the existing toolkit, which consists of a suite of Java components, is not intended to provide an extensive set of interface components. Rather, the aim of the existing toolkit is to demonstrate the applicability of the chosen conceptual model of interaction to the design of modular tools for creating electrophysiologically focussed applications. The description given in Chapter 6 of the toolkit being used for systems development serves to illustrate the flexibility gained thorough choosing to implement the toolkit using a standard development technology.

1.8 Novel Contributions of this Work

The novel contributions of this work are as follows:

- *Identification of a new class of interactive systems*

  Chapter 3 contains a unique, multidisciplinary review of interactive computer systems that incorporate electrophysiological information. Based on common requirements of detection, processing and presentation of human physiological information, these systems are collectively identified herein as being *electrophysiologically interactive computer systems* (EPICS).

- *Identification of a new model of interaction based on sensing*

  EPIC systems are an identifiable subgroup of the group of all systems whose interactions are based on sensing the actions of the user (i.e. voice- and gesture-recognition, motion detection, etc). A high-level conceptual model of interaction based on sensing is identified and used to reason about the future development of EPIC systems.

- *A component-based toolkit for building systems which adhere to this new model of interaction*

  The toolkit developed as part of this work provides the first available support for the development of systems that incorporate human electrophysiological information. The toolkit consists of a suite of software components that can be used for both rapid prototyping and construction of EPICS, with the option for either traditional windows-style or non-traditional interactive interfaces.

1.9 Structure of this Thesis

Chapter 2 examines the range of human physiological information that can be detected and made available to a computer using electronic sensing equipment. This chapter examines the requirements for physiological signal detection, processing and presentation and includes a generic description of the electronic hardware required in order to make physiological information available to a computer. Chapter 2 concludes with a description one of the most widespread applications for physiological signal detection and presentation, namely biofeedback training.

Chapter 3 examines a range of applications that rely on the measurement and monitoring of physiological information as well as its integration into the human-machine relationship. This review includes an overview of systems designed specifically for exploring alternative forms of hands-free human-computer interaction based on consciously controllable physiological signals. Chapter 3 also describes systems designed to monitor the changing physiological condition of system users. This information can be used to identify times when a system's human operator experiences psychological and physiological states that may be detrimental to their continued safe operation of that system. Physiological monitoring will also be shown to provide useful information as part of the wider software evaluation process. Also described in Chapter 3 are so-called *Affective Computing* applications. Affective Computing is a growing field of research concerned with developing machines that recognise and respond to the changing emotional state of a user, based on the monitoring and processing of both physiological and behavioural information. The overall aim of this review chapter is to identify shared features and functionality that need to be provided by an EPICS toolkit.

In order to support the high level design of an EPICS toolkit, a review of development support tools for more conventional interactive systems is provided in Chapter 4. Issues surrounding user-system interaction are discussed with the focus being on the system's view of the user through the types of user-generated event allowable within a certain class of system. So for example, discrete user-generated events, allowable through interaction techniques designed for use with popular electromechanical input devices, are considered within the context of WIMP (windows, icons, menus, pointers) interaction. The focus moves on to tools which have been created to support the development of multimodal applications, in particular VR toolkits, which allow richer forms of human-machine interaction through the use of continuous input devices. Throughout Chapter 4, the relationship between conceptual models of interaction and toolkit design is highlighted. This chapter concludes with identification of a conceptual model of interaction suitable for systems that sense the user. Electrophysiologically interactive systems are a subset of sensing systems and therefore this model describes the interaction that occurs within EPIC systems.

Chapter 5 presents the design and implementation of a component-based software toolkit conceived to support the development of selectrophysiologically interactive computer systems. Following on from this Chapter 6 describes how the toolkit components can be composed using freely available builder tools. This chapter also contains a description of the toolkit being used to rapidly prototype and implement an interface for a biofeedback application used in a study carried out by the Psychology Department at Lancaster University.

Finally, Chapter 7 discusses how the aims and objectives laid out in this introductory chapter have been met by the work described in the thesis. A discussion is presented of potential extensions to this work and Chapter 7 concludes by speculating on both the implications of the work presented by this thesis and the future prospects for electrophysiological information within the human-machine relationship.

Chapter 2

Detectable Human Physiology

2.1 Introduction

This chapter provides an introduction to the detection of human physiology. Both the terminology and technical issues behind physiological signal detection and presentation via a personal computer are discussed here. The purpose of this chapter is to support the review of EPIC systems presented in Chapter 3 and as such it can be used simply as reference material for that chapter[2]. However, the information contained herein serves as a concise introduction to the issues surrounding detectable human physiology, some understanding of which is necessary for the development of electrophysiologically interactive computer systems.

The first section in this chapter describes methods of gathering physiological information and focuses primarily on the generic structure of electronic sensing equipment for detecting and pre-processing this information. The second section describes some of the physiological signals detectable from the surface of the human body. The use of electronic sensing devices means that these physiological signals can be made available to a computer in a digital format and thereafter presented within an application suitable for measurement, monitoring or biofeedback-based training purposes. The final section of this chapter describes the use of sensing equipment and computer technology for biofeedback training, highlighting in particular the role that interactive multimedia systems interfaces play in modern clinical biofeedback applications.

2.2 Physiological Signal Sensing Systems

The human body is a biological system that is chemical, electrical, mechanical, thermal and magnetic in nature. The body receives information about objects in its environment via sensory apparatus tuned to receive data of each type. Smell and taste are chemical, sound is mechanical, touch is both mechanical and thermal, and sight is electromagnetic. Information received from the body's physical environment is translated by the sensory organs into electrochemical signals that are transmitted to the brain via the central nervous system (CNS). The brain controls the body's responses to its environment by sending electrochemical signals via the CNS to control the skeletal muscles and via the autonomic nervous system (ANS) to regulate the organs of the body.

[2] In order to aid referential usage, suitable pointers to the contents of this chapter are provided throughout Chapter 3

Computers, like humans, have sensory organs with which to detect their environment. A computer's sensory organs correspond to the input devices it supports and these are commonly simple mechanical devices such as pointers and keyboards. Developments in interactive multimedia applications such as virtual reality have extended the computer's sensing abilities to include increasingly complex devices that can be configured to detect information of all the same types as can be detected by human sensory organs. Examples include microphones for detecting sound, video processing for visual input as well as electromechanical motion detection equipment.

On the one hand, computer sensory capabilities appear to be quite rudimentary when compared with human sensory capabilities. On the other hand computer-aided sensing technologies make it possible for computers to detect and display information related to human physiology otherwise unavailable via normal human sensory channels. Fir example, computer-based imaging such as X-ray tomography (also known as CAT), positron emission tomography (PET) and magnetic resonance imaging (MRI) all provide non-invasive ways of looking inside the working body (Posner 1997). These technologies have all been designed with medical applications in mind and are helping to answer questions about human physiological functioning. Another means of gathering otherwise invisible physiological information is through the use of electronic sensing technologies, which have continuous detection capabilities. These can be applied to the surface of the body and make it possible for a computer to detect a body's physiological responses to its environment. As we will see in later chapters, these detectable physiological responses provide information that can be used to enhance the human-machine interactive relationship.

2.2.1 Early Physiological Signal Detection

The earliest means of detecting physiological information from the body was direct observation. Human sensory organs were used directly to detect the external manifestations of internal physiological functioning. For example, an ear placed on the chest make it possible to hear the rhythmical beating of the heart. Alternately, fingertips placed on the skin over a pulse-point make it possible to feel the periodic blood surges caused by cardiac valves opening and closing, pumping blood through the heart. However, most physiological information is too subtle to be detected by human sense organs alone. This has led to the development of specialised devices that amplify physiological information and transform it, where necessary, into a form more suitable for human observation.

In 1924 Dutch physiologist Willem Eithoven won the Nobel Prize in physiology for inventing the galvanometer which he used to detect and display electrical information generated by the heart. The galvanometer consisted of a thin wire suspended between the poles of a powerful electromagnet. Sensors, places on the chest of a subject, detected the tiny fluctuations in electrical potential that accompanied each heartbeat. As this electrical activity passed through the wire its fluctuating electrical current generated tiny magnetic fields. These magnetic fields caused the wire to bend with a displacement proportional to the voltage of the detected signal. By attaching a mirror to the wire, it is possible to reflect light onto a rolling photographic paper and in this manner capture the fluctuating voltage as a polygraphic trace on the paper (Figure 2.1). The so-called *mirror galvanometer* made it possible to display the electrical signals detected by Eithoven's device via a mechanical output device (the paper roller).

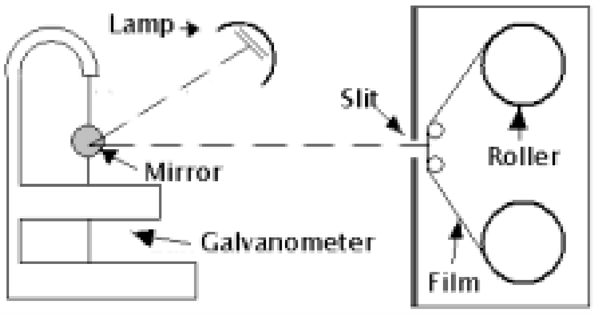


Figure 2.1 A mirror galvanometer

The galvanometer enabled Eithoven to display, for the first time, electrical signals generated by the human body. Probably the first all-electronic device designed for sensing and displaying human physiology was the *toposcope* developed in the 1920's at the Burden Neurological Institute in Bristol, England. Designed to detect brain activity from 24 separate scalp-mounted sensors, the toposcope displayed 22 of those channels on 22 separate CRT displays (the two remaining channels were connected to a pen and paper output device to enable permanent records of brain signals to be made).

2.2.2 Electronic Sensing Devices

The role of an electrophysiological sensing device is to detect small electrical signals generated by human organs and skeletal muscles. The small electrical signals detected through skin-mounted electrodes have to be processed in such a way as to make them suitable for presentation (usually via a computer display). The three stages of physiological signal processing are illustrated in Figure 2.2.

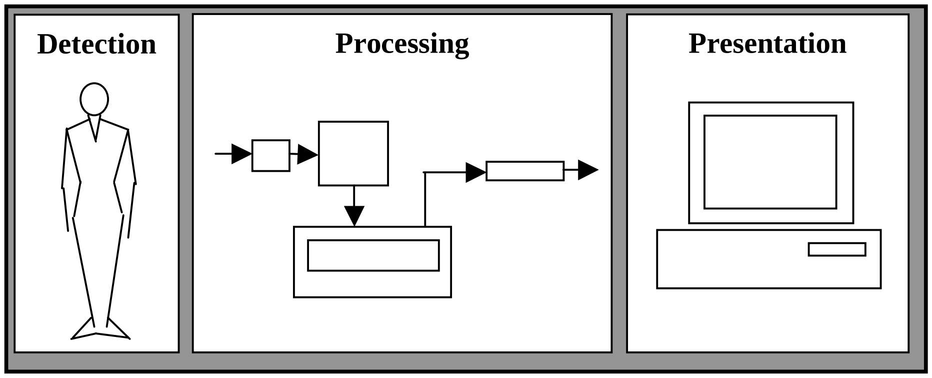


Figure 2.2 The three stages of physiological signal recording

2.2.2.1 Signal Detection

Detection of electrophysiological signals involves the application of surface electrodes[3] to the skin using an electrically conductive paste or gel. Electrodes are small metallic discs that are attached by wires to electronic sensing hardware. They are often cupped in shape in order to hold a suitable amount of paste or gel.

Different sensing applications require different numbers of electrodes, with the minimum requirement being two – one *active* and one *reference* electrode. Active electrodes are used to collect physiological data from appropriate sites on the body. At least one *reference* electrode is also required to collect data from a neutral site on the body for comparison with the active electrode signals. The information collected from the reference electrode is used to filter out

[3]As opposed to subcutaneous /implanted electrodes

electrical noise that can be caused by other electrical equipment and/or other physiological signals beside those being purposefully detected.

2.2.2.2 Signal Processing

Processing of detected electrophysiological signals is also a three-stage process, involving *amplification*, *digitisation* and *filtering* of physiological data (Figure 2.3). The strength of electrical signals gathered through skin-mounted electrodes is of the order of microvolts and must therefore be carefully amplified before being presented. Electrical noise in the surrounding environment can contaminate such a small signal. In order to avoid amplifying this electrical noise, a differential amplifier is employed which amplifies the difference between two signal points (one active electrode and one reference electrode). As electrical noise affects all signals equally, it is filtered out. Power sources close to the sensing equipment are another source of electrical noise. A band-pass filter must be included in physiological sensing set-up in order to remove mains-hum[4] from the detected signal.

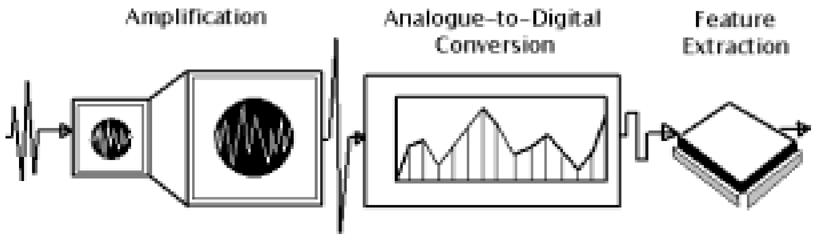


Figure 2.3 The three stages of signal processing

Once amplified, the signal can be sampled using an analogue-to-digital converter and the resulting digital data stream passed to a digital signal processor (DSP). The DSP extracts features and events from the digital stream by applying Fast Fourier Transforms (FFTs) to the data. Finally the digital stream can be presented to a digital computer. This presentation must be achieved through an optical coupling between the sensing device and the computer, due to the potentially dangerous electrical connection that would otherwise exist between a subject and the mains via the computer.

[4] 50 Hz in US, 60 Hz in UK

2.2.2.3 Signal Presentation

A stream of detected electrophysiological signals has traditionally been presented as a real-time polygraphic trace. This method of representation is an artefact of paper-based display techniques such as that employed by the galvanometer described previously. The use of polygraphic display mechanisms can be seen to have influenced the names given to dedicated physiological signal sensing devices – for example, an electroencephalo*graph* is a device that detects brain signals, an electromyo*graph* detects muscle signals and an electrocardio*graph* detects heart signals.

If a physiological signal presentation application requires measurement of some discrete physiological event, a graduated scale or meter is an appropriate method of presenting the signal. Stand-alone commercial sensing devices such as the *GSR2*[5] skin resistance monitor use analogue signal display meters for measuring signal magnitude. Self-contained electronic units such as the *EMG Retrainer* incorporate digital displays on which they can present signal information either graphically or numerically. Other devices like the *MyoTrac* use LED bars and/ or musical tones to indicate the current magnitude of detected physiological signals.

If physiological information is to be presented to a subject for biofeedback training purposes, devices with limited display capabilities are not always appropriate. This is because biofeedback training of some physiological signals can take hours. In order to keep a subject's interest display technologies are required that offer greater signal presentation flexibility than stand-alone sensing devices. It is for this reason that modern integrated sensing technologies are designed to work in conjunction with personal computers.

2.2.3 Computer-based Sensing Systems

Electronic sensing devices coupled to computers are known as physiological sensing *systems*. The sensing device fulfils the detection and hardware-based processing role outlined previously, but rather than utilising some dedicated display technology it passes a fairly raw digital data stream to a computer for both further processing and eventual display. There are a number of advantages to

[5] See Appendix A for manufacturer details of physiological sensing hardware discussed in this thesis

using computer-based systems instead of stand-alone sensing devices. Beside better display capabilities, computer systems provide:

- The ability to digitally archive huge amounts of physiological information. Hassett (Hassett 1978) estimated that sensing devices reliant on paper-based archiving could use over half a mile of polygraphic paper in order to record the EEG material collected from a single night's sleep study.
- The ability to take multiple copies of the physiological data collected during an observation period.
- Access to software tools which support on- and off-line data analysis.
- The ability to play back data during off-line observational analysis.

Furthermore, the multimedia interface capabilities of personal computers allow biofeedback practitioners to present physiological information back to subjects in a variety of audio/ visual formats.

Computer-based sensing devices provide ideal development platforms for emerging electrophysiologically focussed interactive applications. This will be discussed fully in the following chapter, but first it is necessary to be familiar with the types of physiological information that can be detected from the surface of the body using modern sensing technologies.

2.2.4 Discussion

The creation of physiological signal detection equipment has to date been influenced by mainly medical applications. As these applications recognize the utility of computer technology for further signal processing, archiving and display of physiological information, so the sensing devices themselves have evolved to become computer peripherals. Indeed modern integrated physiological sensing devices are designed to be plugged into the serial port of a computer and to feed their digital data streams on to a suitable application.

The detection capabilities of these sensing technologies serve to define the new mode of human-machine interaction explored by this thesis. This definition is based on the conversion of human physiological information into electrical pulses. A stream of these electrical pulses provides a computer with an alternative source of information about its user. We will see in the following chapters that there are several things that a computer can glean from this unusual user input source. But before exploring the view that a computer might get of the user through his or her physiology, it is worth examining the human perspective on human physiology.

2.3 Electrophysiological Signals

For over 70 years psychophysiologists have been studying physiological signals in a bid to understand the body's complex physiological responses to different psychological conditions such as stress and depression (Shagass 1972)[6]. By looking not only at the physiological signals they study, but also at the knowledge associated with different physiological phenomena, a better sense can be gained of the types of applications to which physiological information can be put in future human-machine interactions.

In order to explore the potential of using human physiological information as an input source to a computer, it is necessary to look at what kind of physiological information can be made available to the computer. There are two types of physiological information that can be detected from the surface of the body. The first is electrical information that occurs as a natural consequence of an individual interacting with their environment. This electrical information can be detected using equipment of the nature described above. The second is non-electrical information, such as temperature or blood pressure that can be detected using the similar electronic equipment but with the addition of specialist sensors. Once detected this information can be processed electronically and presented to a computer in a digital data format.

2.3.1 Electrical Brain Activity

By attaching electrodes to the surface of the scalp it is possible to detect electrical signals emanating from the neo-cortex or surface of the brain. The electrical signals detected by scalp-mounted electrodes correspond to two distinct forms of activity within the brain.

[6] Taken from The Handbook of Psychophysiology which despite its age remains one of the definitive texts in its field

1. Continuously oscillating brain potentials, which are referred to collectively as the electroencephalogram or EEG.
2. Temporally discrete potential changes related to individual nerve impulses firing in relation to specific stimulating events. These are alternately referred to as evoked potentials, event-related potentials or event-related desynchronisations. In this thesis the term evoked potentials (EPs) will be used.

2.3.1.1 The Electroencephalogram (EEG)

This electrical aspect of human brain activity was first demonstrated (and named) in 1929 by the Austrian psychiatrist Hans Berger (Gregory 1987). Berger was the first person to demonstrate that certain electrical brain events were related to a subject's state of mind. He observed that when a subject relaxed, the predominant electrical signals detected consisted of 50 microvolt (μV) waves cycling at a frequency of around 10 Hertz (Hz). He named this activity *alpha*.

Berger also discovered that lower voltage higher frequency activity replaced alpha when a subject was involved in any concentrated activity. He called this higher frequency activity *beta*. From this pioneering work emerged an entire field of investigation into the working of the human brain[7].

2.3.1.2 Detecting the EEG

EEG can be detected from the scalp through the application of one or more surface-mounted electrodes. The detected electrical signal is sampled at a rate of between 1 Hz and 40 Hz with the raw EEG signal commonly being separated by frequency decomposition into sub bands of activity[8]. So for example, the waveform shown at the top of Figure 2.4 is part of a raw, unfiltered EEG signal. The traces below correspond to popular frequency bands that have been filtered out of this raw signal. These bands of activity have been identified during the intervening 70 years of study as corresponding to various recognisable psychological and physiological phenomena.

[7] Walter's *Living Brain* (Walter 1953) provides a good overview of early investigations into the human electroencephalography

[8] There is no absolute definition of the limits of each frequency band, so discrepancies and variations can be found in the literature.

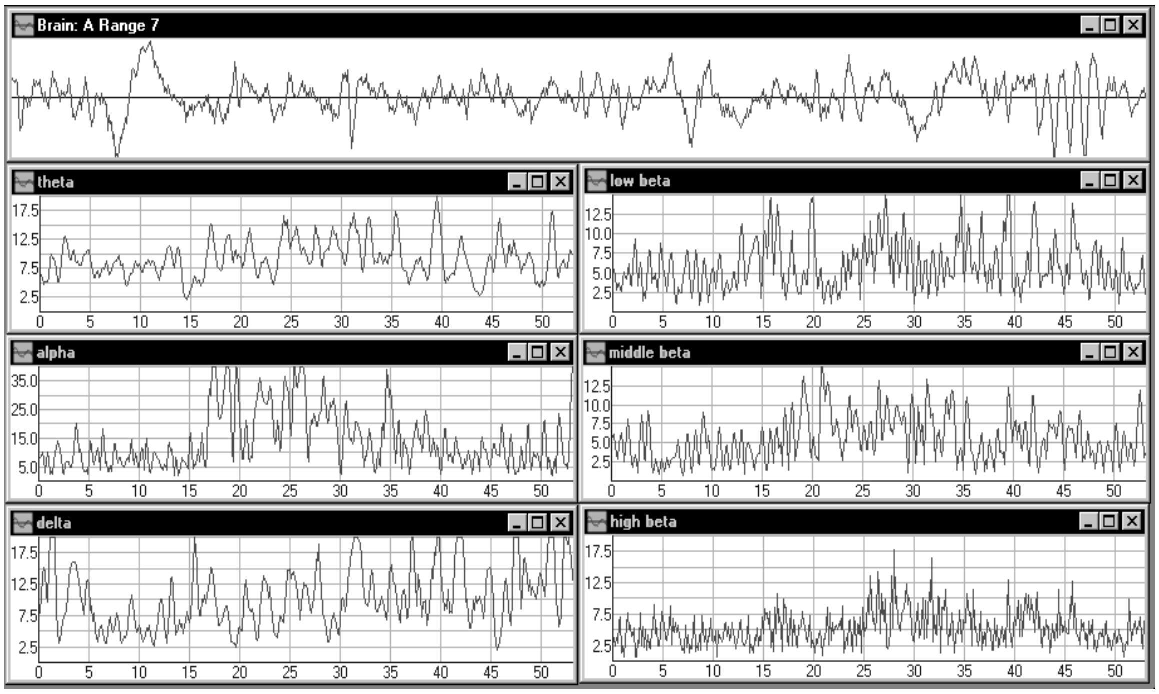


Figure 2.4 Raw and filtered EEG components

EEG Signal	*Frequency Range (Hz)*
Theta	4-7
Alpha	8-12
Beta	15-35
Delta	<4

Table 2.1 EEG component signals

Theta Activity

Theta waves are prominent, normal features in the EEG of children and appear in the adult EEG in times of emotional distress. Theta activity is characterised by moderately low frequencies.

Alpha Activity

Alpha activity has a signal magnitude of between 25µV and 100µV. Alpha activity is detected over those areas of the brain concerned with visual processing namely the parietal and occipital lobes[9]. Alpha activity is normally visible in the EEG of a subject who has his or her eyes closed. For some individuals it can also be present with eyes open when the mind is in a relaxed and unfocused state. The amplitude of alpha is significantly and rapidly reduced by both attentive mental activities and sensory stimulation especially exposure to light. The observation of sudden amplitude reduction is known as *alpha blocking* (Gregory 1987).

Beta Activity

Beta activity is usually associated with concentrated mental activity and the focus of activity within the beta band shifts depending on the mental activity being undertaken. Given the wide bandwidth of beta activity it is convenient to further subdivide beta activity into three distinct sub-bands - low (15Hz-18Hz), middle (18Hz-24Hz) and high (25Hz-35Hz) beta. Focussed attention usually only produces enhanced activity between 17Hz and 19Hz, whereas dwelling and hyper-vigilance has been found to produce activity between 22Hz and 27Hz, and anxiety and panic are often accompanied by enhanced activity in the 30Hz to 33Hz band.

Delta Activity

Delta activity results from extremely low frequency oscillations that occur during deep sleep. They are an abnormal component in the EEG of conscious adults.

2.3.1.3 Evoked Brain Potentials

Evoked potentials (EPs) appear in the EEG as temporally discrete large amplitude spikes of activity. EPs occur in relation to specific stimulating events, such as a sudden sound or a flashing light. Movement (or more precisely the intention to move) produces distinct EPs over the motor cortex of the brain. It is possible to associate EPs to specific stimulating events. This is achieved by repeating the stimulating event a number of times and then averaging between the detected signals in order to extract the response signal from the detected activity. So for example, using this approach it has been found that the signals elicited by movement of the right hand can be distinguished from the signals elicited by movement of the left hand (Penny 1996b). It will be shown in the following chapter that evoked potentials provide useful event-related signals that can be utilised in human-machine interaction.

[9] See Appendix B for a map of the lobes of the brain

2.3.2 Electrical Muscle Activity

It is possible to detect the state of a skeletal muscle from outside of the body by using surface mounted electrodes. A contracting muscle produces a combination of electrical, chemical, structural and thermal changes, which together are known as the muscle's action potential or MAP (Goldstein 1972). A series of MAPs makes up an electromyogram or EMG. By positioning two electrodes on the skin over an appropriate muscle, it is possible to record electrical activity that serves as a correlate of the kinaesthetic activity of that muscle. The resulting signal can be used to deduce the state of the muscle, either complete contraction, partial contraction or complete relaxation.

2.3.2.1 The Electrocardiogram (ECG)

The electrocardiogram is a measure of electrical events associated with contraction of the heart muscle. The frequency of ECG is a measurement of heart rate (HR), i.e. the speed at which the heart is beating. Changes in heart rate are associated with changes in arousal or emotional state. Heart rate variability (HRV), also known as sinus arrhythmia, is a measure of the oscillation of the interval between consecutive heartbeats. HRV has been shown to be responsive to changes in cognitive engagement (Vincente 1987).

2.3.2.2 The Electro-oculogram (EOG)

The electro-oculogram is a measure of the potential difference that exists between the cornea and the retina of the eye. The eye can be viewed as a dipole approximately aligned to its optical axis (Figure 2.5). Measurement of the EOG through surface electrodes positioned around the eye provides information about both the changing position and speed of movement of the eye. Electrical signals generated by eye movement are a potential source of noise during the detection of electrical brain activity. This is due both to the proximity of the eyes to the recording position and the likelihood of eye movement during normal activities. For this reason, electro-oculograms are commonly detected in addition to EEG data so that their signal characteristics can be eliminated from the detected EEG data.

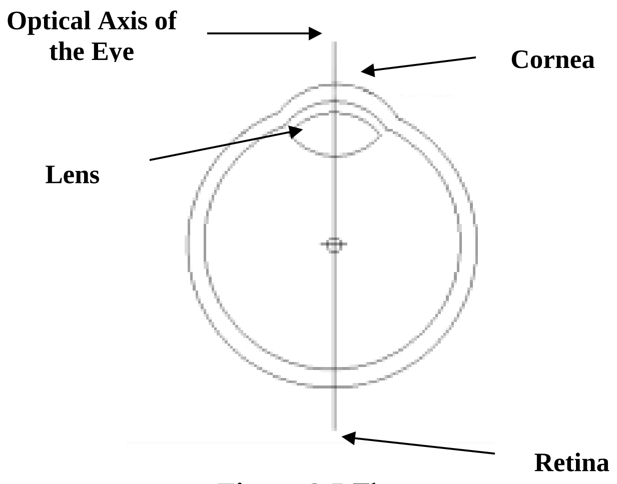


Figure 2.5 The eye

2.3.3 Other Detectable Physiological Information

The signals covered so far have been electrical signals produced naturally by the human body. As described in the introduction to this section, there are other physiological activities which do not in themselves produce measurable electrical information, but which can nevertheless be detected and processed electrically.

2.3.3.1 Electrodermal Activity

If two electrodes are placed on the skin and a small[10] constant current is driven through them, the skin can be seen to behave as a variable resistor. A voltage develops across the electrodes and application of Ohm's law can be used to calculate the effective resistance of the skin. Changes in resistance occur in response to changing excitation in the sweat glands (Edelberg 1972) (see Figure 2.6). Sweat is electrically conductive and the more sweat that is present within the skin, the less resistant (or more conductive) the skin becomes. This measurement is alternatively referred to as an electrodermal response, skin conductance or as a galvanic skin response. Throughout this thesis we shall use galvanic skin response (GSR).

[10] Of the order of 10 μamps

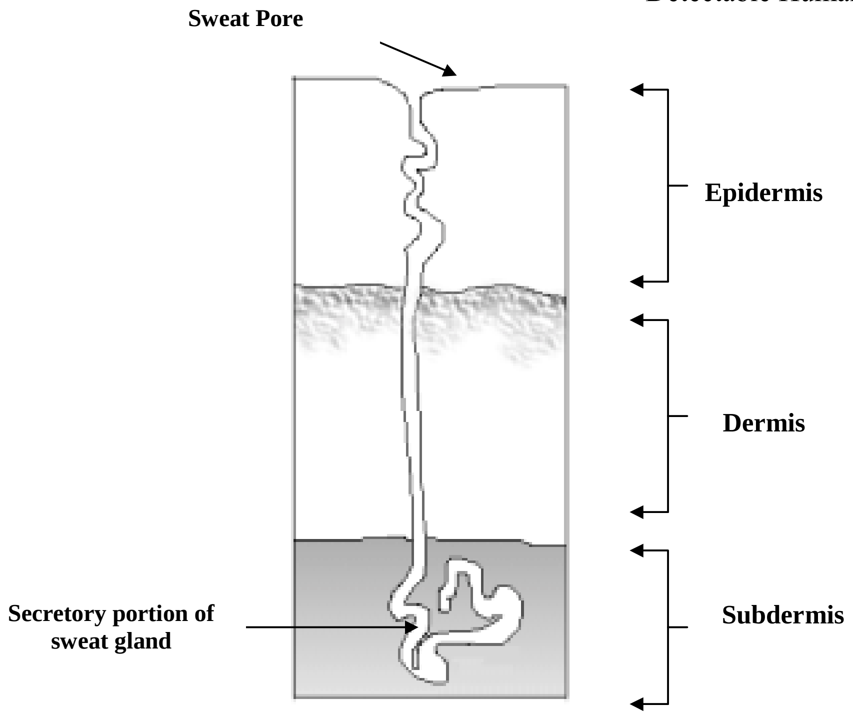


Figure 2.6 A sweat gland

2.3.3.2 Blood Pressure

Blood pressure is a measurement of the force with which the heart is pumping blood around the body. A measurement of blood pressure also reflects the resistance characteristics of the arteries through which the blood is flowing. Changes in blood pressure can occur in relation to changing psychological and emotional state. A plethysmographic device, such as an inflatable cuff, can be used along with an electronic sensor to measure of blood pressure.

2.3.3.3 Peripheral Body Temperature

Dilated or open blood vessels enable more warm blood to pass into body tissue than constricted or closed vessels. Measures of body temperature are therefore really measures of vascular constriction (Schwartz 1995). A measure of temperature can be made available to an electronic device by using a thermocouple, which generates a variable electrical potential according to its temperature, as the sensing mechanism. Vascular constriction is often observed in relation to negative emotional states, such as stress and anxiety.

2.3.4 Physiological Signal Characteristics

What is not apparent from the above description of detectable human physiology is the characteristic nature of electrophysiological activity. As an input source for computers, physiological signals are a less reliable form of data than we are perhaps used to. For a start each individual subject's physiology (and consequently the signatory characteristics of their physiological activity) is unique to them. Potential ranges of "normal" electrophysiological activity available must take into consideration factors such as gender, age and general health. As if this were not difficult enough, most electrophysiological parameters as susceptible to environmental effects such as changes in temperature and humidity. Cardiac activity (heart rate/ blood pressure) can also be influenced by factors such as the smoking, posture and the time of day (Siddle 1980). A comprehensive description of the issues surrounding physiological signal detection and evaluation can be found in Martin et al. (Martin 1980).

These variable factors may appear to preclude the utility of physiological information as an input source. However, if we decide that certain interactive applications would benefit from the incorporation of some combination of electrophysiological signals, it is certainly feasible to detect and/ or control for some of these influences. Of course, any truly useful EPIC system would include the processing capabilities required to learn the response characteristics of a particular user over time and in different situations.

2.3.5 Discussion

What this overview of detectable physiology indicates is that experimental evidence exists for the relationship between detectable physiological signals and psychological state. Currently, there are two main reasons for wanting to detect, process and present human physiological information. The first reason is to enable the quantification or measurement against some scale of a particular physiological signal of interest. Quantification may involve discrete measurement of signals such as blood pressure or temperature. Alternatively it may involve continuous monitoring over time of signals such as heart rate, respiration rate and brain signals. Measurement and monitoring of physiological signals is most often carried out for medical purposes, where the information gathered is used to identify changes in a subject's physiological state which can be indicative of the onset of illness or disease. Alternatively, this information can be used to study a subject's physiological responses to different psychological conditions.

The second reason for detecting physiological information is in order to present it back to a subject in a manner that enables operant conditioning[11] of physiological signals to occur. This process of feeding physiological information back to a subject is known as biofeedback and is the mechanism by which individuals learn to exercise conscious control over seemingly unconscious physiology, such as heart rate and blood pressure (Olsen 1995). The last section of this chapter explores in more detail this second application of signal detection and presentation.

2.4 Biofeedback

In 1948, Wiener first published his treatise on communication and control in biological and mechanical systems (Wiener 1961). He called his collective interdisciplinary scientific theories Cybernetics, the term taken from the Greek word *kybernetics*, meaning steersman or one in control (Wiener 1948). One of the core principles of cybernetic theory is that of feedback, which implies that the controller of a system cannot control a variable unless information about that variable is available to it. In the human physiological system the information of Wiener's model relates to the functioning of the human body, with the variable being a particular physiological parameter, such as heart rate or muscle state data. The system that this information is being reinserted into is the human central nervous system and the controller is, of course, the brain. There is a special term for feedback within the human physiological system and that is *bio*feedback. During the late 1960's work from several disciplines, including psychophysiology, behavioural medicine, stress research, consciousness research and electromyography were diverging on the notion of humans being able to exert conscious influence over seemingly unconscious physiology. In many cases it was found that feeding back relevant physiological information to a subject was the key to successful physiological control.

A comprehensive definition of biofeedback is offered by Schwartz & Schwartz (Schwartz 1995, page 41) (their italics) after Olsen (Olsen 1995, page 29):

[11] A form of instrumental learning or learning through experience

> "…a group of therapeutic procedures that … utilise electronic or electromechanical instruments … to accurately measure, process and feed back to persons *and their therapists* … information with *educational* and reinforcing properties … about neuromuscular and autonomic activity, both normal and abnormal … in the form of analogue or binary, auditory and/ visual feedback signals … to help persons develop greater awareness *of, confidence in and an increase in* voluntary control over their physiological processes that are otherwise outside awareness and/ or less voluntary control, … by first controlling the external signal…"

This definition highlights the roles of both physiological sensing equipment and physiological signal presentation in encouraging an awareness of self-regulatory abilities in an individual. Section 2.2.3 listed some of advantages of using computers over other display technologies for monitoring and measuring physiological data. For biofeedback applications the real power of computers, however, lies in their display capabilities. Where non-computerised biofeedback instrumentation is limited to presenting real-time physiological signal characteristics (magnitude and frequency) as disembodied aural tones or as values on a meter, modern biofeedback systems can utilise the full multimedia capabilities available on desktop computers.

2.4.1 The BioGraph System

Currently, the most popular clinical biofeedback equipment on the market is the ProComp/ BioGraph system by Thought Technology[12]. ProComp is a physiological sensing device with 8 input channels that can be used with a range of different sensors and electrodes to detect any combination of EMG, EEG, temperature, heart rate, skin conductance and respiration. The ProComp system interfaces to an interactive windows-style multimedia application called BioGraph (shown in Figure 2.7).

[12] See Appendix A for hardware manufacturer's contact details

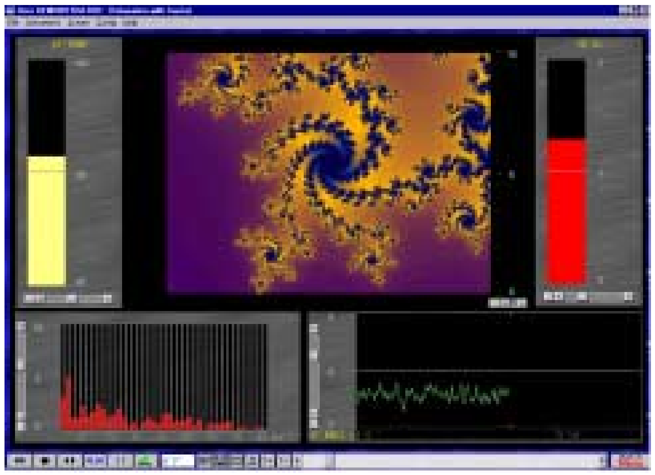


Figure 2.7 BioGraph predefined protocol screen

The BioGraph affords two modes for setting up a biofeedback training session. In the first mode, a clinician selects a predefined screen for a particular training protocol. So for example, Figure 2.7 shows a predefined BioGraph screen for the simultaneous training of 4 signals - temperature, skin conductance, EEG and EMG. The data stream detected for each signal is fed into a different multimedia window. Information about temperature and skin conductance is presented to a subject through animated single bar graphs, shown on the left and right hand sides of the screen, respectively. EMG information is fed back using a polygraphic display (bottom right) and a single channel of EEG is decomposed into its component parts which are displayed in a multi-bar graph (bottom left). The appearance of the fractal in the centre of the screen changes in relation to the changing amplitude of any one of the 4 signals chosen by the clinician.

In the second training mode, a clinician can use the BioGraph software to build a training environment to suit any potential training protocol. This is achieved by choosing different multimedia visualisations from a menu (see upper left corner of Figure 2.8) and associating each visualisation with a particular physiological parameter. Alternative signal representations shown in Figure 2.8 include:

- ❑ An animated face that smiles when a signal’s amplitude goes above a certain value
- ❑ A meter whose needle points to a value corresponding to the current magnitude of a signal
- ❑ Other graphical entities whose appearance changes in response to changing signal magnitude.

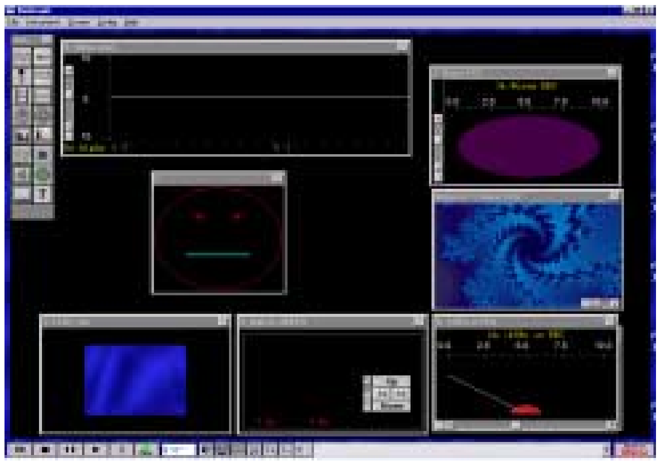


Figure 2.8 BioGraph clinician-defined protocol screen

A clinician-defined protocol screen can be saved and used again. This allows clinicians to build particular training environments for particular individuals and to evolve these environments to suit their changing training requirements. BioGraph also makes it possible for clinicians to record voice commands which can be associated with timers to guide a subject though a training session with a series of instructions. As shown in the predefined protocol screen, BioGraph includes provision of popular modes of signal representation, such as bar graphs that display signal magnitude, as well as polygraphs that display both the magnitude and frequency of a signal. The BioGraph application also allows clinicians to archive data for later analysis. Numerical analysis can take place using a standard tool, such as Microsoft Excel. Alternatively, visual analysis is supported through a playback capability that uses the archived data to reanimate the graphical artefacts within training screen.

2.4.2 Discussion

The BioGraph application is a state of the art biofeedback application, providing clinicians with drag and drop access to a wide variety of biofeedback-specific interaction techniques as well as multimedia signal presentation capabilities. BioGraph is a powerful and flexible windows-style interactive application conceived specifically for clinical use. One problem with BioGraph is that it is a proprietary application written specifically for the ProComp physiological sensing device. This means that the BioGraph application can not easily be ported to other commercially available multi-channel biofeedback devices. Furthermore, as it is a commercial application

BioGraph is closed to further development by third party application programmers. This presents a problem for researchers wishing to explore the potential of integrating electrophysiological information detected with the ProComp hardware into their own interactive applications.

Fortunately, most sensing hardware manufacturers (including Thought Technology) will provide low-level libraries to drive the device's signal detection and pre-processing functionality. However, a lack of software development tools designed to provide support for electrophysiological sensing-based applications means that researchers interested in exploring electrophysiologically interactive computing applications currently have to develop each new application from scratch. This need for development support is the main impetus for this thesis.

Biofeedback is a pioneering EPIC application that has been evolving feedback technologies for 40 years. For this reason, interactive biofeedback applications are good place to start thinking about development support. As will be shown in the following chapter, existing applications of biofeedback-based training stretch beyond clinical amelioration of psychophysiological disorders. Biofeedback is also the enabling technology behind a range of prosthetic devices for the disabled. Within the description of physiological signal detection presented above, it was stated that medical monitoring applications have been the driving force behind fundamental developments in physiological sensing research. Biofeedback-based applications are driving the development of cheap, peripheral physiological sensing devices. These highly accessible sensing technologies provide ideal platforms for exploring electrophysiological human-computer interaction.

2.5 Summary

This chapter has introduced the technologies behind detection and presentation of human physiological information. It has introduced a range of detectable physiological information and has described the kind of sensing equipment required to both sense and help make sense of this information. It provides a background in human physiology, physiological signal detection and signal presentation in order for the reader to appreciate the review of EPIC systems which is presented in the following chapter.

The increasing availability of sensing hardware means that the detection and electrical processing of human physiological information is no longer the preserve of researchers working within medically related fields. Electronic sensing hardware is commercially available which interfaces to a computer, making the physiological signals detected available to interactive applications. Drawing on knowledge about the relationship between human psychological and physiological states, computer systems developers can begin to consider the implications of incorporating physiological information into the human-machine relationship. Physiological signal sensing hardware constitutes a new class of peripheral device that can be used to enrich the relationship between the user and a computer. Indeed, work has already begun on the development of systems that can recognise and respond to some of the known physiological signatures of psychological state. This work is discussed in the following chapter.

This chapter has also introduced the concept of biofeedback as an applied clinical technique. The description of the ProComp/BioGraph system served to spotlight the role that interactive multimedia computing applications play in existing biofeedback training systems. In the following chapter, biofeedback is explored further as both a clinical intervention and as a means to train disabled individuals in particular to generate conscious physiological signals for *hands-free* computer control. The aim of this exploration is to identify popular interface functionality found in different biofeedback applications. Following on from this, other interactive applications that incorporate human physiological information are described, again with a view to identifying common, physiological-signal related interactive functionality. The aim of the following chapter is to introduce the reader to the broad range of existing applications that rely on the detection and presentation of human physiological information. This information is used to identify common physiological signal-related interaction techniques for inclusion in an EPICS development toolkit.

Chapter 3

Electrophysiologically Interactive Computer Systems (EPICS)

3.1 Introduction

The previous chapter introduced the concept of detectable human physiology and examined the technologies required to detect and present physiological information. It was shown that interactive multimedia computer applications such as BioGraph are playing an increasingly important role in signal manipulation and presentation for at least one fundamental EPIC application, namely biofeedback. This chapter introduces other types of interactive multimedia systems that incorporate human physiological information. Identification of common features and functionality through this review has informed the design of the EPIC system support tools covered in later chapters.

The material contained in this chapter constitutes a unique review of interactive computer systems that incorporate electrophysiological information. From the material presented in the previous chapter it is apparent that producing electrophysiologically interactive systems is a multidisciplinary exercise. Following on from the previous chapter, it will become increasingly evident here that many electrophysiologically interactive systems are developed for specific, medically focused applications (i.e. clinical biofeedback). Consequently the majority of research and development in this area has so far centred on building electronic sensing technologies and understanding detectable changes in human physiology. Neither area of research could be said to involve computing science issues (although the help of some software developers have obviously been enlisted along the way).

Up until this point in time computer scientists have perhaps not recognised the vital role they have to play in the evolution of electrophysiologically interactive computer systems. From the use of novel interface technologies in support of medically based EPIC applications (Allanson 1999a) through to an exploration of the roles electrophysiological information may play in our everyday interactions with computers, it is clear that there is much work to be done. However, the field of study itself must first be defined, which is the main aim of this chapter. We begin by outlining what is to follow.

3.1.1 Overview

Advances in interactive technologies that incorporate human electrophysiological information are emerging to support a diverse range of disciplines. These include clinical biofeedback applications, developments in prosthetic technologies for the disabled, investigations into the psychological and physiological effects of task load in human-machine interaction and the development of emotive technologies. The most natural division that is evident in existing research is between *systems that focus on the control of interactive computer systems (and other electronic devices) in a hands-free manner* and *systems that focus on assessing the physiological state of the user*.

Systems in the first category are concerned with training the user to gain conscious control over their own physiology. This may be for a clinical reason or it may be in order that they can operate some prosthetic device. The physiologically enabled interactive systems that fall into this category are classified here as being **training** EPIC systems.

Systems that fall into the second category are those concerned with monitoring a user's physiological state over time. The physiological information collected may not necessarily be presented to the subject (as is required with training systems). However, the collected physiological information will normally be made available in some manner to a psychophysiologist for assessment. In order to carry out real-time, on-line assessment of human physiological state, the more complex monitoring systems incorporate learning algorithms or neural networks, which attempt to extract state-related features from the detected physiological data stream. Regardless of their complexity, the interactive applications that fall into this second category will be classified as **monitoring** EPIC systems.

There are subclasses of hands-free control systems that span both of these two system categories defined above, however. For this reason this review chapter is organised into three main sections. The following section examines systems that use human electrophysiological information as a mechanism for hands-free human-machine control. As stated in the previous chapter, biofeedback training is the key to conscious control of physiological signals. The first part of this chapter takes a closer look at both the interfaces and applications of clinical biofeedback systems as these systems provide functionality that may also be useful for training physiological signals for hands-free control.

The third part of this chapter examines systems designed to explore direct brain-to-computer communication. Some of these so-called neural interface technologies (NITs) monitor and respond to naturally occurring brain activity. Others rely on the generation by a subject of consciously controllable brain signals. In order to convince the reader of both the potential applications and limitations of direct brain-to-computer based interaction techniques as well as the need for support tools for future developments in this exciting area of HCI, a comprehensive description of work thus far carried out in the field is provided.

The fourth part of this chapter focuses on applications where monitoring of physiological information takes place in order to discover something about the state of a computer system user. This monitoring can provide a system with information pertaining to either the health of a user or their psychological state. It will be shown that access to physiological state information can help inform the design of interactive computer systems as well as provide vital information to a running application as to the condition of its human operator.

This chapter closes by highlighting similarities between presentational and functional aspects of both interactive biofeedback-based systems and physiological state monitoring systems. An examination of the architectures of existing EPIC systems introduces some of the issues regarding tool support for interactive systems development. This sets the scene for the following chapter which concentrates on assessing existing support for interactive systems development and identifies the best method of supporting EPIC systems development.

3.2 Training EPICS

It was stated above that training EPICS are systems designed to support biofeedback-based training for conscious control of human physiology. Chapter 2 included a definition of biofeedback that described the feedback loop that must be maintained between a subject and the physiological signal display mechanism. The physiological signal feedback loop for a system incorporating computer-based signal presentation is shown in Figure 3.1. The role of the computer is to retrieve physiological signals from the sensing hardware, pre-process the signals and display those signals back to subject in real time. In order to decide how best to provide support for physiological signal retrieval, preprocessing and display it is necessary to look at the range of applications to which biofeedback-based physiological signal training is being applied.

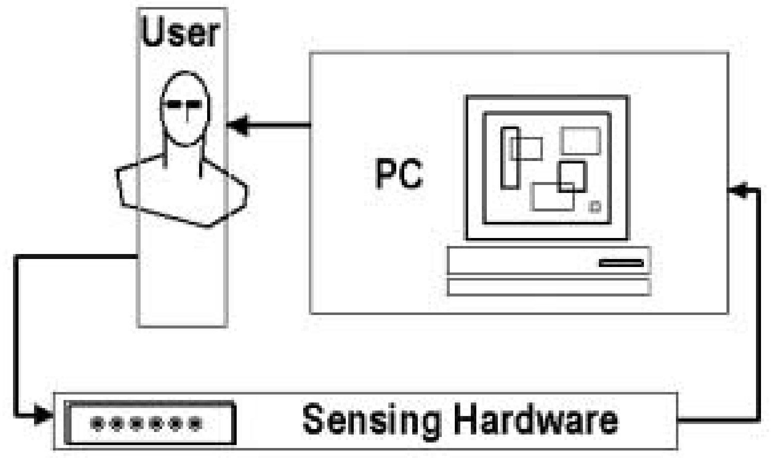


Figure 3.1 Feedback loop for biofeedback-based training

3.2.1 From Clinical Biofeedback to Hands-free Control

At the present time computer-based physiological signal presentation systems are being developed to service two distinct applications. The first, which we have already introduced, is clinical biofeedback. The second application area influencing the development of EPICS technologies is physiological signal-driven hands-free human-machine interaction.

The same signal preprocessing and presentation requirements exist for both clinicians practicing biofeedback and researchers wishing to explore the potential of electrophysiological signal-based hands-free human-machine interaction. The following sections examine technological developments for both application fields.

3.2.2 Electromyographic Training Applications

Electromyography or skeletal muscle feedback is applied clinically to treat a range of physiological disorders from urinary incontinence through to chronic pain. Established as a clinical technique more than 20 years ago (Basmajian 1977) EMG biofeedback is also a popular method for training patients to regain muscle control lost due to accident or illness. Physiological rehabilitation can often be a slow process, particularly in the early stages of treatment, due to remaining muscle activity being so slight as to not cause any obvious physical movement. A seeming lack of ability to produce visible action within a muscle can compound the problem of slow progress by causing patients to lose motivation. However, these motivational issues can be overcome by visually amplifying and feeding back EMG activity. This visual amplification is often achieved through computer-based presentation techniques.

As described in the previous chapter, integrated biofeedback systems such as the ProComp/ BioGraph system provide facilities for feeding back EMG signals using computer animations and/ or sound. However, applications like BioGraph are currently limited in that these two forms of signal display are the only modes of representation available.

In 1997 Bowman reported on an *Enhanced Sensory Feedback Device* (ESFD) being used clinically in the rehabilitation of patients with muscle control problems (Bowman 1997). The device detected EMG information from a subject, making the changing signal characteristics available to a computer. The role of the computer was to transform EMG activity into control commands to a simple interactive game. The game was a form of *PacMan* where progression of the PacMan creature around a maze occurred only when the EMG signal amplitude was maintained above a predefined threshold value. Alternatively, the EMG signal drawn by the computer from the ESFD was used or to operate an electronic remote-control car.

The ESFD, developed by the Institute of Interventional Informatics (I^3 2000), demonstrates a novel evolution of biofeedback interface technology. The technology behind the ESFD is an 8-channel electronic EMG sensing device called AnyWear (Lipson 1998). This device is a serially connected computer peripheral which streams EMG signal data to a purpose-built application called NeatTools (Salgado 1999). NeatTools pre-processes the signals in order to transform their changing amplitudes into simple control commands that can be used with its game-based training application. The NeatTools application also serves as the interface between the AnyWear sensing device and other electronic peripheral devices, such as the car.

The idea for the AnyWear system developed out of I^3's earlier experiences with another peripheral physiological sensing system, known as BioMuse. BioMuse was created by researchers at Stanford University who were interested in using EMG signals to create music in a hands-free manner (Knapp 1990). David Warner from I^3 came across the 8-channel BioMuse device in 1991 and realised that by redesigning its interface, he could employ it as a computer input device for his quadriplegic patients. The first disabled user of the modified BioMuse system was an 18-month-old girl who used muscles around her eyes to move a smiling face icon around a computer display (Lusted 1996). A series of versions of the AnyWear system have subsequently enabled severely disabled subjects to interact with both computers and electronic objects within their environments through EMG signal control alone (Warner 1999).

The ability to apply a physiological signal to the control of an external device leads us to another common application of biofeedback-based technologies – as prosthetic devices for the disabled. Electromyographic limb control technology was first demonstrated in 1958 (as reported in Bennett 1981), but it has only been since the early 1980's that medical engineers have been developing useable EMG-driven electronic prosthetic limbs (Saridis 1982, Kelly 1990). EMG prosthetics are most often upper-body limbs with embedded sensors that detect activity from muscles available around the site of the missing limb. Signals from these muscles are used to operate various aspects of the prosthetic's functionality, for example bending of the elbow, flexion of the wrist and/or gripping actions can all be switched using the varying amplitude of individual EMG signals.

Learning to operate any form of EMG-driven prosthesis involves a period of training where a user experiments in order to discover which physical actions cause the desired control outcome. This period of feedback-based training can be carried out directly with the prosthetic or alternately using a software training system such as the ProComp/ BioGraph. Existing research (Lake 1997) highlights the benefits of feedback training as part of the process of getting new amputees to accept and gain control of prosthetic limbs. Despite this, very few dedicated training technologies are available for prosthetic-specific EMG training. At the present time, recipients of prosthetic limbs learn to generate the right types of EMG response using the prosthetic itself as the feedback interface.

The next logical step beyond hands-free technologies for the disabled is to explore EMG as a mechanism for mainstream human-machine interaction. In 1998 Rosenberg (Rosenberg 1998) described the first interactive system designed specifically to demonstrate the potential of using EMG signals as a means of hands-free control for mobile computing applications. Rosenberg's *Biofeedback Pointer* consisted of three sets of electrodes positioned on the wrist in such a way as to detect EMG activity related to motion in all three degrees of freedom of the wrist (rotation, forward/backward and side-to-side). EMG activity was collected using a custom built sensing device signals from which were fed to a software neural network running on a PC. The initial interface to Biofeedback Pointer was configured to act as a training environment for the neural network. This involved the system automatically moving a cursor around a screen so that a subject could follow the motion with his or her hand. Data collected from the wrist sensors could be fed to the neural network, which used the information to recognise patterns of signal activity from the three wrist muscles associated with wrist position. Subsequent to the network-training period, the system interface was changed so that it presented a biofeedback interface where the subject was given real-time visual feedback in the form of EMG signal-driven motion of a cursor on the screen.

EMG is the most well understood and immediately promising physiological signal source for hands-free human-machine interaction. This is due mostly to muscle control being a skill inherent in all able-bodied individuals. However, there are other signals used commonly in clinical biofeedback training that are now also being explored as potential hands-free control signal sources.

3.2.3 Galvanic Skin Resistance Training Applications

A range of GSR-specific computer-based sensing devices have been developed with biofeedback training applications in mind[13]. One sensing device designed specifically as a hands-free computer control peripheral is the MindDrive system (Other90 1999). Despite professing to "*read the user's mind*" this device simply detects GSR from the tip of a user's finger and utilizes the changing magnitude of the signal to control aspects of interactive computer animations (Figure 3.2). A variety of GSR-responsive animated training environments, each of which provides a single goal-driven task, are available in various software packages bundled with the device. One is a ski-ing game where GSR magnitude is used to control the side to side motion of the skier.

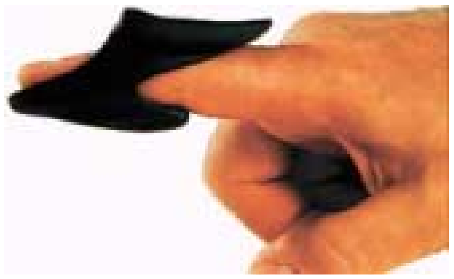


Figure 3.2 The MindDrive device

RelaxPlus by UltraMind (UltraMind 2000) consists of a GSR sensing device and software application. Conceived as a computer-based tool to aid relaxation, RelaxPlus, like the MindDrive packages, provides a user with goal-driven biofeedback training tasks. So for example, the goal in one RelaxPlus animation is to successively change an animated fish into a mermaid, a girl, an angel and finally into a star. Each stage in the progression of the animation is related to a particular GSR signal magnitude value, so that as a user relaxes and their GSR falls, they advance through the sequence of animations. A snapshot of the relationship between detected GSR (shown along the vertical left-hand bar) and actual signal representation (red line plus sequential abstract animations) is illustrated in Figure 3.3.

[13] Most clinical GSR devices have integrated feedback displays are stand alone devices

GSR is unlikely to prove itself a reliable control signal for future HFC applications due to its susceptibility to environmental influence. Nevertheless, there are other reasons for incorporating information about a user's changing level of GSR into interactive systems, as we shall see later in this chapter.

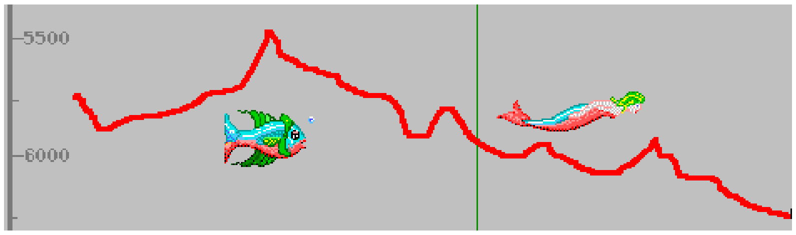


Figure 3.3 Overview of an abstract GSR training interface.

3.2.4 Electro-oculographic Training Applications

EOG-based eye tracking is an example of an established physiological signal-based form of hands-free human-machine interaction. As described in section 2.3.2.2 electro-occulograms or EOGs are detected from around each eye and correspond to measure of changing voltage related to orientation of the eye. By positioning electrodes above, below and to the sides of the eye it is possible to measure EOGs associated with both horizontal and vertical orientation of the eye.

A good example of an EOG-based HFC system is the EagleEyes system developed by researchers at Boston College (Gips 1996). Designed as a prosthetic technology, the EagleEyes sensing device detects electrophysiological information from a subject, amplifies it and makes it available to a personal computer. The EagleEye software pre-processes the detected EOG signals and converts them into the positional coordinates of a cursor on a computer screen. Existing interfaces for the EagleEyes system include a software keyboard designed to enabled a user to input text using EOG signals alone (letter selection is a function of time spent fixating on a letter). Other interfaces include simple interactive games, a small movie clip database browsed using EOG signals and an "eye painting" application which draws a user's eye movement as coloured lines on a screen.

As with any electrophysiological, HFC system, a period of biofeedback-based training is required in order to interact usefully with any EOG-driven interactive application. Most multi-channel signal sensing devices can be used to detect EOG, although as EOG feedback has no rehabilitative applications, existing interfaces to clinically-derived devices are unlikely to contain suitable tasks for training a user in EOG signal control.

3.2.5 Electroencephalographic Training Applications

Beyond its role in physiological rehabilitation, biofeedback is used increasingly as part of the treatment for a growing number of *psycho*physiological disorders. The focus of training for psychophysiological conditions is EEG or brain signal biofeedback. EEG feedback training is also known as *neurofeedback*. Conditions for which neurofeedback is an applied technique include[14]:

- Attention Deficit Disorder (ADD) (Tansey 1993, Lubar 1995a, Lubar 1995b)
- Addictions (Peniston 1989, Denney 1991, Fahrion 1992, Watson 1978, Saxby 1995)
- Anxiety (Hare 1982)
- Depression (Baehr 1997)
- Post Traumatic Stress Disorder (PTSD) (Peniston 1991, Peniston 1993)
- Sleep disorders (Bell 1979).

For each condition, a training protocol is applied which has been found through empirical investigation to contribute to the amelioration of that condition in other subjects. So, for example, Lubar (Lubar 1997) and others have identified signatory dysfunction in the EEG of sufferers of ADD. This dysfunction corresponds to increased activity in the theta band and reduced activity within the beta band. Consequently, a popular protocol for ADD involves training subjects to suppress theta activity and enhance beta activity. In order to achieve this, a subject is presented with real-time feedback regarding the changing magnitude of these two EEG components. Signal representation usually takes the form of individual bar graphs, one for each signal, whose heights corresponds to the amplitude of the signal. A further common mode of signal representation for neurofeedback is disembodied aural tones whose changing pitch reflects changing signal magnitude.

[14] A comprehensive list of conditions treated using biofeedback can be found at www.eegspectrum.com

In recognition of the fact that the majority of sufferers of ADD are children, clinical biofeedback systems such as the Neurocybernetics II (EEG Spectrum 1999) enable feedback to be given in the form of interactive computer games. These are often goal-oriented with the subject receiving some reward for achieving desirable signal characteristics. So for example the Neurocybernetic II includes a PacMan-like game, progression through which continues only if a particular EEG component signal's magnitude stays above/ below a predefined threshold value.

ADD training protocols can require the simultaneous training of multiple components of a subject's EEG. This can only be achieved through use of a multi-channel sensing device detecting physiological signals through an array of electrodes. Signal-specific devices such as the BrainMaster (Brainmaster 1999) and the Interactive BrainWave Visual Analyser (IBVA 1998), can detect only one type of electrophysiological information (EEG in both cases). Other, more general systems like the ProComp, BioMuse and Neurocybernetics II can be configured to detect a range of physiological data.

Apart from clinical applications of EEG biofeedback training a separate body of research is emerging explicitly concerned with EEG-based hands-free human-machine interaction. Direct brain-to-computer communication is causing much interest in the popular press (Boothroyd 1997, Ciarcia 1988, Metz 1997) and for this reason research focusing on EEG-based HFC applications is dealt with separately in a subsequent section of this chapter.

3.2.6 Multiple-Signal Training Applications

As any single aspect of detectable human physiology may by itself prove unreliable as a hands-free control signal, multi-channel and multi-signal devices lend themselves to the possibility of training combinations of physiological signals in order to find a more reliable combi-signal[15].

In 1994/5, the US Air Force's Armstrong Laboratory has carried out a comprehensive evaluation (Junker 1995) of one multi-channel sensing device as a potential hands-free control support technology. The subject of their study was the Cyberlink system (BAT 1998) which can be used to detect both EEG and EMG signals through a sensor-filled headband applied to the forehead.

[15] Combined signal

The detected combi-signal (referred to in the study as the *brain-body* signal) can thereafter be made available to a computer for pre-processing and/or display. The aim of the 1994/5 study was to train a group of aircraft pilots to gain conscious control of both EEG and EMG signals simultaneously with a view to their performing secondary operational tasks, such as switching between the radio and the radar, in a hands-free manner (McKenna 1996).

The Armstrong Laboratory's study involved six able-bodied subjects, who participated in ten training sessions, each of between 45 and 60 minutes duration. A session would involve the subject carrying out multiple trails of one of two tasks. The first task was called *Pong*, a form of the 1970's video tennis game. In this task the user's brain-body signal was translated into horizontal motion of a paddle on a computer display. A single trial involved a ball appearing from the top of the screen, which the subject was to try and intercept with the paddle as it fell (descending on either a direct vertical or diagonal path). The second task was called *Capture* and required a subject to move a box along a horizontal axis in order to capture a ball which appeared at a random point along that axis. Scoring the trials focussed on whether the subject successfully intercepted (*Pong* task) or captured (*Capture* task) a ball.

Four different EEG components were evaluated during the trials, one in the theta band, one in the alpha band and two in the beta band. For each EEG signal (and for each subject) a threshold value was set. Raising the amplitude of a particular EEG component above its threshold moved the paddle/capture box to the right. Alternatively, lowering the amplitude below its threshold moved the paddle/capture box to the left. Results from the study indicated that all of the subjects found each of the EEG components equally controllable. Furthermore, with task accuracy scores for all subjects ranging between 70% and 90% the study demonstrated the promise of brain-body signals as inputs for future hands-free control technologies.

Following the success of the initial evaluation, the Armstrong Laboratory brought the CyberLink device together with existing flight simulator technology to test the viability of electrophysiological hands-free control in a real-world task (Nelson 1996). Within the simulator, subjects were presented with an aircraft flying across a terrain. The aircraft's heading, altitude and forward velocities were all fixed, but its orientation was to be controlled by the subject's brain-body signal. As in the evaluation, a brain-body signal exceeding a predefined power threshold caused the aircraft to roll to the right, while signals dropped below the threshold caused a roll to the left. An ideal flight path was indicated in the simulation as a series of hoops through which the subject was to attempt to navigate. Again, results were promising with an average of 70% of hoops being traversed successfully.

3.2.7 Discussion

All of the systems described in this section have been conceived with the same overall aim - to assist an individual in the training for control of one or more physiological parameters. Regardless of whether the training itself is the reason for creating the system (as is the case with clinical biofeedback applications) or whether the training is a step toward some further goal (such as hands-free interaction with a computer-based system) all of the systems described above share the following features:

1. The need to retrieve electrophysiological data from a dedicated electronic sensing peripheral
2. The requirement to pre-process that data in order to make it suitable for presentation
3. The need to present the data to a subject in a manner suitable for learning to take place

As we have seen electronic sensing peripherals come in all shapes and sizes. Devices such as the MindDrive provide a computer with a single stream of raw information about the changing characteristics of a single physiological parameter. Multi-channel devices like the ProComp can provide a computer with multiple raw streams of data about a single physiological parameter or alternatively multiple raw streams of information about a number of physiological parameters. A biofeedback application needs to include some mechanism by which a clinician or trainer can indicate which physiological signals that application needs to receive information about (signal availability is of course related to the type of sensing device being used).

The physiological signal pre-processing undertaken by an EPIC system varies in relation to the complexity of the sensing device it is utilising. Most commercial devices are *serial* peripherals, meaning that multi-channel sensing devices have to multiplex information about a number of physiological parameters into a single data stream in order to send it over a serial connection. The first task for an EPIC application therefore is to de-multiplex the physiological data streams and identify which data belongs to which physiological parameter.

Each physiological parameter has its own frequency characteristics which in turn increases the level of complexity in the pre-processing stage of data management. EEG components for example are defined by their frequencies and this can in turn effect the rate of data retrieval necessary within the application. With EEG components, the amplitude of a component's signal is of importance. So whereas the sensing equipment itself will be processing physiological signals at particular rates, the application may not need to retrieve data from it at exactly that rate so long as the retrieval rate is high enough to provide an accurate picture of the changing amplitude of the signal. The only time that the rate of retrieval is important is when the application is focusing on physiology that is frequency-specific (heart rate for example).

Perhaps the most obvious feature of the biofeedback training systems we have described is the diversity of their display requirements. Methods of feeding back physiological information range from multimedia displays through to the use of alternative electronic peripheral devices. Information pertaining to each individual physiological data stream needs to be displayed in a manner appropriate to both the aim of a training session and to the trainees themselves. Graphs are useful as they present both magnitude and frequency information, although they represent a method of presentation that is often only meaningful to a trained expert. Trainees tend to benefit more from abstract means of signal presentation such as animated graphics and disembodied tones (or the motions of an electronic car!).

One feature found in a number of the systems described is the idea of rewarding physiological control in a trainee. The interface to the RelaxPlus application for example demonstrates the use of pre-set reward thresholds, where the GSR signal values that cause the animation to change from one state to another are hardwired into the application. This means that the actual GSR values that cause change in the animation will be the same for every user of the RelaxPlus system. Alternatively, the BioGraph application allows a clinician to set reward threshold values for a given signal, and change the value of that threshold value across training sessions or even during a session in order to suit the changing abilities of a subject.

One issue that will come up again in the next section of this chapter is the lack of flexibility within systems designed for exploring hands-free control for training apart from the actual control task. By providing both the range of signal presentation options and the ability to engineer incremental reward-based training tasks, systems such as BioGraph demonstrate the functionality required in systems used for training users of electrophysiological hands-free control systems.

One final point that needs to be stressed is that existing biofeedback applications are, with very few exceptions, proprietary applications that each support the retrieval of physiological information from one particular manufacturer's sensing peripheral. One function of this chapter is to support an argument for how best to impact upon this situation and encourage development of EPIC applications that are sensing hardware independent. In order to fulfil this function the chapter will continue by examining other types of interactive applications that rely upon the retrieval, processing and presentation of physiological information.

3.3 Neural Interface Technologies

Howard (Howard 1996) refers to traditional computer interfaces as *"the Big Barrier between the computer and the human mind*". Underlying Howard's view seems to be the notion that traditional mechanisms for human-computer communication require the translation of a user's thought and intention into physical interaction with a contrived user interface consisting of physical input device and display. Interaction is limited by the constraints provided by the interface to the computer and because of this, creative human-machine interaction is stifled. For Howard, breaching the Big Barrier requires the building of computers that will recognise and respond directly to human thought.

Such mind-operated machines may sound like science fiction, but they are slowly becoming a scientific reality. This reality is taking shape in research laboratories that are exploring the possibilities of transforming human EEG information directly into computer control commands. Work being carried out at the extreme edge of brain-computer interface research is looking at implantation of electrodes directly into the brain (Graham-Rowe 1998). There are obvious issues with the intrusive nature of this approach that currently limit its applicability to use by only the most severely disabled individuals. However, a more acceptable approach to brain-computer interfacing involves the continuous sensing of EEG activity via surface-mounted electrodes. This is exactly the same approach as is used in clinical neurofeedback applications (section 3.2.5) and will be the mechanism for brain-computer interfacing that we shall consider here.

Existing research investigating EEG-based brain-computer control can be divided into two categories:

- Systems which detect and process naturally-occurring brain signals
- Systems which aim to train a user to produce consciously-controllable brain signals

In the two sub-sections that follow, each category of brain-computer interface research will be considered in turn. The terms *neural interface technology* (NIT) and *brain-computer interface* (BCI) are both popular within the literature and are used inter-changeably throughout the rest of this thesis.

3.3.1 Processing Naturally-occurring Brain Signals

The main feature shared by the BCI research presented in this section is that all the systems described aim to allow the user to behave *naturally*. In other words the role of the computer is to recognise and respond to behaviour-related brain phenomena extracted from a stream of detected EEG data. From a high level, systems designed to process naturally occurring brain signals can be envisaged as human-machine systems where the computer is doing all of the work, adjusting its behaviour to suit that of the user. This is a very different style of interaction than is common with existing methods of human-machine interaction, where the user has to modify his or her behaviour to match the requirements of the machine.

A "thought-induced" computer action may be no more complex than notification within an application that the magnitude of some EEG component's amplitude has passed above or below some predefined threshold value. Such a switching signal (which is effectively the result of notification of a signal passing a single threshold value) can be used to instigate some action within the application. These systems require provision for setting threshold values for each EEG component signal to be used as switch. The requirement to be able to set reward thresholds is a feature we have already encountered in the goal-driven biofeedback system interfaces described previously.

More complex brain-signal processing systems require the incorporation of neural network technology in order to try to recognise a user's particular thought or action. This second approach to utilising naturally occurring brain activity, though more ambitious, is nevertheless yielding some interesting early results. However, we will begin by looking at systems created to investigate simple EEG component amplitude-based switching.

It is widely accepted that the earliest investigation into the use of EEG component amplitude to control an electronic device was carried out by Edmond Dewan (Dewan 1967). Dewan filtered alpha activity from the EEG, connecting the output from the filter to a Schmitt trigger, which generated an electrical pulse whenever the amplitude of the alpha signal exceeded a predefined threshold value. The pulse was fed to a computer, which was programmed to translate pulses into Morse code dots or dashes (depending on the pulse duration).

During evaluation of the system each subject’s task was to transmit, in order, the letters of the alphabet using only alpha-generated Morse code. Each subject was instructed to look upwards with eyes closed to achieve maximum signal voltage (alpha present = pulse on). For minimum voltage (alpha absent = pulse off) subjects were either to open their eyes and fixate them on a nearby object, or to fixate closed eyes on an imaginary point some distance in front of them.

The results of Dewan’s study where not overly promising in that 33% of all Morse characters transmitted to the computer were erroneous and had to be re-sent. Taking into account the time to re-send the correct character, the inter-character pause period and time allowed for noise present in the signal, it was found to take approximately 35 seconds to transmit a single correct letter using alpha signals only. Of course the bottleneck in a system designed to respond to alpha amplitude switching is the binary nature of the control signal. A two-state system is limited by the speed at which a user can close and open his or her eyes and does not therefore provide the best mechanism for transmitting text. This has not, however, deterred others from exploring brain-computer interaction based on alpha signal switching (Kirkup 1997, Patmore 1994).

In 1988 Farewell et al. (Farewell 1988) developed a system to allow disabled users to input Roman characters into a computer using a single, recognisable evoked brain potential. Farewell’s system used an event-related brain signal known as the P300 response (so called because it appears as a spike in the EEG signal 300 milliseconds (mS) after a stimulating event). The interface to Farewell’s system consisted of a 6-by-6 matrix presented on a computer screen, which contained the 26 letters of the alphabet (see Figure 3.4). In order to select a particular letter, a subject had to focus their attention on that letter for the duration of a selection period. During that selection period each row was intensified for a period of 100mS in turn. When all of the rows had been intensified, each column was intensified for 100mS in turn. As the P300 appears 300mS after a stimulus, a period between each intensification of at least this was required to ascertain whether a response was present for a given row or column. The final inter-stimulus period (ISI) used was 500mS.

MESSAGE					
		BRAIN			
Choose one letter or command					
A	**G**	**M**	**S**	**Y**	*****
B	**H**	**N**	**T**	**Z**	*****
C	**I**	**O**	**U**	*****	**TALK**
D	**J**	**P**	**V**	**FLN**	**SPAC**
E	**K**	**Q**	**W**	*****	**BKSP**
F	**L**	**R**	**X**	**SPL**	**QUIT**

Figure 3.4 Farewell's soft keyboard

EEG data was detected from an electrode positioned over the occipital lobe[16]. Pre-processing involved filtering and amplifying the signal after which real-time analysis of signal data could be carried out. The system used a covariance algorithm to identify which letter had elicited a P300 response. Using this system, four able-bodied subjects successfully completed the task of spelling the work *BRAIN* using only their P300 response. However, the character identification (and thus transmission) rate of the system was little better that Dewan's, at 26 seconds per character.

Four years later, researchers at the Smith-Kettlewell Eye Research Institute (Sutter 1992) attempted to improve on Farewell's character transmission rates. The interface to their system consisted of an 8-by-8 matrix containing labelled fields or keys. Beneath the initial matrix were constructed 32 levels of key functionality. So, for example, the 1st key overlay contained the alphabet and a set of most frequently used words. If the word that the user required was not on the first screen, then he or she selected the first letter of that word. The system then switched to an overlay of words beginning with that letter, and so on. In total, the system provided access to 700 commonly used words.

[16] see Appendix B

A prototype system was evaluated using 20 disabled and 70 able-bodied subjects. After a training period of up to one hour, the system could recognise which cell in the matrix an able-bodied subject was focussing on in between 1 and 3 seconds. Muscle generated signal contamination proved to be a problem with disabled subjects, and this was overcome in one instance by detecting EEG using an inter-cranial implant. This implant remained in place for an 11-month evaluation period, toward the end of which the subject achieved communication rates of around 11 words per minute using the system.

Keirn and Aunon (Keirn 1990) chose a different type of brain response to study in order to identify patterns of brain activity associated with particular mental tasks. Their notion was to translate distinct brain responses into a computer control language. The underlying principle behind this system was one of hemispheric specialization – effectively the ability to distinguish between a task which involves activity mainly in the left hemisphere of the brain, and a task that involves activity mainly in the right hemisphere (Springer 1993). Keirn's notion was that if it were possible to distinguish between a left brain task, a right brain task and a third resting or no-activity state, then it would be possible to construct a language of commands such as:

Let the letter A signify the detection of a right hemisphere task
Let the letter B signify the detection of a left hemisphere task
Let the letter C signify the detection of a resting condition for x seconds

Which could be translated into a command sequence such as:

AC = Stop
CAB = Go
BAC = turn right 90 degrees etc…

Subjects were given five mental tasks to perform. The first was a baseline measure of activity and involved the subject relaxing and trying to think of nothing at all. The second task was a non-trivial multiplication problem to be solved mentally. For the third task, a drawing of a three-dimensional object was given to the subject to study for 30 seconds, after which he or she was instructed to visualize the object being rotated about an axis. Mental composition of a letter to a friend was the fourth task. The fifth task required the subject to visualize numbers being written on a blackboard sequentially, with the previous number being erased before the next was written.

In all of the tasks the subjects were instructed not to vocalize or move. EEG information was detected from a number of electrodes positioned across the surface of the scalp. Further electrodes detected signals generated by muscle and eye movement and these signals were eliminated, in real-time, from the EEG data.

A Bayes Quadratic Classifier (BQC) was used to test task accuracy against feature sets created from estimates of the spectral density of EEG in four frequency bands – delta, theta, alpha and low/middle beta. Keirn and Aunon were surprised to find that after a training period, the classifier could distinguish between all five tasks, with an accuracy of over 95% based on only 2 seconds of data. Unfortunately, Keirn and Aunon appear to not have pursued their studies beyond this point.

In 1994 Pfurtscheller et al (Pfurtscheller 1993, Pfurtscheller 1994) used mu-rhythm blocking in the development of a system known as the GRAZ Brain Computer Interface. Mu is a distinctly pattered movement-related brain signal appearing within the alpha frequency band[17]. Khulman (Kuhlman 1978) had previously established that contralateral[18] mu-rhythm blocking (i.e. notable drops in mu signal amplitude) occurred during the 1-second period prior to motor activity. Pfurtscheller's group set out to establish whether it was possible to distinguish between three distinct physical actions by studying the brain's mu rhythm responses alone.

For the study, 30 electrodes were applied in a rectangular array over the left and right post-central (roughly beneath the top part of the skull) areas of each subject. Features related to mu-rhythm blocking were extracted from the EEG data stream and passed to a pattern classifier. An initial training period for each subject involved using data from all 30 electrodes to train a Learning Vector Quantifier (LVQ) artificial neural network.

During a trial, subjects were visually cued to perform one of three movements.

1. Press a micro-switch with the right index finger
2. Press a micro-switch with the left index finger
3. Flex the toes of the right foot

[17] Alpha is detected over the visual cortex of the brain, whereas mu activity is detected over the motor cortex.

[18] Occurring in the brain hemisphere opposite the limb moved i.e. left hand produced mu-blocking in the right hemisphere of the brain

After the visual cue appeared, a period of 125mSec followed during which the subject was to mentally prepare for movement. After this period, the cue disappeared from the screen and the subject was expected to start the movement. As is the case with pattern classification based on neural networks, each movement had to be repeated by a subject at least 30 times before the system was able to recognise it.

Khulman's findings were confirmed – that the final step in this cycle (the actual movement) is not necessary for the invocation of mu-rhythm blocking. It is the *planning* of movement that elicits the response. Discrimination between right and left finger movement using information within the 10-12 Hz frequency band alone was possible to an accuracy of 83%. Discrimination between all three different movements revealed a less impressive classification accuracy of between 51 and 58% using the alpha component alone. However, when the classifier was reconfigured so that it was looking at data from alpha, beta and gamma bands together, the accuracy rate improved to between 62 and 70%.

Pfurtscheller's study has recently been repeated by researchers at Imperial College who have created their own neural-network-based system for processing movement-related brain responses (Penny 1996a, Penny 1996b, Penny 1998). Their first system relied on processing data from a 22-electrode array. As with Pfurtscheller's study, subjects were to repeat a particular task (movement of the right hand for instance) at cued intervals. The aim was to train the network to recognise from the EEG pattern of activity when movement of the right hand was occurring. The second task subjects were instructed to undertake was to *imagine* the same movement, again on cue. The results from this study again confirmed that these imagined movements produce exactly the same patterns of EEG activity as the actual movements.

Due to the amount of data processing that must take place in order to decipher the information from the EEG detected from 22 positions on the scalp, systems such as these have to process their data off-line. A second simpler system was subsequently developed by Imperial, which relied on only 3 electrodes and carried out real-time, on-line signal processing on the detected EEG data. This data was used to demonstrate 1-dimensional cursor control on a computer monitor. The activation mechanism required the subject to use mental imagery to produce two discernible EEG patterns. The first task was motor imagery (e.g. holding a glass) and the second was mental arithmetic. Recognition of one task was translated into upward vertical motion of a cursor, recognition of the other into downward motion.

3.3.2 Biofeedback Conditioned Brain Signals

What separates the following brain-computer interface systems from those of the previous section is that now both the user and the computer are actively engaged in the process of producing useful brain activity through the process of biofeedback. The computer receives an EEG signal stream and feeds it back to the user in real time. The user concentrates on this feedback data, and attempts different strategies in order to learn to consciously influence the signal. The computer subsequently uses the consciously controllable brain signal to operate a cursor or navigate around a soft keyboard. In fact, in the majority of these systems, there is no separate biofeedback training session, the user is presented with the task interface immediately and uses the action of the brain-controlled cursor (or alternative on screen representation) in order to associate their efforts with movement of that on-screen representation.

Biofeedback conditioned EEG component amplitude-related cursor control was first demonstrated by Wolpaw et al (Wolpaw 1990). The EEG component in question, as with Pfurtscheller's system, corresponded to motion-related mu activity. Wolpaw related the amplitude value of a subject's mu rhythm to the position of a cursor on a computer display. Raw EEG data was collected from two electrodes positioned over the motor cortex of the brain. A digital signal processing board digitised the subject's EEG and performed real-time frequency analysis on every 333msec segment. This provided the segment's mu rhythm power, which was relayed to a computer that calculated the amplitude of the mu rhythm.

This detected amplitude value was then compared to 5 pre-set voltage ranges, i.e. 0-1μV, 1-2μV, 2-3μV, 3-4μV, >4μV, and on the basis of this comparison the amplitude value was translated to one of 5 possible cursor positions along a vertical axis. The voltage ranges were pre-set so that high amplitudes moved the cursor upwards, and low amplitudes moved it downwards. A target was placed in the screen, alternatively at the top for one trial and then at the bottom for the next.

During a trial, the subject was required to control their mu rhythm amplitude in order to move the cursor either up or down to meet the target. Successful interception between cursor and target resulted in a visual reward in the form of a checked pattern flashing over the target. The system then paused for 1 minute after which time the cursor would reappear in the centre of the screen and a new target would appear at either the top or bottom. Subjects were not explicitly instructed on how to control the signal, but were encouraged to experiment during early sessions with different strategies. They subsequently reported thinking about physical activities, such as lifting heavy weights to move the cursor downwards and thinking about relaxing to move the cursor upwards (the success of these strategies again support Khulman's assertion that simply thinking about physical activity attenuates mu rhythm amplitude). At the end of approximately twenty 25-minute training sessions, 4 out of 5 subjects achieved success rates for hitting the target of over 80%. In these latter sessions subjects were successfully hitting between 10 and 30 targets per minute. Of particular note is the fact that as the training progressed subjects reported that they no longer had to evoke mental images in order to control the amplitude of the signal/ position of the cursor.

The success of this initial investigation led to a further study, which looked at the possibility of simultaneously training and subsequently using two EEG components for two-dimensional cursor control (Wolpaw 1994). In this second study, two pairs of electrodes were placed over both the left and right hemispheres of the brain, proximal to the motor cortex. This effectively provided two separate channels of EEG data, out of which could be filtered two separate mu rhythm signals. A linear equation transformed the sum of the two mu rhythm amplitudes (right plus left) into vertical cursor movement. Low sums moved the cursor down, while higher sums moved the cursor up. A second equation transformed their difference (right minus left) into horizontal cursor movement. Lower differences moved the cursor to the left, while higher difference values moved it to the right.

This time the targets were placed in the corners of the display After approximately seven 1-hour training sessions, 4 out of 5 subjects achieved target hitting success rates of between 57 and 70%, with a hit rate of between 16 to 26 targets per minute. And as in the earlier study, initial control strategies involving imagery were no longer required as training proceeded. Perhaps most significantly of all, subjects in this second study were seen to hold brief conversations during cursor movement, without detrimental effects on control performance.

The latest exploration into biofeedback based brain-computer control is the Thought Translation Device (TTD) created by Kübler et al (Kübler 1999). This system is designed to transform slow cortical potentials (SCPs)[19] into two-dimensional cursor control on a computer screen. The aim in creating this technology was to produce a non-muscular control technology for three amyotophic lateral sclerosis (ALS)[20] patients. These patients along with thirteen able-bodied subjects were trained to consciously produce changes in SCPs, lasting for between 2 and 4 seconds. Subjects undertook an extended training period (approximately 200*10-minute sessions), where their aim was to move a box over a ball that was fixed in position on the screen. Once accuracy in cursor control tasks was observed to be around 70% subjects moved to a text-input task.

The interface to the text-input task contained alphabetic characters split across the two halves of the screen. In order to select a letter, subjects were instructed to move the cursor to the half of the alphabet containing the chosen letter. That half of the alphabet was then split again across the two halves of the screen and again the subject was to select the half containing the chosen letter. This design required a subject to make up to five selections in order to choose single letter. Not surprisingly, letter transmission rates were found to be low (anything up to 192 seconds per character). However, this technology has already proved itself invaluable to two of the disabled patients whose only means of communication it now is.

3.3.3 Discussion

The ultimate aim of this thesis is to discover how best to support the development of all electrophysiologically interactive computer systems (EPICS). Neural interface systems developed to explore direct brain-computer communication are a subcategory of EPIC systems, with examples of NITs falling in both of the EPICS categories defined at the beginning of this chapter. As an EPICS subcategory, brain-computer interface systems all include the three defining EPICS features identified above, namely physiological signal detection, pre-processing and presentation.

What is not explicit in the preceding description of NITs is the fact that signal detection is most often carried out using proprietary hardware configurations of sensors, filters and amplifiers. Despite this, signal detection occurs in exactly the same manner as with commercially available,

[19] SCPs are of a different nature to EEG signals, although detected and processed in a similar manner

[20] ALS is a degenerative neurological disease which eventually leaves its suffers totally paralysed

multi-signal sensing devices, with filtered and amplified EEG data being made available to a computer for further signal processing and presentation.

What, we can say is that neural interface technologies require a higher level of pre-processing than is necessary for more general biofeedback-based systems. Apart from simple switching systems such as Dewan’s Morse code system, signal processing for direct brain-computer communication requires the utilisation of learning algorithms or neural networks. Any tools conceived to support the future exploration of brain-computer interaction must be flexible enough to incorporate adaptive algorithms where required.

One feature common to all of the BCIs described is the task-specific nature of their interfaces. Where some biofeedback applications were seen to incorporate highly abstract signal representation mechanisms, such as disembodied musical tones, BCIs require meaningful, goal-driven signal representation. A number of the systems described present soft keyboards as intermediate interfaces to speech synthesisers for physically disabled users for example. Key choice can be facilitated through a variety of mechanisms, from flashing keys which provide detectable brain responses, to EEG controlled cursor-based selection.

Beyond presentation of a keyboard, interface requirements can be seen to be quite diverse, with exploratory systems such as Keirn’s requiring an interface that displays the results from a network classifier not to a subject, but to a professional observer. At first glance, both the GRAZ and Imperial College BCIs focus on the classifying of brain signals, without much of a role for the systems’ interface. However, in both of these systems the interface plays a vital operational role in cueing the user to perform actions at intervals suitable to the processing capabilities of the neural network classifier. Common to all three of these exploratory systems, the ability to recognise mental tasks or imagined movements has interesting implications for EEG-driven computer control, especially for disabled subjects.

Wolpaw’s and Kübler’s systems, like the CyberLink evaluation, all require setting up of a feedback loop between the user and the system in order for the user to condition their EEG components appropriately. The form that the feedback takes in each of these instances is related directly to the final hands-free control task – move a cursor, navigate a soft keyboard, etc.

An obvious issue within the description of these systems is the amount of time spent training within the task space before useable control is achieved. This training time can be seen to be in the order of tens of hours[21] and time and again training can be seen to take place within the task space. There is little flexibility in terms of altering the task in order to maintain the interest of the subject. We earlier considered motivational issues surrounding clinical and medical applications of biofeedback-based training and the importance of maintaining interest and thus a subject's motivation to learn is equally relevant to EEG-based hands-free control training. Only Kübler's system provides a specific training task that a subject can use separately in order to experiment with control strategies before moving onto to using the final interactive interface.

Issues such as extended training times, the potentially limited nature of the applicability of direct brain to machine communication, and the unknown effects of adopting this method of human-machine interaction in real-world situations may discourage many from pursuing the dream of thought-controlled machines. Indeed, when fully explored, brain-computer interface technology may be found to be applicable only in limited situations. However, the importance of continuing exploration of NITs for users with limited physical capabilities in particular is immeasurable.

It is fair to say that developing enabling technologies for the disabled is often a low priority concern for software developers. This is at least part of the reason why few tools exist to support the development of assisitive technologies[22] generally. But the implications of hands-free control for physically disabled users is perhaps reason enough to engineer tools that support the further exploration and development of electrophysiological hands-free human-machine interaction.

Before we can discuss the form that tool support may take, it is necessary to complete our exploration of EPIC applications by looking at the second major category of electrophysiologically interactive computer systems defined within this thesis, namely **monitoring EPICS**.

[21] The issue of protracted training times is discussed in Chapter 7

[22] A collective terms for aids for the disabled

3.4 Monitoring EPICS

In this section, interactive systems that fall into the second of our two main EPIC categories are described. This description serves to augment the list of system features thus far identified as being common among EPIC systems generally and therefore suitable for tool support. Monitoring EPIC systems provide physiological information to support the active observation of a subject's physiological state. As we will see, analysis of that physiological data may be undertaken by a psychophysiologist or by an EPIC system itself.

3.4.1 Physiological Monitoring of Complex System Operators

Although monitoring human physiology for health purposes is commonplace and acceptable, monitoring the physiological state of a complex system's user is potentially more controversial (Hagan 1995). Nevertheless, monitoring the physiological state of the system user can provide a range of useful information to either a running interactive system or to the designers of such a system.

In order to monitor the changing state of any system, it is necessary to identify and then observe a suitable range of parameters whose behaviour actively reflect changes in that system's state. We have already encountered, physiological parameters that can be used to monitor changes within the human system. Furthermore, we are now familiar with the kind of equipment required to make physiological information available to a computer in the first instance and perhaps thereafter to an interested observer[23]. With both the knowledge and the tools for physiological monitoring available, it is useful to look at situations where operator monitoring can be applied.

As we progress through this description of EPICS conceived to monitor subject state, it will become apparent that monitoring EPICS share their major functional features with training EPICS in that both categories of system detect, pre-process and present electrophysiological information. However, it should also become clear that monitoring EPICS are concerned less with signal presentation than with signal pre-processing and that this difference in focus is the feature that defines a monitoring EPICS from a training EPICS.

[23] A psychophysiologist

In order to convince the reader of the growing role for physiological monitoring within interactive applications we begin by considering a situation were the availability of *any* information about the real-time physiological condition of a complex system's user could improve that system's ability to operate safely.

3.4.2 Is the Pilot Dead?

Following an aircraft crash in 1979, the co-pilot submitted a report that described his attempts to alert the pilot to the fact that the plane had begun to lose height. The pilot had failed to respond and as this was not an unusual reaction for this particular individual, the co-pilot failed to notice that the pilot was, in fact, dead from a heart attack (Norman 1992). Fortunately the cockpits of commercial aircraft are usually staffed by up to three personnel, increasing the chances not only of someone noticing if the main pilot becomes incapacitated, but also of there being another trained pilot who can assume command of the aircraft. But what happens in situations where there is no other operator available to either check the state of the main system operator or take over control if they do become incapacitated?

Who monitors a light aircraft or fighter aircraft pilot, for example? A crane driver? A heavy plant operator? Or even a car driver? These systems are all configured to rely, to some greater or lesser extent, on a human operator to perform tasks vital to their continued safe operation. Worryingly the answer for the time being is that no one is checking on the state of the human operator and in each of these and many other situations. Sudden incapacitation of the operator would leave each system (car, aircraft etc.) in an unpredictable and potentially dangerous condition.

If a system were able to monitor the state of its human operator (in the same way that the human operator continually monitors the state of a complex system) then the system would be in a position to react if, for any reason, its operator lost consciousness. We have already established here that such information can easily be made available to a system. By embedding the means to continually detect and process physiological information about the operator, the system could use the results in order to make decisions about its own operation. The operational possibilities afforded by embedded physiological detection capabilities are illustrated in Figure 3.5.

We have already spoken at length about physiological signal detection. Once physiological data about a suitable set of parameters is made available to the system, then real-time assessment of the data can indicate the current physiological condition of the operator. While the operator's physiological signals remain within certain boundary values, the system will continue to operate normally, expecting to periodically receive input from the operator. If at any time the physiological data analysis indicates that something is wrong with the operator, the system can take some action. The form this action takes is context-dependent but would probably involve the system bringing itself to some safe state or alerting a third party about the condition of the operator.

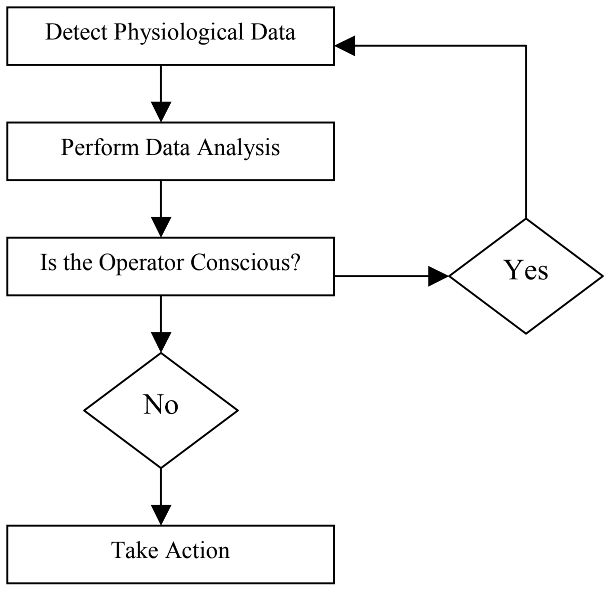


Figure 3.5 Embedded consciousness monitoring

3.4.3 Hazardous States of Awareness

Total loss of consciousness of a system's operator is a relatively rare occurrence. More common is an operator experiencing a shift in attention that could ultimately be detrimental to their normal operation of a complex system. Examples of psychological states that can effect the performance of a system's human operator include stress, high anxiety, boredom, absorption, fatigue and inattention. Pope et al. (Pope 1993) has classified these conditions as being *hazardous states of awareness*.

One practical example of operator monitoring can be seen in alertness alarms that employ cameras to monitor the eye blink rate of a car's driver (Carnegie 1999). As blink rate is known to increase in direct relation to increasing levels of fatigue, this monitoring system alerts the driver if their eye blink rate indicates that there is a danger of their falling asleep. An alternative, electrophysiological method of blink rate detection involves detecting EMG activity associated with the blinking action itself (TDC 1996). This method relies on the detection of fluctuating electrical activity in the orbicularis oculi[24], via sensors positioned at the outer corners of each eye.

The series of events - *detect, analyse, action* that underlie the operation of an alertness alarm are also found in many continuous monitoring systems developed for medical applications, ECG monitors for example. The key feature of such systems is that activation of some event within the system's interface is dependent upon a particular physiological parameter crossing a critical boundary value. In monitoring EPICS, the act of analysis is an act of pre-processing that takes place within the monitoring application. The *action* event is a feature of the interface regardless of whether the recipient of the information is the monitored subject (the driver hears the alarm) or another party (the aircraft signals the tower for another human operator to take over control).

Existing alertness alarm systems are prototypes conceived as purpose-built stand-alone technologies that can be positioned and used within a given vehicle setting and usually look at only a single parameter of interest. The next incarnation of alertness monitoring will almost certainly involve the building of embedded, multi-parameter physiological monitoring capabilities. The flexibility that such systems afford researchers is that they can be used to detect any of the hazardous states of awareness listed above.

In a bid to identify the physiological signatures of a range of hazardous awareness states, the Crew Hazards and Error Management (CHEM) project at NASA have been developing their own integrated technologies for monitoring human state.

Pope's analysis of data relating to aviation safety (NASA 1999a) has shown that flight crew often report mistakes made as being directly related to the psychological state they were experiencing at the time. During simulated flight tasks, information can be gathered about a subject's changing

[24] The muscle configuration surrounding the eye

physiological state. This information can either be presented to an experimenter in real-time for on-line analysis or can be archived for later off-line analysis. The underlying operation of the physiological monitoring for awareness states is illustrated in Figure 3.6.

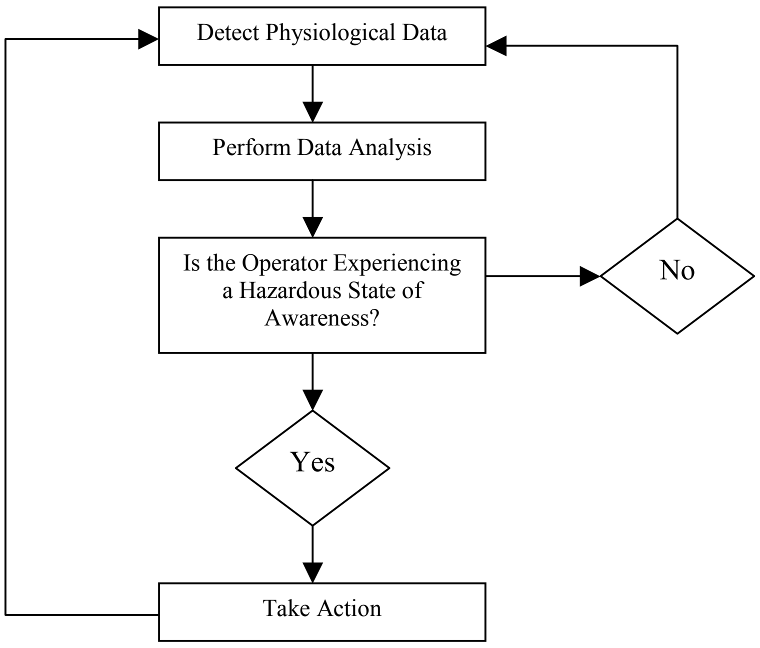

Detect Physiological Data
Perform Data Analysis
Is the Operator Experiencing a Hazardous State of Awareness?
No
Yes
Take Action

Figure 3.6 Embedded hazardous awareness state monitoring

This is obviously a similar pattern of activity to that shown for consciousness monitoring systems, the only differences being that the action occurs on a positive response to the state information question and the system continues to monitor the user after responding to a detected negative physiological state change. Consciousness modelling and awareness modelling are really asking the same question however, namely is something wrong with the operator? A positive answer to this question leads the system to a position where it can, if required, assume a state of autonomy from the operator. In the latter case, this would only happen if the subject physiology indicated a continuing hazardous awareness state.

What cannot be made explicit in flow diagrams are the temporal aspects of a system's operation. Consciousness monitoring must take place within a real-time system, which requires a time-critical response to the discovery that something is wrong with a part of that system (i.e. its operator). The reason that this is an important point to stress is because the temporal constraint implicit in consciousness monitoring systems is not present in all hazardous awareness state research currently being undertaken.

Some classes of awareness monitoring, such as the alertness monitors described, are obviously time-critical applications whose *action* (alerting a driver that he/ she is in danger of falling asleep) is only valid if carried out within a small time window (just before he/ she actually falls asleep). However, many of the current investigations related to awareness monitoring are carried out *off-line*, in other words from archived data. This is because researchers are still at the stage of trying to discover whether it is indeed possible to recognise the physiological signatures of different awareness states identified as being potentially detrimental to the performance of a system's human operator.

Interestingly, an increase in the level of automation in human-machine systems has been identified as a major contributing factor to the experiencing of hazardous awareness states by systems operators (Hockey 1989). At the extremes of human-machine interaction, two models of interaction exist which are equally likely to cause a system operator to experience undesirable states of awareness. At one extreme, the role of the human operator of a system is simply to monitor information pertaining to the state of that system. While the state information indicates that the system is functioning normally, the human operator plays no active role in the operation of the system. Prolonged periods during which the operator has few challenging tasks to perform can lead to boredom, complacency and reduced vigilance (NASA 1999b). The generally accepted term for the condition experienced by operators of highly automated systems is *task underload*.

At the opposite extreme of human-system interaction, a poorly designed system may impose upon its user an excessive amount of cognitively challenging tasks. The operator of such a system is prone to experiencing stress, feelings of anxiety and subsequently fatigue. This is a situation known as *task overload*. Physiological monitoring may be used in order to design systems with the correct mix of tasks to hold the operators interest without overloading them. So let us examine the issues surrounding physiological measurement and monitoring as an indicator of task-related cognitive load.

3.4.4 Measuring Cognitive Load

Identifying optimal mixes of operator and system functionality is obviously an important part of the design of any interactive system. Cognitive psychologists involved in both the design and evaluation of complex systems have developed a suite of techniques for assessing the amount of mental effort involved in performing operational tasks. One of the most popular techniques is performance-based assessment, which studies the ability of a user to carry out two cognitively challenging tasks simultaneously. An alternative assessment technique concentrates on monitoring particular physiological characteristics known to be responsive to cognitive loading whilst a user performing a particular task.

It seems obvious that cognitive task load should be amenable to a measure of task related brain signal responses and indeed since the 1980s psychologists have been studying brain activity as a means to identify task-related changes in cognitive load. The focus for many studies (Horst 1984, Isreal 1980, Kramer 1987, Natani 1981, Strayer 1986, Wickens 1983) has been the P300 response, which as described above is an event-related brain response (see Section 3.3.1).

A further popular physiological correlate of cognitive load is heart rate variability (HRV). In 1995 Tattersall and Hockey (Tattersall 1980) monitored the HRV of flight engineers undertaking tasks during a simulated training flight. Tattersall knew from previous studies (Hart 1990) that heart rate is sensitive to both physical and emotional stress and could therefore be expected to be elevated during the stressful takeoff and landing phases of a flight (which indeed was their observation). Their findings indicated that flight tasks that involved mentally demanding problem solving caused HRV to be markedly suppressed, whereas less demanding routine tasks did not impact on normal HRV. These findings support those of other studies (Aasman 1987, Kramer 1991, Mulder 1987, Vincente 1987) indicating the suitability of HRV for identifying changing levels of engagement during concentrated information processing.

Issues that have affected the widespread take-up of physiological assessment as an evaluation technique are less to do with required expertise, than the previously unavailability of suitably portable monitoring technologies (Wastell 1999). Before the creation of integrated electronic sensing devices, cumbersome configurations of amplifiers, meters and separate recording media limited the applicability of psychophysiological monitoring to the laboratory. Integrated sensing technologies of the kind discussed in Chapter 2 have made this less of an issue. Indeed, one of

NASA's systems, known as CREW (Crew Response Evaluation Window) includes physiological stress monitoring and brainwave signal processing along with eye-tracking and video information (NASA 1999b) with a view to assessing mental states, task engagement and stress, as well as mental loading and situation awareness. Data is gathered about a subject undertaking flight-related tasks and is assessed off-line in order to discover how those tasks affected the behaviour and physiology of the subject.

A further NASA system, called the Closed-Loop Biocybernetic system (Pope 1995) extends the monitoring and display capabilities of CREW so that the physiological and behavioural information gathered by the system can be presented in real time to a trained observer. The observer performs continuous real-time visual assessment of that data, and is able to dynamically reconfigure the interface in order to match the changing awareness condition of a subject (Figure 3.7). The aim of the project is to demonstrate whether negative psychological states can be counteracted through dynamic reconfiguration of human-computer interfaces.

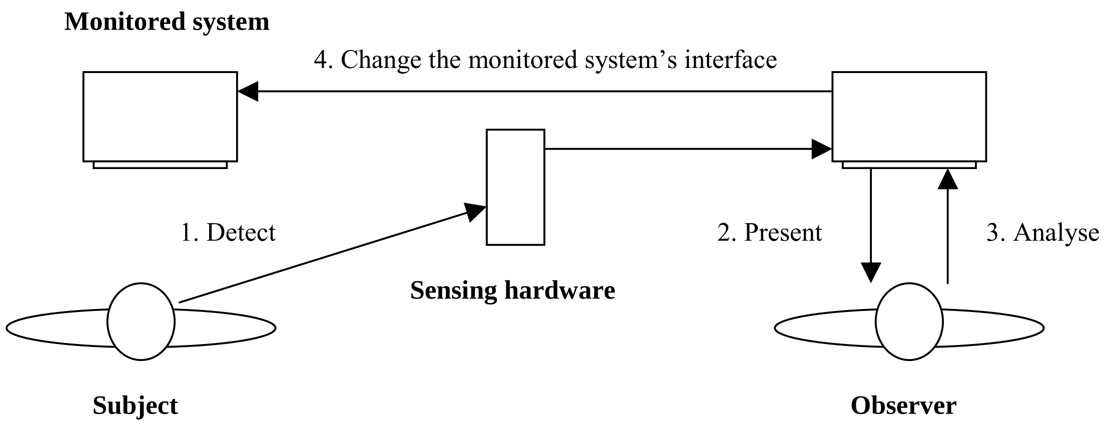


Figure 3.7 The Closed-loop Biocybernetic system

3.4.5 Affective Computing

A further exciting possibility for integrated monitoring technologies is the development of affective computers (Picard 1997) - literally computers that know how their users feel. It has long been the goal of artificial intelligence (AI) researchers to develop systems that behave autonomously, effectively displaying intelligent behaviour. However, Goleman (Goleman 1995) speculates that intelligence is far more than mere independent cognition. His thesis is that intelligent human-to-human exchanges are influenced by empathetic and emotional information transmitted between individuals through body language, facial expression, vocal intonation etc. This is one of the reasons why some AI researchers are beginning to consider how computers can gather suitable information about their user's changing physical and physiological condition from which they can extract emotional state information thereby improving their ability to display truly intelligent behaviour.

On the 17th of February 1998 the front page of the Independent newspaper announced the arrival of "*The computer that can hack into your emotions*" (Schoon 1998). The article described the work of researchers at MIT who have been exploring the possibilities of training computers to recognise and respond to human emotions. It was from this group's work that the term *affective computing* emerged as a classification for emotionally responsive, artificially intelligent interactive systems.

Unsurprisingly, work carried out within MIT's Affective Computing Group is not very different from NASA's research into hazardous awareness state detection and avoidance. Based on the premise that emotional states are accompanied by detectable changes in physiological functioning, affective computing research aims to train computers to recognise a human psychophysiological state through its physiological signature. So far the Affective Computing Group have had limited success in their attempts to recognise *frustration* through off-line examination of features extracted from a subject's GSR and blood pulse volume (BVP) using Hidden Markov Models (Fernandez 1997). Other studies have attempted to discern between a range of emotions using a wider range of physiological markers, including not only GSR and BVP, but also EMG, heart rate, heart rate variance and respiration rate (Healy 1997).

The results from studies based on physiological signal monitoring alone have indicated that although physiological monitoring it good for detecting levels of arousal (i.e. the strength of various physiological parameters), it is no use for detecting *valence* – the positivity or negativity of an emotion (Archer 1999). So, for example joy and grief both produce high physiological arousal, but without additional information it is not possible to discern between the two emotional states even though the first has a positive valance and the second has a negative valance. In order to address this problem MIT are now looking to supplement the physiological evidence of emotion by providing more cues to the computer. Their hope is that voice inflection, facial expression, posture and gesture recognition will provide enough information for the system to make a more accurate identification of the emotional state of the user. It is also probable that affective computers will require both environmental and task related data in order to make decisions about emotional state in context.

3.4.6 Discussion

This description of monitoring EPICS enables us to consider further the features that need to be supported by EPIC-specific development tools. We began this section by indicating that the major difference between monitoring-based EPICS and training-based EPICS is one of focus, with monitoring EPICS concentrating on signal pre-processing over signal presentation. In each type of monitoring EPIC application described in this section, physiological information made available to a system through suitable sensing equipment undergoes some form of analysis in order to help determine the physiological and psychological state of the monitored subject. Data analysis is a form of signal pre-processing which, as shown, can be either carried out within the EPIC system itself or by human observer. The use to which the state information derived by this analysis is put depends upon the final application.

Table 3.1 maps different monitoring applications against both the potential modes of physiological signal analysis and the use to which the derived psychophysiological state information is to be applied. So for example, consciousness monitoring requires on-line data analysis to be undertaken by the EPIC system itself. The destination for the resulting state information is also the EPIC system, which uses the information to gauge whether it needs to take any action in order to maintain its continued safe operation.

	Data Analysis			Data Presentation	
	On- /Off- line	EPIC	Other	EPIC	Other
Consciousness Monitoring	On-line	*		*	
Hazardous Awareness Monitoring	Off-line		*		*
	On-line	*	*	*	
Measuring Cognitive Load	Off-line		*		*
Affective Computing	On-line	*		*	

Table 3.1 Data Analysis in Monitoring EPICS

As described above, there are two models for what happens to data in hazardous awareness monitoring. Off-line processing of physiological (and other, related) data can be carried out by a human observer. This requires the archiving of physiological data for later analysis. In order to assess what situational features may have initiated the shift to a hazardous awareness state, it must be possible to analyse the data alongside task-specific and situational information. This can be achieved through time-stamping each physiological data point. The resulting state information gathered in off-line analysis is destined to feed into the overall systems design process.

In systems such as NASA's Closed-Loop Biocybernetic system, a trained human observer[25] can carry out on-line analysis on the data made available through a monitoring system. By assessing the changing state of a subject in real-time, an observer can instigate dynamic changes to the interface between the human operator and the system in order to counteract negative awareness states.

We can see from Table 3.1 above that measurement of cognitive load involves the same pattern of activities as hazardous awareness monitoring, with data analysis being carried out off-line by a human expert with the resultant information feeding into the systems design process. This is, as we would expect, given that Pope's definition of hazardous awareness states includes psychophysiological conditions that one would associate with cognitively-overloaded task configurations.

[25] In the near future we can expect this assessment to become partially or totally automated

Finally we can see that affective computing applications share a data analysis and result destination profile with consciousness monitoring applications. While the consciousness or emotional state of the user is stable, the system continues its operation as normal, but a change in state requires a dynamic change in the actions of the system. The ability to automatically respond and adapt to the changing state of the user will undoubtedly be the next step for hazardous state-aware systems which will dynamically reconfigure their own interfaces to suit the changing cognitive requirements of the user. Thus the differences inherent in current incarnations of each example of physiological monitoring systems can be seen to be converging onto a model of increased system autonomy facilitated through real-time analysis of human physiological information.

The development by commercial organisations of computer peripherals designed specifically to collect physiological information (Ark 1999, Hinckley 1999) can be taken as an indication that research into this challenging new field of human-machine interaction is destined to expand. In order to support the potential explosion in EPICS-related research, this thesis aims to provide both contextual information for future EPIC system designers and support for the process of development of physiologically enhanced interactive applications.

3.5 EPICS Architectures

For the purposes of establishing the development support requirements for systems such as those described in this chapter, an assessment of the overall architectures of these systems is required. As demonstrated by the description of biofeedback-based applications in particular, EPICS support direct manipulation of common interaction techniques (menus, buttons, dialogs etc.) in order to create suitable interfaces, manipulate information for analysis purposes etc. These more traditional human-system interactions take place via conventional input devices (mouse and keyboard). In addition to this, EPICS require continual access to human physiological information available through specialised sensing devices of the kind described in Section 2.2.2.

The majority of contemporary interactive computer systems present information to their users via windows-style application interfaces. Unsurprisingly therefore, the classic WIMP (windows, icons, menus pointers) style of human-machine interaction predominates in the overall design of many of the EPICS technologies we have described. The development of interactive systems based on the WIMP interaction paradigm is fully supported through a huge (and growing) range of interface toolkits, frameworks and integrated development environments (IDEs). Included in these development tools are support for common interaction techniques and interaction devices. This final section considers the use of physiological sensing devices as additional computer interaction peripherals which are currently unsupported in existing development tools.

Figure 3.8 shows a simplified architecture for the physiological signal-handling portion of a typical EPIC system designed for biofeedback-based training. A software device driver (usually a dynamic link library or DLL) provides a means of communication between the EPIC application and the sensing device. Once a connection is established between the application and the sensing device, the device makes physiological events (corresponding to new physiological signal data points) available to the application. For the purposes of illustration, the sensing device shown in Figure 3.8 provides its application with information about four different physiological parameters

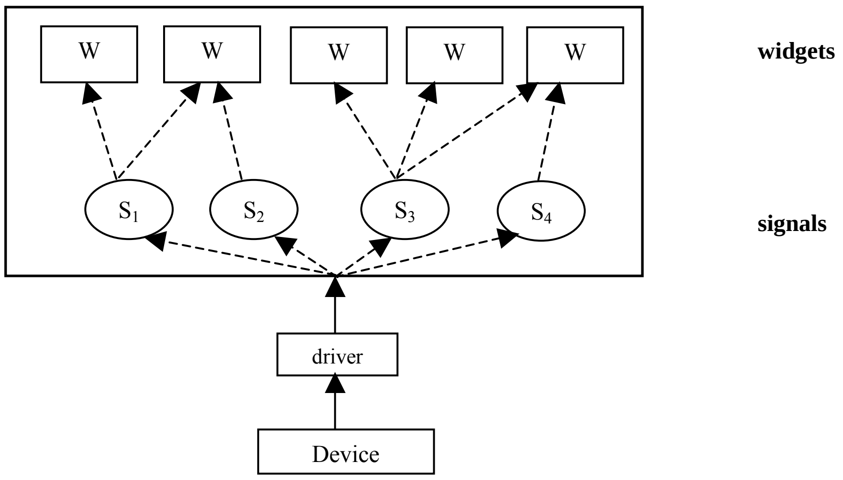


Figure 3.8 Training system architecture

labelled S_1 to *S4*. Within the EPIC application, the signal points are passed to EPIC-specific widgets (suitable audio/ visual displays) which are used to display signal information in different formats according to the requirements of the training application.

Figure 3.9 shows how the software architecture for the training application is embedded within a conventional windowing application. The larger windows application provides access to conventional interaction techniques via conventional input devices. Thus the event queue for the larger windows application contains discrete events provided by conventional devices interleaved with the stream of events generated by the physiological sensing hardware indicating the availability of new physiological signal data points.

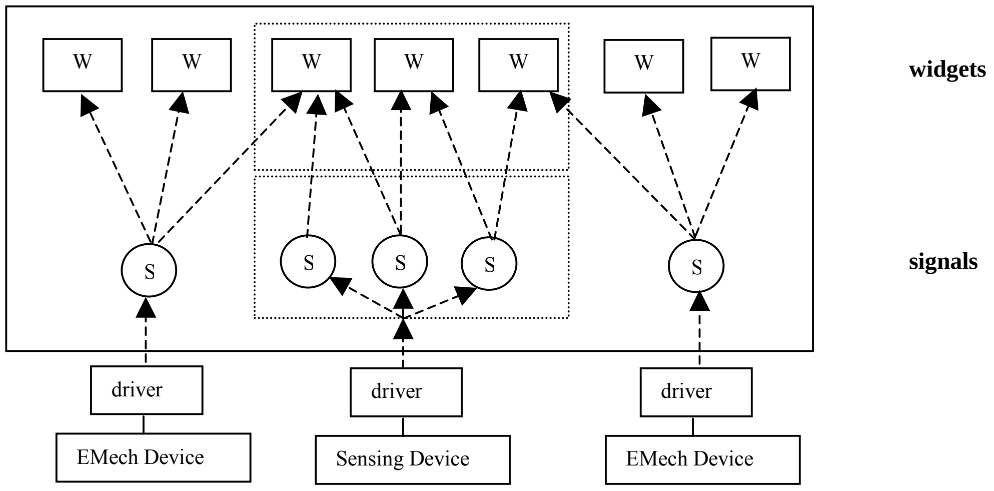


Figure 3.9 Embedded training architecture

The second software architecture, illustrated in Figure 3.10, is that of a monitoring EPIC system. The hardware and signal generation requirements remain the same as in the training application, and EPIC-specific interface widgets are used to display information where required. The difference here is that there is often some intelligence embedded within the system that tries to

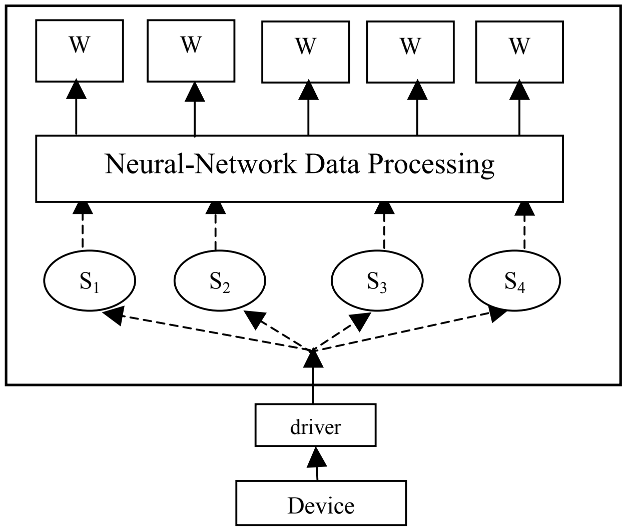


Figure 3.10 Monitoring system architecture

make sense of the physiological data stream. As described in previous sections, this “intelligence” takes the form of a learning algorithm or neural network, which attempts to identify features from within the physiological data stream. As with the training EPIC architecture, the monitoring EPIC architecture is itself often embedded within a traditional windows-style application. This is shown in Figure 3.11, which also illustrates the fact that monitoring systems are usually run in parallel to some other interactive system. The role of the EPIC system is to monitor the physiological state of a user while they are interacting with the second system. Furthermore, it is usual for the application being evaluated, i.e. used by the monitored individual, and the EPIC application to be running on separate computer systems.

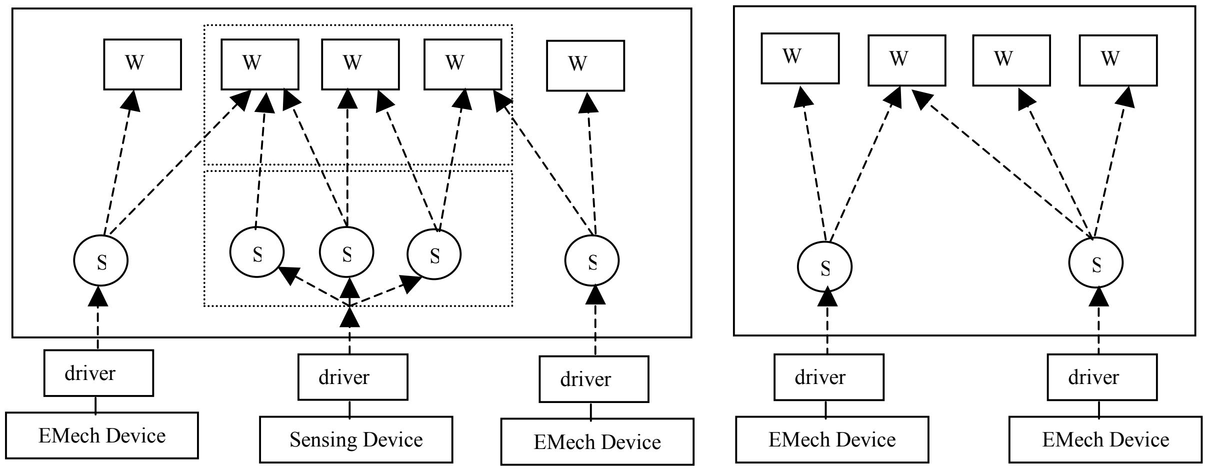


Figure 3.11 Embedded monitoring architecture and system being evaluated

3.5.1 Discussion

The inclusion of an overview of EPICS architectures serves to illustrate that many EPIC systems are constructed within the constraints of a windowing model of interaction. This is particularly true of clinical biofeedback training applications such as the BioGraph (see Section 2.4). There are a couple of reasons why developing EPICS within the confines of an interactive windowing application is an attractive option for EPIC system developers.

Firstly, windowing systems provide the single most popular form of human-computer interaction. One benefit of this is that any interactive system based on application presentation and interaction through windows will be at least partially familiar to a new user. As the operators of many training and some monitoring EPICS are clinicians and psychophysiologists (largely noncomputing experts it is safe to assume), they will better be able to learn to use a system which presents physiological information in this manner. Dividing a screen into independent windows is also a particularly appropriate way of simultaneously presenting real-time information about different physiological signals.

Perhaps the main reason however for the popularity of building windowing applications is the wide availability of tools to support their development. The major part of any windowing application can be constructed rapidly from pre-existing interaction techniques (available as code fragments). Interaction techniques are discussed further in the following chapter, but briefly what is being considered are popular interface entities, such as menus and buttons, along with actions required to operate on the interaction techniques. Windows-based interaction techniques can be used to construct interactive applications that can, for example, operate with any one of a number of virtual pointing devices which includes the mouse, joystick and trackpad.

Identification and provision of EPICS interaction techniques enables us to support the notion of virtual physiological sensing devices. The availability of EPICS-specific interaction techniques provides a means of abstracting over different physiological sensing devices. Provision of a device independent software toolkit is one of the objectives laid out in Chapter 1. The issues surrounding toolkit support in general and how best to provide support for EPIC systems in particular, is the focus of the following chapter.

Using development tools for windowing systems, an application developer can quickly build the skeleton of an EPIC application adhering to a general windows style and thereafter hand-code the parts of the application specific to the retrieval, processing, utilization and presentation of the physiological information. But as we have seen, this leads to sensing device-specific applications, as well as the re-implementation of popular EPIC interface features by each new EPIC application developed.

In order to move toward our goal of developing an EPICS support tool, we need to start considering some of the more popular feedback entities we have encountered (graphs, animations, external devices etc) as the interaction techniques that need to be provided in support tools for future EPIC systems development. Identification of these interaction techniques is an implicit aspect of the objectives for this work as set out in Chapter 1.

Physiological sensing devices differ from the majority of traditional interaction devices in that they provide applications with continuous streams of data rather than a series of temporally discrete events. This makes the management of data relating to user-generated events more of a burden on the system, and if not managed carefully, streaming data can cause all sorts of performance problems both within the application itself, and at the interface between the application and the user.

A further issue to be addressed focuses on the fact that systems which aim to utilize the physiological information directly as a mode of interaction, for example hands-free control applications, require a higher level of pre-processing to be carried out on the incoming physiological data stream. In order to provide maximum flexibility for future system designers we need to consider this issue of different pre-processing requirements within the interaction model. This along with other physiological data-management considerations forms part of the following chapter.

3.6 Summary

In this chapter, we have examined a range of interactive applications which all rely upon the detection, processing and presentation of human physiological information. Some of the areas of research discussed here such as BCI research and affective computing are emerging as fields in their own right. However, it seems logical to identify their subordinate position within a larger classification of interactive systems, based on the common requirements for detection, processing and presentation of human physiological information. We have collectively identified these (and other systems described in the chapter) as being examples of *electrophysiologically interactive computer systems*. This is a new classification of interactive system defined by this thesis.

Within this chapter EPIC systems have been divided into two major categories – training EPICS and monitoring EPICS. As described, the defining feature in training EPICS is the method of signal presentation, as this plays an important part in the feedback loop that must exist between a subject and a system in order for physiological signal conditioning to occur in that subject. Alternately, the key feature in monitoring EPICS is the signal pre-processing stage, where the physiological signal stream was assessed in order for the system to make some assertion about the physiological state of a monitored subject based on the available data.

What we can say about every EPIC application described in this thesis is that they are all proprietary applications. Furthermore, each of the applications described relies on a different sensing hardware device/ configuration to retrieve physiological data for the application. What has hopefully become clear to the reader is that across these systems there are many shared features related to physiological signal detection, pre-processing and presentation. Currently, these features have to be re-implemented each time a new EPIC application is conceived and built.

The overview of system architectures towards the end of this chapter makes it clear that the design of many EPICS has been influenced by the availability of windowing development tools. In order to see how development tools evolve, the following chapter reviews software support tools for the development of traditional windows-based interactive systems. We have already introduced the argument that these tools cannot be easily extended to support the functionality identified as being common amongst EPICS applications, especially the requirement to pre-process and present continuous streams of physiological signal data. This view is reiterated in context as part of the next chapter.

It is clear from the work presented in this chapter that there is a growing role for human physiological information in the future of interactive systems development. The form this work will take depends upon the ease with which EPIC systems can be built and applications tested. With the right tool support a broader range of individuals could undertake research into different aspects of electrophysiological human-machine interaction. The development of such tool support is the raison d'être of this thesis.

The next chapter concentrates on reviewing interactive systems development support tools. As well as examining windowing toolkits that favour discrete input devices, tools for developing applications that support continuous input devices are presented. In the light of these reviews a proposal is made for a component-based approach to toolkit support for EPICS development. A high level model of human-machine interaction based on sensing-based interaction is identified, which provides a foundation upon which we can base the design of the first dedicated EPICS development toolkit, itself described in Chapter 5.

Chapter 4

Interactive Systems Development

4.1 Introduction

Chapter 3 described a range of situations where the ability to detect and process human physiological information is influencing the design and development of electrophysiologically interactive computer systems. Monitoring users physiological state was seen to be informing the design of complex systems in some cases, and providing useful interaction information to working systems in others. Presentation of physiological information via a suitable interface was also shown to be instrumental in the learning of conscious control of aspects of a human physiological functioning. This ability can subsequently be used to facilitate hands-free human-to-machine control. Examples of each of these applications were provided and these systems were classified collectively due to their utilization of detectable human physiology as being electrophysiologically interactive computer systems (EPICS).

It was demonstrated at the end of Chapter 3 that the most common format for presenting physiological information was through a windows-style interactive application. The development of windows-based applications is fully supported through a plethora of development tools. What these tools do not provide, however, is an easy means of incorporating drivers for non-standard input devices, such as those used for detecting human physiology. In addition, these tools do not support the development of interface entities that can handle the streaming data that physiological sensing devices produce. The current approach to EPICS development is to build a windows-style interactive application, and then hand code everything related to the sensing, processing and presentation of physiological information.

This chapter examines popular conceptual models that have emerged to describe the interactions that take place between users and systems that support interactions through traditional windows, icons, menus and pointers (WIMP) interfaces. It will also describe how these models are realised through the tools that have been created to support both the design and implementation of these types of systems. It will also be argued that an attempt to extend WIMP tools to accommodate EPIC-style interaction is neither an easy nor appropriate way to offer support for the development these novel interactive systems.

EPIC systems are not the only class of interactive application to suffer the limitations of tools designed to support the development of common styles of interaction. Virtual reality (VR) applications, like EPICS, require support for alternate input modalities through non-standard devices. Furthermore, VR systems have interfaces that are not based on the WIMP model of interaction but rather on the tracking of users' movements and the recognition and translation of their gestures into commands. By establishing a conceptual model of interaction that is common among VR applications, it has been possible for suitable tool support to emerge. Therefore, by establishing a similar conceptual model for the interactions that occur within electrophysiologically-focussed interactive systems, it should be possible to decide what kinds of tool support best meet the needs of developers of future EPIC systems.

4.2 Conceptual Models of Interaction

In order to begin to consider the design of any one of a group of interactive systems, it is necessary to identify the high level features that those systems share. Conceptual models of interaction provide abstractions over the implementation details of a class of systems by considering very high level concepts such as *the interface*, *the application* and *the user*. Working from this perspective it is possible to talk about communication between user and application and how this might be facilitated through the interface.

4.2.1 Communication Between Interface and Application

Two common models exist for supporting communications between an interactive system's interface and its application. These are *linguistic dialogue* and *shared state* (Dix 1998)**.** A linguistic dialog is a formal language that converts user-generated events, such as mouse button press actions, into requests for application services. Conversely it converts application-generated events into user interface actions. The linguistic approach to interface-application communication is demonstrated by the Seeheim model (see Figure 4.1), an idealized architecture which was proposed by the Seeheim Workshop (Pfaff 1985). The Seehiem architecture effectively partitions an interactive system into three distinct components:

- **Presentation component**

This component controls the appearance of the interface as well as promoting services available to the user through the interface. It interfaces to the input devices, transforming signals generated by them into an abstract language that it shares with the dialogue control component.

- **Dialogue control component**

This component serves to regulate the communication that takes place between the presentation component and the application component. It effectively defines the structure of the dialogue that takes place between the user and the application. User inputs are passed to the dialogue component by the presentation component as tokens. These user-input events cause the dialogue component to send tokens to the application interface component.

- **Application interface component**

This component represents the view of application semantics provided by the interface. On receipt of information from the dialogue controller, the application interface component may do one of two things. If the information received is syntactically valid, it is passed on to the application. If it is not syntactically valid data, the application interface informs the dialogue controller of this fact.

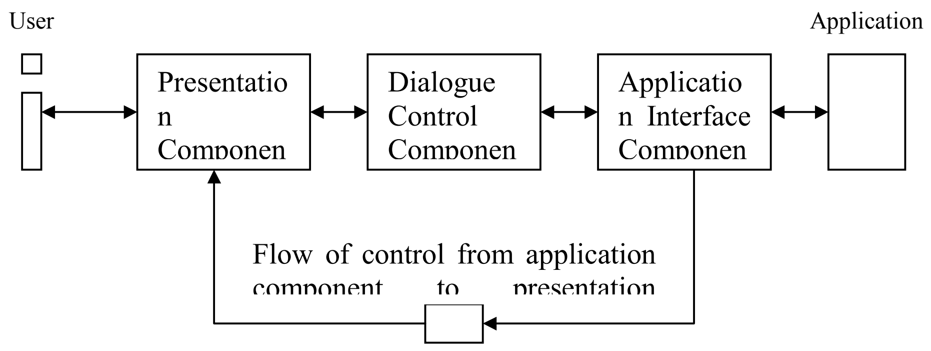

User
Application
Presentatio n Componen
Dialogue Control Componen
Applicatio n Interface Componen
Flow of control from application component to presentation

Figure 4.1 The Seeheim model

It is possible for the application interface to bypass the dialogue control component and communicate with the presentation component directly. This is in order for the application to provide greater semantic feedback about application level structures. This bypass also serves to overcome some of the performance limitations inherent in the model due to the communication overhead produced by token passing between components.

Interactive system implementations based on Seeheim must not only determine how the separate components are to be realized, but also support communication between those components using linguistic dialogue. Examples of systems that reflect the Seeheim model of interaction include the DRUID (Singh 1990) user interface management system (UIMS), which supports communication between interface and application components by the passing of tokens. Also VUIMS (Pittman 1990) which provides inter-object communication via asynchronous message passing. User interface objects in VUIMS can communicate by sending messages, where a message takes the form of a request to the receiving object to call a method, or change some state within in the sending object.

The notion of token passing between components does not scale up to the building of large and complex interactive systems out of small building blocks such as those found in many development toolkits. As is the case with most idealised models of how things should be done, Seeheim addresses the *logical* concerns of interface-application communication, but not the *implementation* concerns. This limitation has since been recognised and is the reason for the bypass, which enables the application to deal with the presentation component directly. When required to do so, usually for reasons of performance, it is possible to switch communication from one mode (dialogue) to the other (direct).

A more suitable paradigm for building systems out of small, preconceived parts is the Model-View-Controller (MVC) (Krasner 1988). This model, illustrated in Figure 4.2, is the model employed by the Smalltalk programming environment. Smalltalk (Goldberg 1983) is built around a library of object-oriented classes, which are inherited by interface object instances and modified to create different interaction techniques. The object classes in Smalltalk incorporate the behaviour of models, views and controllers.

Rather than supporting communication based on linguistic dialog, the MVC maintains the link between application and interface by means of shared state. The *model* of the MVC represents the application semantics, and serves as the state shared between application and interface. The *view* is similar to the presentation component in the Seeheim model, in that it manages the graphical and textual output of the application. But whereas the presentation component also handles user input, in MVC, a separate entity called the *controller* is assigned the task of managing input into the system. The view represents the ‘look’ and the controller the ‘feel’ prescribed within the Smalltalk toolkit. Models are necessarily more general as they are required to represent any potential application semantics. A feature of this generality is that it is possible to reuse a model in various MVC relationships (whereas the view-controller can only be associated with one model).

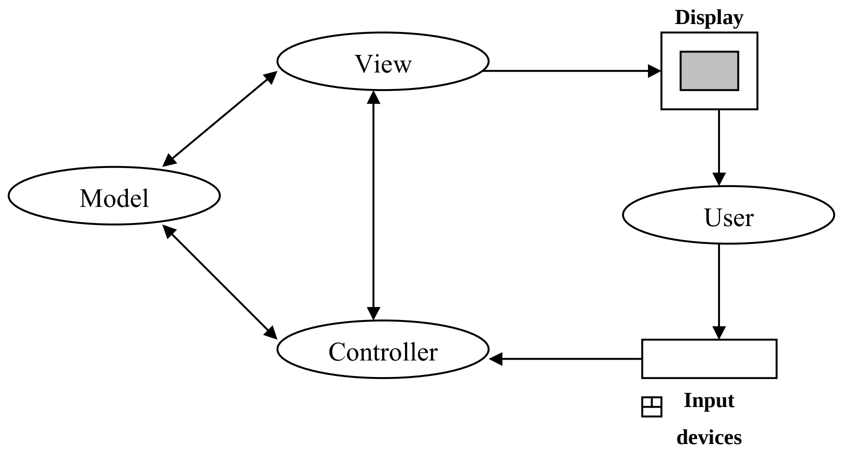


Figure 4.2 The MVC model

The MVC model uses *callbacks* to support communication between a *view* or *controller* and its associated model. The presentation component (*view* or *controller*) has to register itself with the application component (*model*). Communication to and from a *model* to either a *view* or *controller,* or between a *view* and a *controller* occurs in use of normal use of method calls used in object-oriented programming.

The X Window System's (Scheifler 1986) programming model supports user interface and application linking via callbacks. It is possible to add callbacks to any event that can be generated within the user interface. The callbacks themselves reside within the application code. When a user interface event is generated, the callback invokes a procedure within the application corresponding to that user interface event.

A third model for interface-application communication is *presentation-abstraction-control* (PAC) (Coutaz 1987). Shown in Figure 4.3, PAC is a conceptual hybrid of Seeheim and MVC. It has a *presentation component* similar to that in the Seeheim model which groups input and output together. The application semantics reside in the *abstraction component*, which is equivalent to MVC's model. But whereas in MVC the application semantics act as shared state between the input (controller) and output (view) components, in PAC a separate *control component* is assigned the task of explicitly managing the dialog between the presentation component and the abstraction component. Neither the Seehiem model nor MVC provide a means of separately managing the dialogue between interface and application.

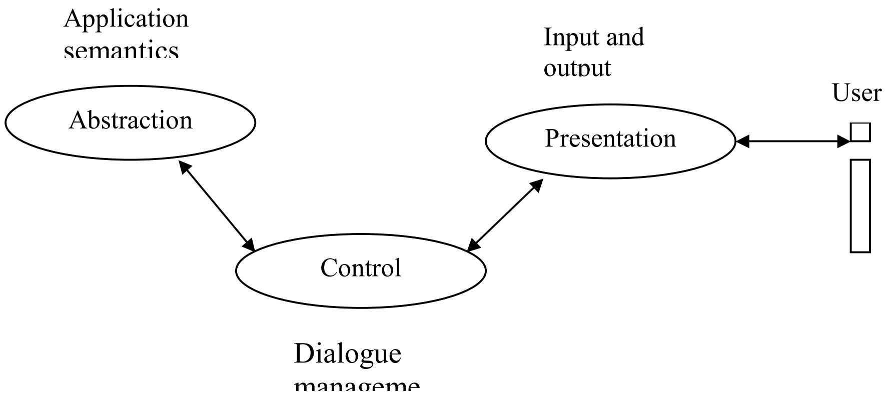


Figure 4.3 The PAC model

4.2.2 Communication between User and System

Within these high level models of communication between interface and application, a notional user is always present, but little has so far been said about how communication between the user and the system is realised (although user generated events are implicit in the models presented above). There are a number of issues to be considered whilst thinking about user-system interaction. The first is what the system's view of user-generated events is - in other words what mechanisms are in place within the interface to deal with such events? Other issues include what kinds of events do the modes of interaction offered by a system enable users to generate, and related to this, how are these events representative of the user and his or her real intentions?

These issues are addressed throughout the rest of this chapter, but to begin with it is useful to consider Took's (Took 1990) Surface Interaction model, which elaborates upon the notion of interface and application divided yet focuses on user-generated events and how these are handled within the system.

According to the Surface Interaction model (Figure 4.4) there are two types of user-system interactions that can occur. Surface interactions are those where a user's interactions with an interface are received by and handled within the system's interface. In other words, the nature of the user request is such that the interface contains the functionality to address the requirements of the interaction without having to call any functions from the underlying application. So, for example, when a user moves a graphical object on the screen, the interface can manage this action itself and therefore does not need to access functionality from the underlying application.

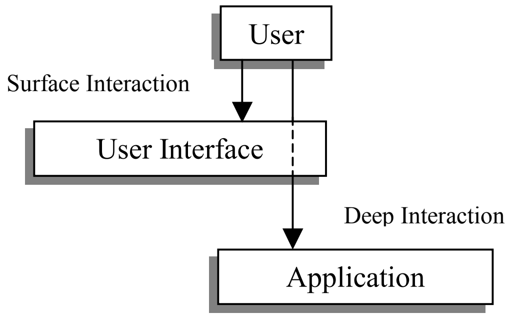


Figure 4.4 The Surface Interaction Paradigm

Deep interactions on the other hand are those where user actions at the interface do require the invocation of functionality from within application in order to fulfil the requirements of the action. An example would be a user requesting access to information stored in a database, which would involve a call being made to the application database to retrieve the required information.

In Took's model, the interface acts as filtering mechanism on user-generated events, capturing interface level events and passing on application level events. This obviously results in a reduction of the number of events flowing down into the application. Furthermore, the notion that user actions can be supported by functionality *within* the user interface software means that the interface cannot be viewed simply as facilitating veneer over the application, rather it is a functionally integral part of the system.

4.2.3 Discussion

The conceptual separation of the user interface from the application is a widely accepted technique for managing the development of interactive systems. This division serves to simplify both the design and implementation of what are both complex and code intensive systems. One advantage of developing the interface separately from the application is that it makes for easier subsequent maintenance and modification of the overall system. Other advantages to be gained by developing separate interface and application include:

- **Portability** – By developing the interface independently of the application, it is possible to use the same interface to present different applications. Also, the application can be used with various systems as the interface encapsulates any device-dependant functionality.

- **Multiple Interfaces** – By developing an interface that is largely independent of an application, it is possible to provide a single application with a choice of multiple interfaces to suit different situations and user needs.

- **Reusability** – Code developed in the manner can be reused in other systems. Reuse can be partial (e.g. some useful widgets) or total (e.g. the whole interface or application). The ability to reuse code reduces development time and consequently cost.

- **Customization** – The interface can be customized, either to suit the requirements of the design or to match the preferences of the end-user, without these changes affecting the underlying application.

4.3 Developing Interactive Systems

Beyond the identification of a suitable high level model of interaction, there are a number of practical considerations that can influence the form a particular development support tool may take. The first of these centres on the problems associated with interactive systems development. As Dix points, it is not possible to completely specify all the features of an interactive system at the beginning of that system’s development lifecycle (Dix 1998, page 205). For this reason it has been found necessary to adopt a development strategy that supports, in its early stages, the gathering of requirements for both the functionality and interactivity of the final system. In order to achieve this, software developers have adopted an incremental development strategy based on prototyping.

4.3.1 Prototyping

System development based on prototyping involves modelling a system (or parts thereof) based on some initial requirements. Evaluation of the model takes place in order to establish further requirements for the overall design of the final system. The importance of considering the end-users within the development cycle of an interactive system is now widely recognised. High level cognitive models such as GOMS (Card 1983) have been formulated in an attempt to describe formally how a user *may* interact with a system. But undoubtedly the most powerful way of finding out how an end-user *will* use a system is to mock it up and let them try it out. The mock up of a system does not have to be functionally complete and only has to give an impression of how the final system will look, feel and ideally how its interface will behave. For this, a simulation of the interface only will usually suffice.

Figure 4.5 illustrates the iterative nature of user interface development which cycles around three principle stages - prototyping, evaluation, and evolution. An initial prototype of the interface is constructed by examining the interfaces to other functionally similar interactive systems. A mock-up is created and presented to an end user, who attempts to carry out some mocked-up tasks that will be supported by the final implemented system. By observing the results of the end-user's efforts, missing functionality and/or user misinterpretations can be identified.

The requirements identified from the evaluation are used to make modifications to the prototype, which is then presented to the user again. This cycle continues until all of the requirements for the interface have stabilized. If up until this point, the prototype has been created using a suitable high level language then the evolved prototype could form the actual interface to the final system. This is known as evolutionary prototyping, and ensures that all time invested in the development of the prototype contributes to the final system implementation time.

The process of user interface prototyping provides a dialogue between the developer and the user about the features in an evolving system. Prototyping of user interfaces often takes place apart from the application, although some functionality is usually embedded within the user interface prototype that mimics application tasks in order to give both the user and designer a feel for the system's response to user actions.

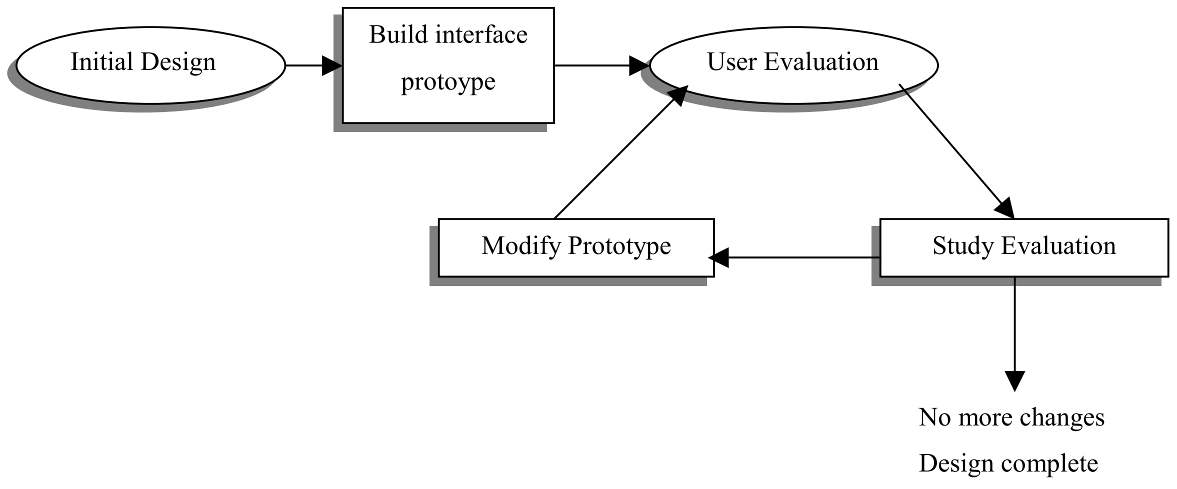


Figure 4.5 User interface prototyping, after Pressman (Pressman 1994)

Evolutionary prototyping is a process that requires at least two pools of expertise. Firstly prototype creation involves the design and possibly the coding skills of an interface developer. Graphical prototyping tools such as Smalltalk and HyperCard (Apple 1998) allow the simulation of interactive behaviour through the writing of scripts associated with graphical images placed on cards which represent sequential events. The activity provided by the scripts enables a user to get a feel for interactive properties of a system. Providing a real sense of the usability of a real-time interactive system invariably requires more complex prototypes which require more involved coding on the part of the developer.

Secondly knowledge of the application domain for the system (i.e. what tasks and work processes the system is to support) must be provided either by the end-users of the system or by other domain experts. This information augments the set of initial requirements gathered from assessment of similar systems, with both sets of requirements feeding into the creation of the initial prototype. Feedback regarding the usability of the evolving prototype should ideally come directly from a potential user or group of users.

4.3.2 Supporting the Activities of Non-programmers

Myers (Myers 1995) indicates that as well as the end-user and the application programmer, there may be a separate interface designer, whose expertise lies in overcoming the gulfs of execution and evaluation identified by Norman (Norman 1986) through considered layout and selection of appropriate metaphors and modes of interaction. Finally, there is the tool creator who produces the tools that support the work of both the designer and the application programmer.

For the creation of an interactive application such as Microsoft Word it is feasible to have individual designers and application programmers (probably teams of both) involved in the development of the interface. Justification for this comes from the fact that such an application can be expected to have large numbers of users and as such to generate large revenue for its creators. For smaller applications however, it is more common for the designer and the application programmer to be the same person. In this case, support must be provided within the development tools for 'good design', such as the semantic structures offered by interface frameworks.

A more difficult group to accommodate are the application domain experts, who are more like users than designers, but whose input to the design of the interface to a particular application is invaluable. For example, Chapters 2 and 3 indicated that the domain experts in physiological signal monitoring and assessment are not programmers, but clinical psychologists and psychophysiologists. Experts from a range of fields including psychology, social science, ergonomics and graphic design, have a great deal to contribute regarding what is and is not acceptable in the design of interactive systems. Few of these people have programming expertise, and in order to support their involvement in the process of prototyping user interfaces, high level tools are required that abstract over the implementation details of the interfaces they wish to create.

A variety of tools exist which support, to a greater or lesser extent, the development activities of non-programmers. The simplest form of prototype is a static storyboard that visually describes the series of steps a particular user-system interaction involves. Storyboards do not require computer support, and can be implemented on paper with a pen. They are obviously non-functional prototypes, providing instead snapshots of the different state transitions that the interface undergoes during a particular exchange with the user. If implemented using a computer-based graphical tool, then an animated storyboard can illustrate the transitions from state to state.

Card-based prototyping tools such as HyperCard and Trillium (Henderson 1986) provide a form of storyboarding, by enabling the creation of series of usually static pages that sequentially describe a system's functionality through its interface. Trillium, for example, is a graphical tool initially conceived for constructing interfaces to photocopiers. The photocopier display panel is represented by a collection of functioning frames, each containing a number of visual items which can be any combination of artwork, dynamic information displays, sensors, initializers of interface-state, inhibitors or change propagators which together describe the interactivity of the photocopier. These are photocopier-specific widgets, which can be viewed as being analogous to the interface widgets popular in WIMP interfaces.

Both of these forms of prototyping tools enable a user interface to be defined by positioning graphical objects on a screen. What makes these tools accessible to non-programmers is the ability to construct interface visualizations by direct manipulation of graphical entities. Unfortunately the prototypes that are produced have limited functionality and serve really only as concept demonstrators. They are a form of prototype commonly identified as being throw-away, in that their usefulness is limited to presenting ideas and serving as a discussion platform between potential system users and designers as to requirements that need to be incorporated in the final implemented system.

Most graphical tools available for supporting the development of interactive systems are themselves interactive systems. Graphical entities corresponding to user interface widgets are presented as programming building blocks that can be manipulated visually. These can be combined on the screen to produce a visual representation of the user interface. The environment generates the actual code to implement the final interface design.

The main difference between visual prototyping tools such as Trillium and visual *programming* tools such as Visual Basic (Microsoft 1998), is that the prototype that is created using the latter type of tool is a runnable simulation of the system interface, rather than a walk-through. This runnable simulation provides a better entity to present to an end-user for evaluation. It gives the user a better sense of what the interface to the final system will look and feel like, and it gives the developer a better sense of the interface's responses to user-generated events, such as mouse clicks over buttons.

Visual programming environments also take a step beyond interface prototyping, toward interface implementation. The interface prototype can be built and evaluated. Rather than discarding the prototype after evaluation it can be evolved into the final system interface. The output from visual programming environments is compiled high-level system code, which can be incorporated into the final implementation. This obviously saves development time and by eliminating the need to discard the prototype after evaluation and re-implementing the final interface from scratch.

4.3.3 Discussion

Sommerville (Sommerville 1992, page 106) points out that the cost of prototyping parts of a system can represent a large fraction of the cost of the total systems development and for this reason it is desirable to avoid spending time producing a prototype that will subsequently be thrown away. The only justification for the use of throw away prototypes is that they sometimes provide the best mechanism for system designers who are also non-programmers to convey their ideas to a system developers and/ or end users. In order to support the design activities of non-programmers, whilst avoiding the cost of throw-away prototyping, visual programming environments are powerful tools which provide an abstraction over the underlying implementation issues through the use of graphical drag and drop interfaces.

Regardless of its form, the user interface prototype is key to the development of an interactive system. For this reason it is important to produce an initial prototype quickly so that the evaluation and refinement cycle can begin. In order to reduce the cost of development, the best kinds of prototyping tools are those that enable the creation of prototypes out of code fragments that can be reused in the final implementation. For developing WIMP-style interactive systems, this approach is supported through the provision of a wide range of software toolkits and user interface management systems.

4.4 Supporting the Development Process

By far the most popular class of interactive computer systems are those based on the windowing model of interaction. The idea behind windowing systems is that they enable users to maintain a number of simultaneous tasks on a computer screen at the same time by presenting these tasks in separate windows. Windows support different user-machine dialogues as disparate entities giving the user freedom to shift attention from one task window to the other. The underlying window system manages the production, placement and destruction of windows on the screen as well as managing the raw events generated by input devices such as the mouse and keyboard. Figure 4.6 is taken from Myers' taxonomy of graphical user interface development tools (Myers 1995). It illustrates the layering of tools that have emerged to support the development of interactive computer systems conforming to the WIMP style of interaction. A taxonomy of window systems can be found in Jones et al (Jones 1991).

A variety of tools have emerged to support the development of applications which run on windowing systems such as X (Scheifler 1986) and Microsoft Windows. These tools define a particular look and feel on the user interface to an application by prescribing what interface entities such as buttons and menus should look like, and how they should behave. Just above the window system, the first layer of tool support takes the form of a user interface toolkit.

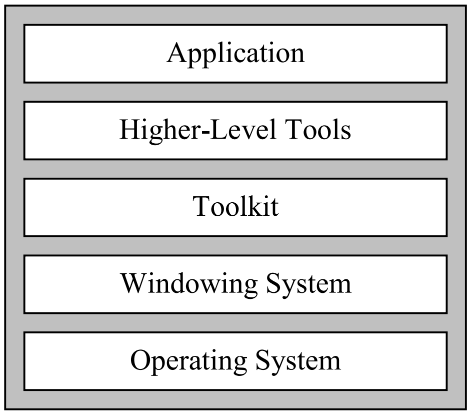


Figure 4.6 Components of user interface software, after Myers

User interface toolkits support the construction of graphical user interfaces by providing *libraries of interaction techniques that can be invoked by application programs* (Myers 1995, page 331). An interaction technique is a mechanism for supporting user interactions and takes the form of a graphical object with associated user-generated event handling capabilities. Interaction techniques, more commonly referred to as 'widgets', can be combined to specify an interface, the behaviour of that interface and in some cases the connections the interface maintains with the underlying application (Bentley 1994). Interaction techniques provide the system with a view of the user by supporting communication between the user and the application through events generated within the interface.

4.4.1 Toolkits

There are two forms of user interface toolkit, libraries of procedures such as TCL/Tk (Ousterhout 1994) and object-oriented interface classes such as those in Smalltalk. The first advance in user interface development support took the form of pieces of reusable code that could be combined to produce common functionality within the interface of an application. TCL/Tk is a purpose built scripting language, which provides an abstraction over libraries of useful interface procedures implemented in the C language. Although users of TCL/Tk do not require an intimate knowledge of C in order to create a user interface, they do need to understand the TCL scripting language, and scripting is just an another form of coding, albeit using a higher level syntax.

Object-oriented interface classes such as those in Smalltalk and Java's AWT (Flanagan 1997) can be instantiated in order to create user interface widgets. As with procedural libraries, creating user interfaces with object classes involves low-level coding, despite the availability of graphical drag and drop interface building tools. There are, however, several benefits of object-oriented toolkits over procedural toolkits. To begin with objects are a natural way to think about the components that make up the interface, having both representation (state) and a method of interaction (procedures to operate on that state). Object-oriented toolkits also provide more structure than procedural libraries to the process of user interface creation, as the design of widgets within object oriented toolkits is strictly hierarchical. This hierarchical structure is reflected in the architecture of the interface to an interactive application (see Figure 4.7).

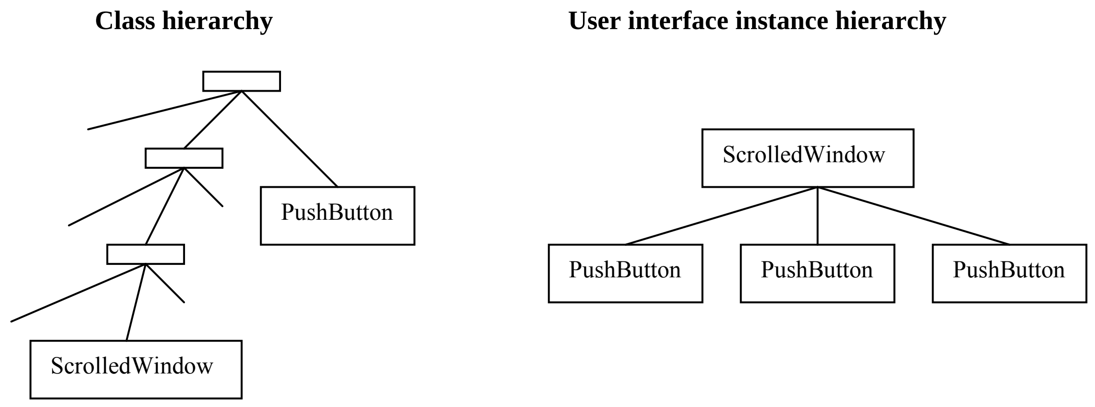


Figure 4.7 A class hierarchy and a user interface hierarchy, after Smith (Smith, 1995)

Due to the expense of developing toolkits the initial set of interaction techniques provided by them is necessarily small. Of course it is impossible for their developers to envisage all potential modes of interaction that will be required in future applications. Addition of new interaction techniques into procedural libraries involves coding them from scratch. Object-oriented toolkits are a little better in that new interface entities can be created by sub-classing existing classes.

Object libraries also tend to have larger initial widget sets than those provided by procedural toolkits.

Toolkits provide code for the instantiation of common interface widgets such as buttons, menus and text dialogues. The aim of toolkits is to support rapid prototyping of user interfaces and shorten the implementation time by providing useful code that can be reused in the implementation of many interfaces. At the beginning of development, using a toolkit is not particularly rapid, even for experienced programmers as it takes a while to become familiar with the available interaction techniques and how they can be used.

From a design point of view, the main disadvantage of interface toolkits is that their procedural interfaces make them difficult to use by anyone other than application programmers. Building an interface with one of these tools involves hand coding, as there is no support for laying interfaces out graphically. To find out what a particular interface configuration looks like involves an extended *code, compile* and *review* cycle.

Frameworks such as MacApp (Schmucker 1986) provide a level of support above that of toolkits and effectively provide rules of 'good design' in the form of semantic structures. A complete application or interface framework is presented in the form of a generic structure of classes. By sub-classing the framework's classes, a system developer can provide and connect to suitable, application- or interface-specific functionality. As with toolkits, frameworks are designed to support the developmental work of experienced programmers, who are required to understand both the framework and a suitable programming language. Used in conjunction with toolkits, frameworks aim to support both interface design and usability through the use of consistent layout, use of metaphor and action.

Fortunately, many toolkits are supported not only by frameworks but also by graphical interface builders, which offer support to both the design and development of user interfaces. Interface builders, such as XF builder (Delmas 1993) which supports TCL/Tk, overcome the limitations of the toolkit elements' procedural interfaces by providing a graphical environment that supports both direct manipulation of components and access to their functionality via graphical widgets.

4.4.2 User Interface Management Systems

User interface management systems (UIMSs) correspond to the 'windowing system' layer shown in Figure 4.6 above. Dix (Dix 1998, page 395) defines a UIMS as being "*a conceptual architecture for the structure of an interactive system which concentrates on the separation between application semantics and presentation.*" UIMS bring visual programming techniques together with toolkits and frameworks as well as code generation and compilation facilities. Together these tools provide integrated environments for the prototyping, implementation and in some cases evaluation of interactive interfaces. Within a UIMS, provision is made for developing not only the interface and related parts of the application but also the communication that is to take place between the interface and application.

UIMSs often profess to support the activities of non-programmers, although not all UIMS offer visual programming techniques but rather require users to learn a special purpose programming-like language in order to specify the contents of an interface. Druid and Garnet (Myers 1990) are UIMS that *do* allow interface designers to draw aspects of the user interface directly onto the screen using a mouse. Lapidary, the interface builder in Garnet, enables relationships between interface objects to be established through the drawing of lines between interface objects. A designer can then either visually demonstrate the response an interface object should provide to user-generated mouse actions, or specify these responses using a dialogue box.

Some UIMSs, such as XXL (Lecolinet 1996) support the integration of both visual and textual programming styles within a single framework. This means that all interface components have both a visual and a textual representation. Both types of representations can be mixed in the construction space, which is a graphical direct manipulation environment. XXL is built to provide support to the Motif toolkit. This means that the only widgets available within the environment are those provided within Motif.

The problem of restricted numbers of interaction styles in interface toolkits obviously propagates up to the graphical editors that support them. Adding new interaction techniques to most UIMS involves low-level coding by an experienced programmer. Although user interface management tools such as Peridot (Myers 1989) try to overcome this requirement for hand coding by allowing users to create new interaction techniques by editing the functionality of existing techniques using direct manipulation of a graphical representation of the original. The user can demonstrate the new functionality of the interaction technique by using the mouse to show the system how an object should respond to user input. Interaction rules provided by the system are used to infer behaviour from these user actions. Peridot attempts to infer relationships between objects within the interface from set of rules.

UIMSs are designed to support the development cycle of user interfaces. Tools such as Gamut (McDaniel 1997) are beginning to emerge that support the development of entire interactive applications. Gamut is a tool for building complete interactive applications such as games and educational software. Development takes place via programming-by-demonstrations (PBD) and unlike many UIMS there is no need to look at or modify code to create any behaviour. An interface developer draws interface configurations directly on screen and then indicates the desired behaviour of objects in the interface by demonstrating the desired program response to input events using the mouse. Gamut is able to infer single conditional relationships between objects (respond or not depending on some condition). It can also infer chains of relationships between objects.

4.4.3 Supporting Interaction

It is possible to identify and support common modes of interaction based on the types of user-system dialogues that occur within a particular class of system. An important concept in interactive systems development is the notion of device independence, which involves classifying input devices within generic classes of what are known as *virtual devices* (Buxton 1990). The form that a dialogue for a given application takes is described in terms of the virtual input devices that that application supports. Once defined, an application's dialogue can be supported by any of the associated class of virtual devices. So, for example, interactive systems with WIMP interfaces support both mouse- and keyboard-generated events. Communication between the user and the machine takes the form of a turn-taking dialogue, usually within a graphical, direct manipulation environment, driven by the generation and processing of discrete, user-generated input events (key presses and mouse button press, drag and release actions). A conventional QWERTY keyboard, a Dvořák keyboard or a chord keyboard could theoretically be used to generate the keyboard events. Similarly a joystick, trackball or trackpad could replace the mouse.

4.4.4 Discussion

The relationship between virtual device and interaction dialog is the reason why tools for developing WIMP-style applications cannot easily be applied to the development of systems requiring non-WIMP interaction. WIMP tools support dialogues based on discrete user-generated events whereas gestural input devices such as those employed within virtual reality applications require user-system exchanges based on continuous user-generated event streams. The controller that exists within the application to multiplex input events to their meaning works in conjunction with the windowing system on which the application is based. The lack of support for alternative methods of user input in user interface development tools reflects what Buxton (Buxton 1990) identifies as the low priority input devices are given in the overall development considerations of interactive systems.

What can be gleaned from this overview of WIMP systems development is that an understanding of high level conceptual models of interaction eases the design of tools that support the development of these systems. The two key features provided by conceptual models of interaction are:

1. A set of clearly identifiable high level components
2. A well defined architecture

By examining the process model adopted for the development of interactive systems an emergent requirement is to provide tools that support the broadest possible involvement in system development. With these issues in mind, it will be helpful to look at tools that have been created to support the development of non-WIMP systems. To this end, the following section reviews tools for developing virtual reality applications that incorporate both continuous and discrete input and output devices.

4.5 Virtual Reality Systems

The aims and objectives section of Chapter 1 implied that in order to design suitable EPICS development support tools it is necessary to first identify an appropriate model of interaction, based on sensing, which describes the information exchange between user and system. We have stated in the discussion immediately prior to this section that the WIMP model of interaction is not an appropriate model on which to base electrophysiological human-machine interaction. In order to work toward identification of a more appropriate interaction model, we shall now look at the more novel interaction techniques developed for virtual reality applications, and the tools that have been developed to support this model. As we shall see, many methods of interacting with virtual reality applications are based on sensing the user.

The interaction requirements demanded by virtual reality applications throw up a crop of challenges for interface designers. For a start, interaction between the user and a VR system takes place within a 3-dimensional computer-generated space. Input and output devices required for interaction within a 3D space are all non-conventional, from head-mounted displays (HMD) with stereoscopic output and magnetically tracked gloves/bodysuits to servo-mechanical input devices of all shapes and sizes, from wands to bicycles. A VR system receives a stream of data from each continuous input devices, which it translates into movement and actions on objects within the virtual environment. Semantics within a VR system are application-dependant with existing applications for VR technology ranging from medical and military simulations through to supporting remote users involved in collaborative work.

In order to build systems adhering to the VR model of interaction, tool designers have to rethink the services provided by the tools they create. At the end of their overview of traditional WIMP development tools Baecker et al (Baecker 1995) state that *"A completely new style of toolkit is required to address these types of interface, for they must maintain the integrity of the underlying graphics database and views, as well as managing the difficult problem of three-dimensional multimodal input."* (page 310).

In stark contrast to the wide range of support tools for building applications based on the WIMP interaction model, developers wanting to create VR applications currently have access to relatively few integrated prototyping and implementation support tools. The problems associated with the development of VR applications at the present time are summed up by NASA's VR lab (Engelberg 1994):

"*For a standard VR application, our development process is composed of three basic steps:*

1. *Create the 3D models using Solid Surface Modeller (SSM) or translate models from Inventor format to SSM format.*

2. *Build a hierarchical structuring of these models using Tree Display Manager (TDM).*
3. *Program all interactions in C*"

In a bid to address the problem of developing interactive VR systems, toolkits are required to support the development process. As with interactive systems based on the WIMP paradigm, the first step in the creation of support toolkits for VR is to identify a suitably high-level model of interaction toward which these toolkits can adhere.

4.5.1 Conceptual Model of VR Interaction

In order to facilitate the creation of tools to support the development of VR systems, identification of a high level conceptual model is required to which these tools adhere. The high-level components shared by all virtual reality systems are illustrated in Figure 4.8. The *environment manager* provides a run time environment for the virtual environment, whose physical properties and behaviours are described within the *world model*. These two components together provide an interface, in the form of an interactive 3D world, between a user and an application.

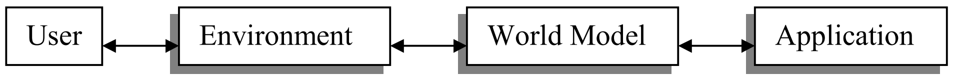


Figure 4.8 Conceptual model of VR interaction

4.5.2 VR Toolkits

As with WIMP toolkits the most basic support available for development of virtual reality systems are collections of subroutines to be used by application programmers. Perhaps the most widely used of these at the present time is the Virtual Reality Mark-up Language (VRML) (Ames 1996) which consists of libraries of objects for building world models through the definition of geometry and object behaviour. Creating world models using VRML involves either writing them directly by hand or using a graphical modeller such as Cosmo Worlds (Cosmo 1999a). VRML is designed for deployment through a web enabled environment browser such as blaxxun (blaxxun

1999) or Cosmo Player (Cosmo 1999b) and as web-based VR is predominantly desktop VR, most interactions between the user and the VRML application occur through mouse and cursor key actions.

An alternative toolkit, Java3D (Sowizral 1997) is a collection of Java class libraries for building 3D applications. These classes provide the functionality for building 3D geometry with associated behaviours, as well as constructing the structures that are required to render the geometry. This equates to creation capabilities for both the environment and the world model. A variety of continuous input devices such as six-degrees-of-freedom (6DOF) trackers and joysticks are supported by Java3D (with access to keyboard and mouse events being provided by the standard Java API classes). Java 3D also defines an abstract class for dealing specifically with streaming input, which it collects as a time-stamped sequence of input values plus button/switch states. The advantage of Java3D over VRML from a developer's point of view is that Java3D world models and environments can be easily integrated with applications built using standard Java classes.

Both Java3D and VRML were designed initially to support interaction between a single user and a single application. More sophisticated toolkits such as MASSIVE-2 (Greenhalgh 1999), DIVE (Carlsson 1993), SPLINE (Waters 1996) and NPSNET (Macedonia 1995), enable the prototyping and building of distributed multi-user interactive virtual environments. In DIVE for example, a number of application processes can exist within a single DIVE world model, each maintaining its own user interface of graphical objects. Once established, DIVE applications listen to events generated by users within the world. When an event specific to an application occurs, that application reacts according to some predefined control logic. Communication between DIVE entities takes place via shared data objects. It can be seen in Figure 4.9 that the core model of interaction is maintained, despite the numbers of users and applications.

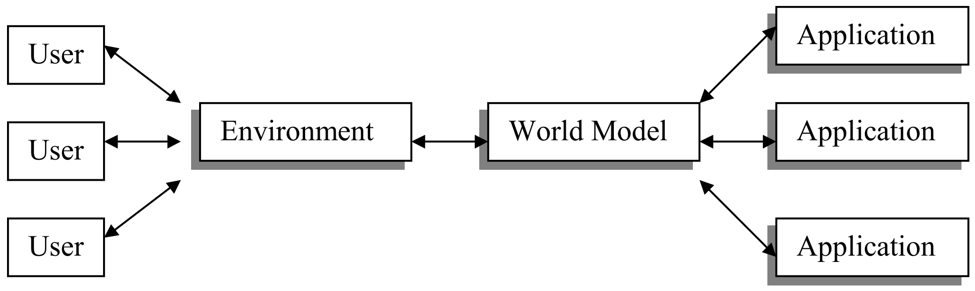


Figure 4.9 Multi-user model of VR interaction

Development of both DIVE and MASSIVE-2 applications involves programming in appropriate scripting languages (TCL in DIVE and a proprietary language for MASSIVE-2). More comprehensive tools are available which, like visual programming for 2D interfaces, enable the construction of 3D interfaces using environment builders with graphical drag and drop capabilities. Superscape is one such tool (Superscape 1999) which provides graphical world building capabilities over a proprietary file format called SVR. A toolkit with similar visual programming capabilities is MR Toolkit (University of Alberta 1999) which consists of set of subroutine libraries, support programs and a language for describing object geometry and behaviour. 3D environments are constructed out of an Object Modelling Language (OML) using a 3D modeller called JDCAD+. Support is provided in the form of appropriate device drivers for common VR input devices including a variety of 3D trackers, gloves and HMDs

The architecture of MR Toolkit, which is shown in Figure 4.10, consists of three layers of software. The bottom layer offers client/server support for input devices. The server continually samples the device and performs filtering on the received data. The client consists of a set of library routines that interface with the server and request data from it. The second software layer

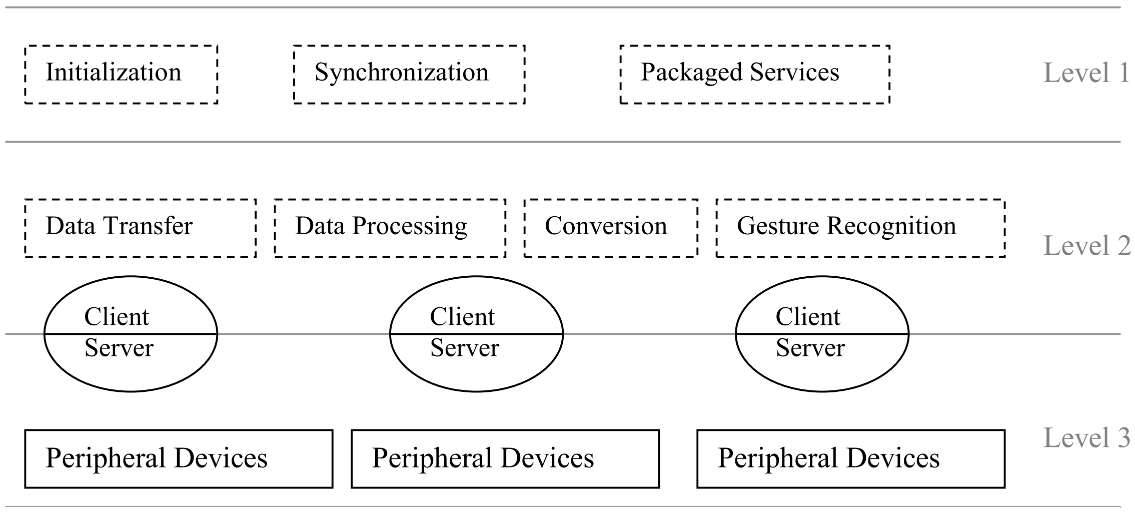


Figure 4.10 Architecture of MR Toolkit

processes data from devices and converts it into a format useful to an interface programmer (for example, the gesture recognition software associated with a dataglove could be embedded within this layer). The third layer provides services corresponding to initialization of all current input and output devices, as well as synchronization of data structures and display operations between the two workstations required to provide binocular vision. MR Toolkit provides libraries of useful 3D interaction techniques as well as frameworks for producing new interaction techniques. The client/server model also makes it easy to introduce new devices by defining new device-specific client/server pairs.

VPL's Reality Built for Two (RB2) is another system for rapidly prototyping virtual environments. It allows an interactive virtual environment to be created quickly without writing any code. A 3D modelling program called Swivel 3-D Professional is used to create 3D objects to inhabit the virtual environment. A central control system called Body Electric links objects within the virtual environment allowing behavioural and interaction rules to be defined. A single data array is fed integer values from each input device. This array is passed to a data flow network where scaling, offsetting and filtering can be performed on the integer values before they are passed to the graphical renderer where they become positional/ rotational values for objects and/or viewpoints in the world. Body Electric controls the flow of data from range of input devices to range of output devices defined within a VR application.

With RB2, a user constructs data flow networks using direct manipulation techniques on graphical representations of raw input, data manipulation modules, points and connecting wires. Raw input can come from any of the defined virtual devices and serve as the main source of data flowing into a data flow network. Data manipulation modules transform data using mathematical, logical and other operators. Points are the destinations of data leaving a network and have many attributes including position and orientation as well as associated constraints on the values of these attributes.

4.5.3 Discussion

The advantage MR Toolkit and RB2 have over systems such as DIVE and MASSIVE-2 is the ability they provide to create environments, objects, behaviours and interactions graphically using integrated tools. This is in contrast with toolkits consisting of code libraries that are only of use to experienced programmers. For example, the behaviour of objects within a DIVE environment requires programming knowledge of C and TCL/ Tk as well as DIVE itself. Building a world model including object behaviours in MASSIVE-2 requires knowledge of a proprietary scripting language.

Graphical 3D interface builder tools such as VRMog (Colebourne 1998) and AC3D (Colebourne 1999) can export graphical entities in DIVE, MASSIVE-2, VRML and Java3D file formats, but tend to do so inefficiently due to their generic nature. Generic graphical tools builders provide world-building capabilities for non-programmers, although addition of behaviours and constraints within the resulting world involves coding, which implies understanding of both the appropriate coding language and the toolkit-specific extensions to that language.

Despite these issues, what is clear from this review of VR toolkits is that their architectures adhere to a clear, conceptual model of interaction, in the same manner that WIMP development tools reflect aspects of Seeheim, MVC or PAC. In the case of VR systems the conceptual model is represented through an alternative set of high level components – *the environment, the world model and the application*. These high level components provide specific foci for different tools which support the creation of one or more components of the model.

4.6 Modes of Interaction

One consequence of the emergence of VR is that, as a discipline, it has challenged the way people think about human-machine interaction. Previously, the relationship between an interactive computer system and its user had been defined solely by a mechanical coupling paradigm, where input signals to the computer were generated through a direct mechanical linkage between the user and some discrete electromechanical input device (Figure 4.11). Popular input devices such as mice, joysticks and keyboards are all examples of transducers that turn discrete mechanical actions performed by the user into discrete electrical signals which can be interpreted by an interactive system.

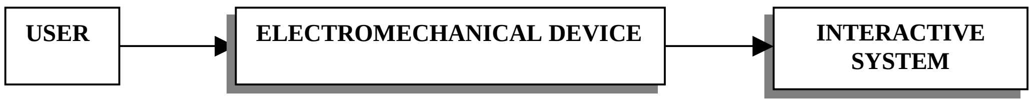


Figure 4.11 Conventional mechanical coupling model

Notification of discrete input events occur through the generation of interrupts within the system. These interrupts cause the system to retrieve the input events and place them in an event queue.

An event loop fetches events in sequence from the event queue. Pre-processing at this stage involves combining raw input actions into compound events in a way that is meaningful to the system. Olsen (Olsen 1997) specifies two tasks that an interactive system's interface controller (after the MVC model) should carry out on user-generated input events. The first task is *spatial dispatching* which involves working out where on the screen an event took place. In a windows-based system this positional information is used by the system to discern which window (i.e. which application) the event was intended for, and if the position corresponds to a particular graphical widget, which service the user was trying to invoke. The second task the controller performs is *sequential dispatching* of input events, which involves assigning some meaning to a sequence of input events. So, for example a mouse-down event over a widget, followed by a drag and a mouse-up event can be interpreted as the user intending to move the widget on the screen.

At the beginning of this chapter it was stated that the type of user-events generated through the input devices supported by an interactive system provides that system with its view of the user. Buxton's look beyond manual input devices (Buxton 1990) captures exactly the essence of how our choice of interaction through discrete mechanical transducers reflects our appearance to the machines we interact with.

An alternative user-machine control model which supports the use of more expressive, non-discrete interaction devices is McMillan's (McMillan 1995) Structural Coupling Paradigm. In this model the physical link between human and machine is severed and replaced with sensing technology. This sensing technology may be motion tracking equipment such as that used in VR applications, or it could be a microphone, video camera or physiological sensing technology of kind described in Chapter 2. An adaptation of McMillan's model is shown in Figure 4.12.

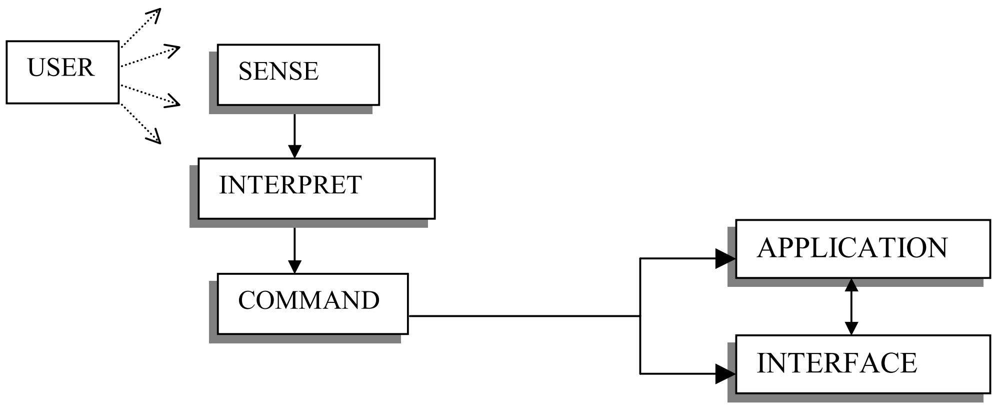


Figure 4.12 The Structural Coupling Paradigm

Sensing technologies gather events related to user actions more or less continually and new forms of human-machine interaction based on sensing are more expressive than those based around discrete mechanical transducers. They enable the user to behave in more naturalistic ways, by gesturing to the system rather than pointing and pressing buttons. Gestures that can be interpreted by modern systems include handwriting, sign language, head motion, body language, facial expressions, voice and vocal intonation. As demonstrated in Chapter 3 even emotions are being investigated as a source of interactive information.

The requirement to sense and make sense of gestural input is usually embedded within complex multimodal, multimedia systems, such as VR systems. An example of the complexity of a non-VR, multimodal, multimedia systems is Dannenberg's (Dannenberg 1992) Piano Tutor. This system, designed to support the learning of piano by students, can hear notes of piano music and compare them to notes on a page. The output media of the Piano Tutor is video, pre-recorded notes, and graphical displays containing diagrams, as well as text, voice comments and music. Input is usually music played by student although access to features within the system requires menu selections made using a mouse.

Sensing-based interactive systems like the Piano Tutor or VR systems allow richer interaction to take place between the system and the user than can ever be achieved using discrete mechanical input devices alone. Sensing systems that deal in gestural user input shift the onus onto the system to modify *its* behaviour to match the user rather than the user having to learn the dialog required to interact on the system's terms. This need for adaptive system behaviour was also found in Chapter 3 to be a requirement of EPICS that undertake processing on naturally-occurring brain signals, as well as emotive and user-state based adaptive systems such as affective computing applications.

The algorithmic structures required to support gesture recognition and other adaptive behaviour are most often embedded within the application. However, if the system is being designed from a high level, rather than being constructed in an ad hoc manner, it worth reconsidering Took's (Took 1990) interaction strategy. This model of interaction, which applies equally well to non-WIMP interactive systems, implies that the interface should contain the functionality required to support direct interaction between it and the user. This in turn implies that the signal pre-processing functionality which influences the behaviour of the system with respect to the user-input stream should be embedded within the interface.

However, embedding signal pre-processing within the interface means that tools designed to support the development of sensing-based systems require the creation of increasingly complex interaction techniques. To add to this complexity, Blattner (Blattner 1996) points out that with many emerging interactive applications, it is difficult to see exactly where the interface is. This is certainly the case with immersive VR applications, where the interface is actually on the body (data gloves, motion tracking, gesture recognition, haptic feedback devices, and HMDs). And as described in Chapter 3, the interface to an EPICS application might be a remote controlled car or a prosthetic limb.

4.6.1 Integrating New Input Modalities into Interactive Applications

By adopting McMillan's (McMillan 1995) paradigm as the high level conceptual model of interaction for systems which incorporate sensing, it is possible to begin to think about how best to provide support for the development of this type of interactive system. The main focus for consideration has to be the pre-processing of input signals that takes place in the *sense-interpret-command* portion of the model shown in Figure 4.12.

The first level of signal pre-processing involves detecting raw data from a sensor and converting that data into a format useable by an application. This may simply involve reading digital signals for the device port and converting them into floating point values for example. Some applications will be more demanding than others in terms of what is a useable format for the data, therefore a second level of pre-processing may be required. For example, applications that depend upon gesture recognition require a controller[25] to deduce the user's intentions from the continuous stream of input information. This deduction requires support in the form of learning algorithms or neural networks, which subsequent to a period of training perform real-time pattern matching on the incoming data stream. Once the user's intention has been interpreted, a command can be issued to that part of the system best geared to deal with the command, be it at the interface or the underlying application.

[25] MVC model

To support the development of sensing-based interactive systems, a method of providing signal sensing and first level signal pre-processing is required. This must be provided in a manner that allows subsequent layers of processing to be added as required by a particular application. Furthermore, the signal processing must exist apart from the interface, for as shown in Figure 4.12 the results may be commands to either interface level entities or application level entities. Therefore, systems based on sensing require signal pre-processing to be carried out apart from both the application and the interface.

4.6.2 Discussion

By reviewing the types of tools that have emerged to support the development of interactive systems, it has been possible to identify a number of features common amongst them. The first is that tools for developing a certain class of systems adhere to high level conceptual models of interaction. So tools for developing WIMP-style interactive systems support the Seehiem, MVC or PAC model of interaction. Similarly, VR toolkits are designed to support the development of a combination of the environment, world model and application.

The input event streams generated by sensing devices are non-discrete and time varying. Continuous input devices are most often supplemental to existing discrete input devices such as mice and keyboards and the interleaved event stream (and multiple dialogue controllers) soon becomes extremely complex. It is not appropriate to try to force an existing interface-building tool to support the signal processing and presentation requirements of continuous input devices. In the first instance, continuous data streams should be dealt with separately from discrete user events, as they are subject to different degrees of pre-processing before they are suitable for presentation to either interface or application level entities. In the second instance the presentation mechanism for a continuous input device may be a continuous output device not necessarily of a nature found in traditional interfaces.

McMillan’s paradigm therefore provides a conceptual model which can be used to inform the design of tools to support continuous interaction events generated through sensing devices such as those found in EPIC systems. The model, like those describing WIMP and VR interaction includes both clearly identifiable high level components as well as a well-defined architecture.

The three components labelled *sense, intepret and command* serve to define the sensing model of interaction from previous models[26].

Within this work we have chosen as our application domain systems that incorporate physiological information. These so-called EPICS provide specialized sensing input devices that produce continuous data streams that require pre-processing. Rather than try to extend an existing toolkit to include pre-processing functionality as well as support a suitable sensing device, we have chosen to implement both the signal detection and pre-processing functionality within a suite of reusable software components adhering to the JavaBean standard. The following chapter provides a description of the design, implementation and application of these Java components

4.7 Summary

This chapter has provided a review of tools to support the development of interactive systems. It was argued that the existence of high level conceptual models such as Seehiem and MVC have served to influence the design of tools which can be used to support the development of any system whose architecture reflects these conceptual models. Tools aim to support the development of one or more of the identified components in a high level model, such as the interface or the application. High level models can be used to reason about the kinds of dialogue that may exist between a user and an application. In the case of WIMP systems, this dialogue is driven by discrete user interactions with recognisable interaction techniques provided by the interface facilitated through the use of discrete input devices.

An alternative high level model of interaction was also presented which reflects the architecture of VR systems. This model partitions VR systems into 3 separate components, the environment, the world model and the application. Toolkits designed to support the development of VR systems were seen to provide tools for developing one or more of these components as well as the communication that would take place between them. Where WIMP interaction was driven by discrete user-generated event, VR interaction was seen to support a multi-modal approach to interaction, relying in part on continuous user-input devices such as motion trackers.

[26] Of course, much of VR interaction is based on sensing although this is not made explicit in the model of VR interaction shown in Figure 4.8.

A conceptual model of interaction was identified for systems that incorporate continuous input devices used for sensing the user's actions. A key component of this model is involved with detection and pre-processing of signals in the input stream. In order to support the development of sensing-based applications, and in particular those that incorporate electrophysiological information, a new approach was proposed based on the incremental construction of high-level software components. These components support sensing, first level pre-processing and then the addition of subsequent levels of signal processing based on the requirements of the application. The details of these components are discussed in the following chapter.

Chapter 5

Design and Implementation of the EPICS Toolkit

5.1 Introduction

Chapter 3 introduced a range of interactive applications, identified collectively as EPICS, that shared the requirement to detect, process and present physiological signals. Figure 5.1 illustrates a typical EPIC system, consisting of an electronic sensing device coupled with a personal computer. The architecture of an EPIC application is shown with the user interface and the underlying application code divided in a manner that reflects the conceptual architectures examined in the immediately previous chapter. It can be seen from this illustration that physiological signal data generated by the sensing hardware is passed directly to the application, which carries out any required pre-processing on those signals and manages the response to/ presentation of incoming physiological data within the interface.

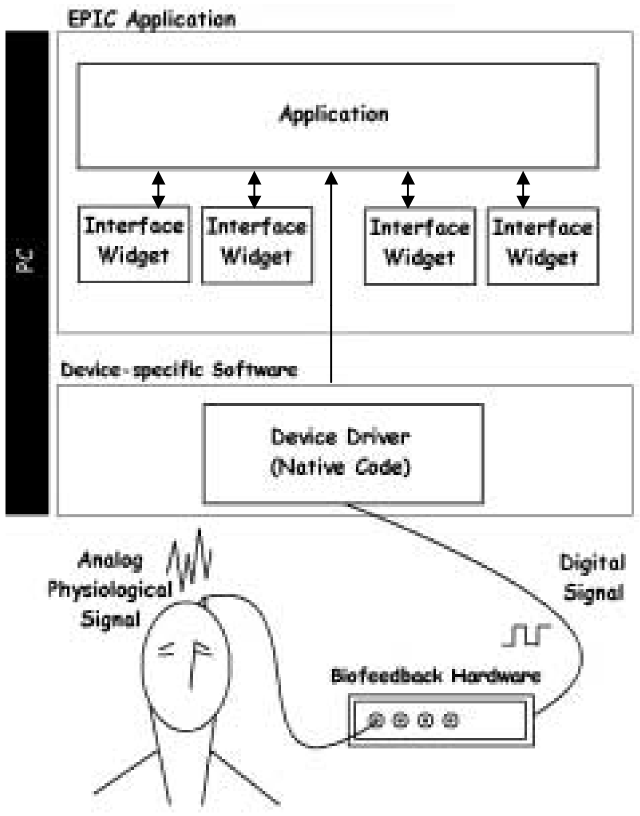


Figure 5.1 Overview of an electrophysiologically interactive system

The overall aim for the work presented in this thesis and laid out in the introductory chapter has been to design and implement a set of interaction techniques to support the future development of EPIC systems. This is to be done in such a way as to provide an abstraction over the low-level sensing hardware. It was pointed out in previous chapters that WIMP interface development tools abstract over discrete user-generated events, such as *point* and *click,* in a way that enables those events to be carried out by any pointing device with a button. The aim for the EPICS toolkit is to achieve a similar level of abstraction so that any electronic sensing device can provide the pre-processing, manipulation and presentation of user-generated electrophysiological events.

Achieving the same interactive functionality over a range of commercial physiological sensing hardware is a more challenging proposition than accommodating a range of traditional computer input devices, however. As previously mentioned, physiological sensing devices produce continuous streams of data that interactive applications have to process in a manner appropriate to their operational goals. In Chapter 3 some sense was given of the diversity of sensing devices in terms of the range of physiological signals they could provide to an application and the number of channels of physiological information they could make available. In order to abstract over devices with different sensing capabilities, a great deal of flexibility needs to be built into any development tools. These issues are addressed in this chapter.

In Chapter 4 the identification of high level models of interaction for different classes of system was shown to have facilitated the design of tools for the subsequent development of these systems. At the end of Chapter 4 an interaction paradigm was introduced which reflects the interaction model of sensing-based systems, including EPICS. In the model traditional electromechanical input devices, which require specific user actions, were replaced with devices that sensed the naturally expressive behaviour of a user. The main feature of the sensing model of interaction was the requirement to undertake advanced processing on the input stream provided by the sensing hardware *before* it was passed to the interface or application. This is in contrast with the EPIC system illustrated in Figure 5.1 above, where the raw signal stream is passed directly into the application for pre-processing.

As a starting point, we can identify 3 functional layers that reflect the McMillan's sensing model of interaction. The layer nearest to the sensing hardware (labelled *sense* in Figure 4.12) is a pre-processing layer which we are going to call this layer the *device layer*. This layer provides the functionality for retrieving data from the sensing hardware as well as providing capabilities for pre-processing the raw signals in order to convert them to a useable form. Pre-processing in this layer corresponds to the series of activities described in Section 2.2.2.2, namely amplification, analogue-to-digital conversion and feature extraction. Device layer signal pre-processing is a necessary requirement for all sensing systems.

It was indicated in Section 4.6.1 that applications that use physiological signals as interaction parameters require further processing to be undertaken on signal streams in order to decide what action (if any) to take based on the current state of (a/some set of) physiological signal(s). For example, a number of the brain-computer control-specific EPIC systems described in Chapter 3 relied upon EEG components being processed by neural networks or other adaptive algorithms in a bid to recognise particular thoughts or actions which could in turn be used as control commands within the system.

This level of signal processing is *not* a requirement of all sensing-based applications, however. Biofeedback-based systems are an example of EPIC systems where the first level of signal processing is adequate to provide data in a suitable format to the interface of that particular class of applications. For this reason, the processing part of our sensing model is best divided into two layers of functionality. The first layer we have already identified as the device layer is essential in all EPICS. A second layer, which we will call the *command layer*, is optional. A revised version of the sensing-based model of interaction reflecting the different signal processing requirements of different EPIC applications is shown in Figure 5.2.

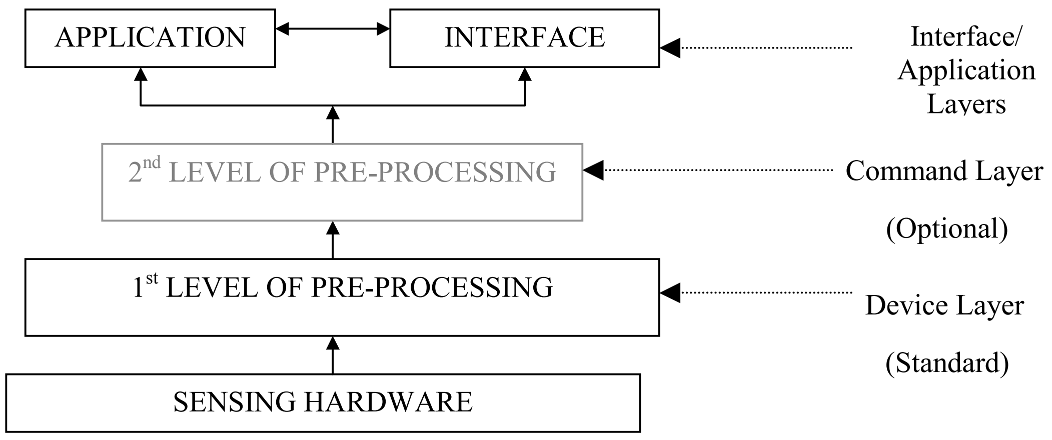

APPLICATION
INTERFACE
Interface/ Application Layers
2nd LEVEL OF PRE-PROCESSING
Command Layer (Optional)
1st LEVEL OF PRE-PROCESSING
Device Layer (Standard)
SENSING HARDWARE

Figure 5.2 A revised sensing model incorporating 2 layers of signal processing

The device layer handles events generated directly by the sensing hardware and as such is device-dependent. The command layer provides both advanced feature extraction and command generation capabilities and is application-dependant. Above the signal processing layers of the interaction model sits the traditional interface and application layers.

The structural coupling model indicated that output from the pre-processing stage could be directed to either the interface or directly to the application. The model suits our classification of EPIC systems from Chapter 3 based on which parts of a system provide the main destination for processed electrophysiological information. Biofeedback and other training-based applications rely on their interfaces to present information about human physiological functioning in real-time to a subject in order to facilitate signal conditioning by that subject. So for training EPICS, the interface would be the direct recipient of the pre-processed signal stream. Systems designed for the continuous monitoring of a user's physiology, on the other hand, require features from the data stream indicative of problems/ state changes to be passed directly to the application, which can then take suitable action.

The rest of this chapter will provide a description of the design and implementation of a set of software components suitable for composing simple EPIC applications. This component suite constitutes the first purpose-built toolkit for supporting the development of EPIC systems. The argument for the choice of a component-based approach to tool support for this new class of systems comes later in this chapter. In the meantime it is important to note that the components presented herein demonstrate only the most basic functionality for each of the device, command, interface and application layers.

The features described for interface layer components (and the mode of communication between these and lower level components) can be assumed to be the same for applications which require direct access to the physiological data stream provided by a given hardware device. The choice of interface layer components described in this chapter is influenced by the review in Chapter 3 of existing systems designed for detecting and processing electrophysiological information. These interface components represent popular modes of signal representation descried in the EPIC systems review. It should be noted that components in the interface layer of our model are referred to in this chapter alternatively as *interface components*, *interface layer components* or *widgets*.

5.2 Physiological Signal Processing in the Device Layer

The feature identified in Chapter 3 as being fundamental to all EPIC systems is the management of physiological information made available to an application through a specialised sensing device. The main device management activities required include the retrieval and pre-processing of hardware generated physiological signal data points. Once retrieved, this data must be transferred from the hardware specific device layer to either the command layer for further feature extraction, the interface for display, or to the application for some other action[27]. The first layer of data manipulation functionality is device-specific, in that the component(s) in this layer talk directly to a particular sensing device, retrieving raw physiological data from it in the form of a digital data stream. Further functionality is required in this layer to convert the digital data stream into a stream of numerical values corresponding to the changing magnitude of a physiological signal, along with identification of which physiological parameter the data point corresponds to. This is shown in Figure 5.3.

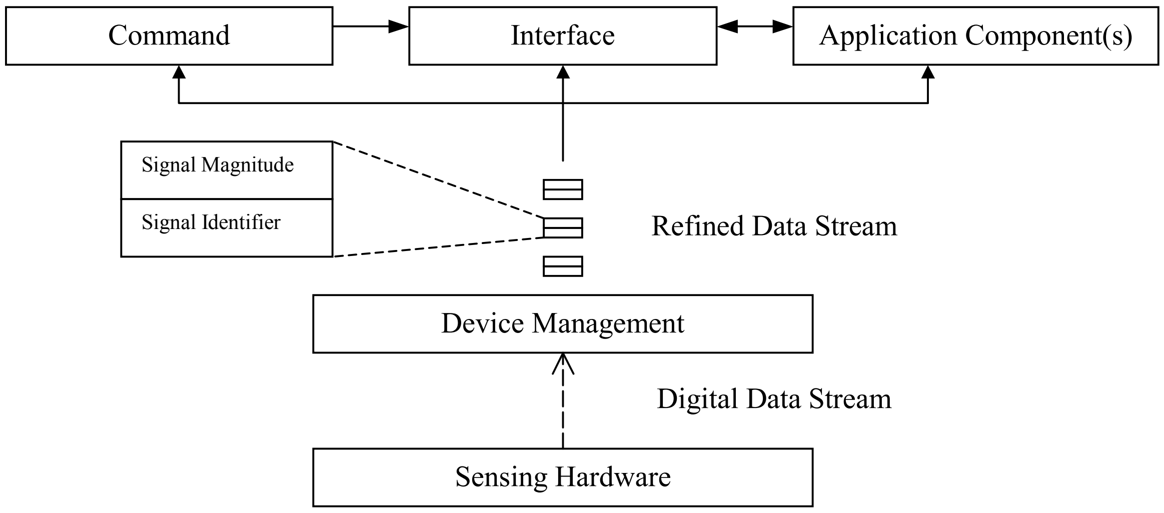


Figure 5.3 First level of physiological signal pre-processing and data distribution

In a bid to separate the device layer (and thus the sensing device) cleanly from the higher layers of this interaction model, a single component[28] will be described in the following section that performs data retrieval from a given sensing device. This component also performs the first level

[27] archiving for example

of pre-processing on the data. This component, which will henceforth be referred to as the device management component (DMC) and will be shown to provide data filtering for the components in the command, interface and application layers.

5.2.1 Device Management Component (DMC)

In its simplest form, the DMC maintains three key pieces of information:

A reference to the sensing hardware
A list of available physiological signals for the device
A list of command, interface and application layer components interested in being the recipients of information pertaining to one or more physiological signals.

Commercial physiological sensing devices, such as those described in Chapter 3 support communication between the sensing hardware and a proprietary software application through a device-specific software component. This software component most often takes the form of a dynamic link library that enables a windows-based application to establish a connection with a particular sensing device.

5.2.1.1 Communication with the Sensing Hardware

The DMC holds a reference to the sensing hardware specific DLL, which contains the serial port access functions for its particular sensing device. Once the DMC has established a connection with the hardware via the DLL, there are two potential models for how the DMC might subsequently receive data relating to any particular physiological signal.

In the first model, the sensing hardware generates events corresponding to new data points at a rate at which it digitises the detected physiological signal(s). The DMC can then pass these events (corresponding to a stream of new data points) onto interested components in higher layers. A major drawback with this approach is that the signal-generation rate of the sensing hardware could, in certain circumstances, exceed the data handling capabilities of a command, interface or

[28] In reality this component may itself consist of several sub-components, but for ease of understanding we shall describe it as a single functional entity.

application layer component that has registered interest in a given signal. In this case either the DMC or the receiving higher level component would need to discard the excess data. These two potential scenarios are illustrated in Figure 5.4a and 5.4b wherein the command, interface and application layer components are labelled *CC*, *IC* and *AC* respectively.

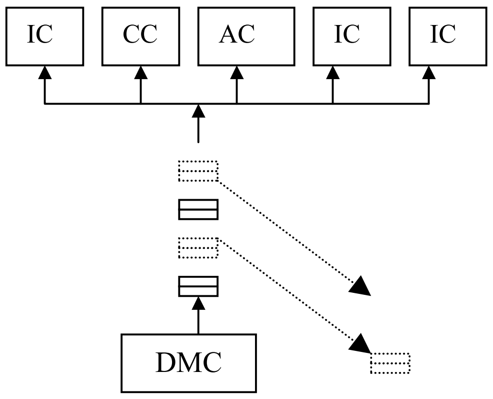


Figure 5.4a DMC discards excess data points

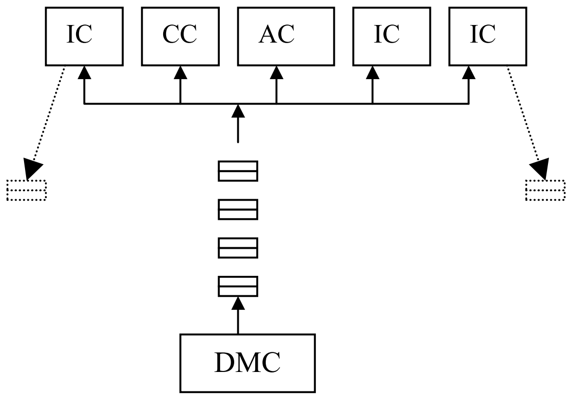


Figure 5.4b Higher-level components discard excess data points

We can assume that the data requirements for individual command, interface and application layer components will generally differ. Consequently the DMC would not be able to discard excess data related to a particular physiological signal without checking whether there was an interest registered in that signal at that instance in time by another component which could handle a high data rate. Conversely, if the discarding of excess data is left to the higher level components themselves, then all data will be transferred to all components with a registered interest in a signal, only to have it discarded by those components not requiring that instance of the data. Both of these approaches waste system resources and thus reduce time available for other activities.

The second manner in which a DMC could receive data from the sensing hardware is by polling it for information about each physiological signal at a rate specified by interested upper layer components. This is a more efficient device management strategy as it reduces the requirement for discarding unwanted data. Consequently, it is the model of data retrieval selected for use in the EPICS toolkit's DMC. This data retrieval strategy will be referred to as being an *application-driven event management strategy*, where the requirements of a particular EPIC system (made up of command, interface and application components) dictate the rate at which the sensing hardware is to be polled.

5.2.1.2 Listing Available Physiological Signals

As illustrated at the end of Chapter 3, each individual sensing device can potentially make data available to an EPIC application about a different range of physiological parameters. In order to preserve the sensing hardware's independence from the command, application and interface layer components, components in each of these layers must have individual access to the list of signals available on a particular sensing device at run time. In order to avoid upper layer components attempting to register for information about a physiological signal unavailable on the current sensing device, the device-specific DMC holds a list of signals available for its particular sensing device.

5.2.1.3 Application-driven Event Management

In order to support an application-driven event management strategy, each higher layer component is required to inform the DMC about both the signal(s) it is interested in *and* the frequency at which it can handle notification events related to changes in each signal's characteristics (i.e. new data points). We must, therefore, extend the list of information that the device management component must maintain to include:

- ❑ Reference to a timer
- ❑ Some sense of the relationships that exist between particular physiological signals and interested upper layer components

The role of the timer is to produce periodic events which trigger the DMC to poll the sensing hardware for new data points. Theoretically, the timer interval could be fixed at run time, but this produces a similar problem to the one caused by letting the sensing hardware drive data into the overall application, in that data may be retrieved in excess of the requirements of any registered upper layer components. If upper layer components are to be allowed to alter their signal requirements whilst the DMC is running (more about this later), then the polling interval is better set dynamically by the DMC as the requirements of registered components change. To achieve this flexibility the DMC uses the frequency data submitted by each registering upper layer component to dynamically alter the timer notification interval in a bid to satisfy the requirements of the most demanding component currently registered with it. At this point in the DMC design, three potential models for passing events between the hardware and the upper layers were identified.

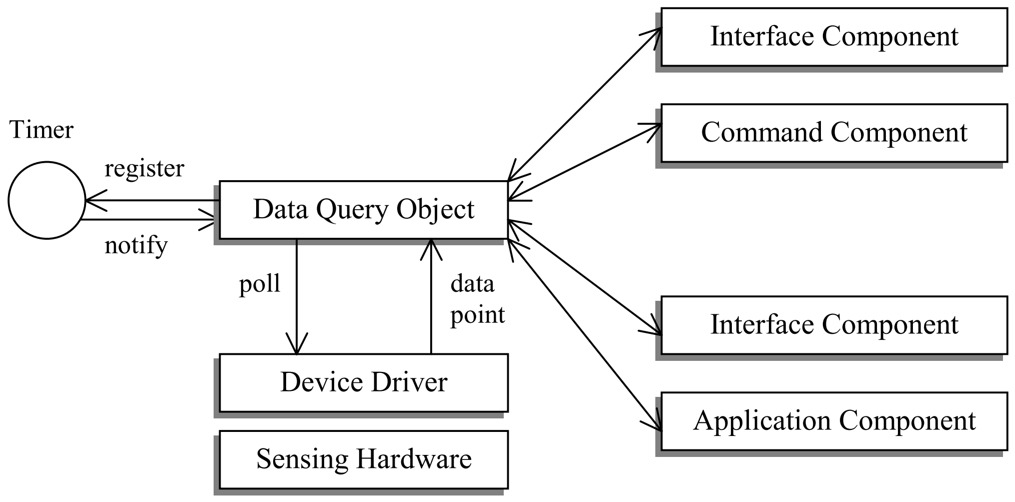


Figure 5.5 Model 1 - A single data query object

The first model is illustrated in Figure 5.5. In this model all command/ interface/ application layer components register with a single entity, which we have called a *data query object*, that exists within the DMC. On notification of a timer event the data query object cycles through the list of physiological signals, retrieves a data point for each one in turn from the sensing hardware. It then attempts to identify which upper level components are interested in receiving new data points corresponding to which physiological signal of interest.

There are several problems with this model however, the first being that a single data query object is trying to manage all of the events occurring between the hardware and the higher layers of the model. For one or two signals and one or two simple upper layer components this may work fine. However it is possible that, an interface layer component will want to register interest in a number of physiological signals. So, for example, a bar graph may want to display information about several EEG component signals along a single scale. A display-dependent EPIC system, such as one designed for a biofeedback-based application, may contain a number of interface components, each responding to a different combination of physiological signals. This begins to greatly complicate the data query object's job of distributing signals on to particular components.

To compound this problem, during a running EPICS application, it may be desirable to change the signal(s) that a higher layer component is interested in, or change the time intervals between receipt of data points from the DMC. This would further complicate an already complex distribution procedure. For these reasons a second model was conceived where each component, as it registered with the DMC, was assigned its own data query object (Figure 5.6).

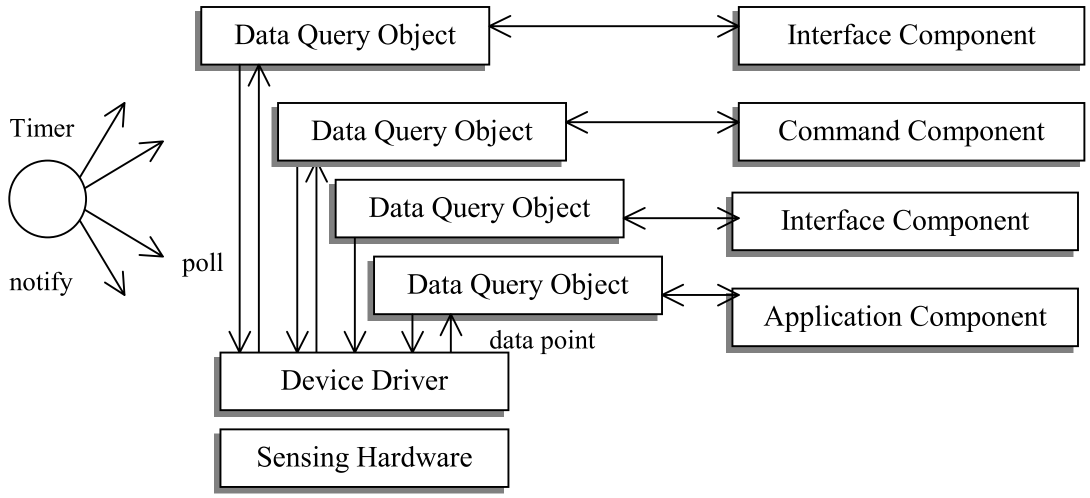


Figure 5.6 Model 2 - Every *higher level component* has a dedicated data query object

In this case each data query object only needs to poll the hardware for a subset of the physiological signals required within the scope of a given EPICS application. The data query object maintains information about the time interval that its particular upper layer component has specified for receipt of data, and consequently polls the hardware at that rate. It also maintains a list of physiological signals of interest to a particular component throughout the lifetime of that component. When the component is removed from the running application (this is equivalent to closing a window in a windows system) its data query object is removed from the DMC.

This model, though better than the first model, nevertheless has some problems of its own. For example, two interface components could have exactly the same data retrieval requirements, in terms of signals of interest and data-rate handling capabilities. However, in this model two separate data query objects would exist within the DMC, both effectively doing the same job. On a particular clock cycle both data query objects would, in turn, poll the hardware for data about the same signal(s). This potential for redundant activity within the sub-components of the DMC has been overcome by adopting a refined third model. This final model is illustrated in Figure 5.7.

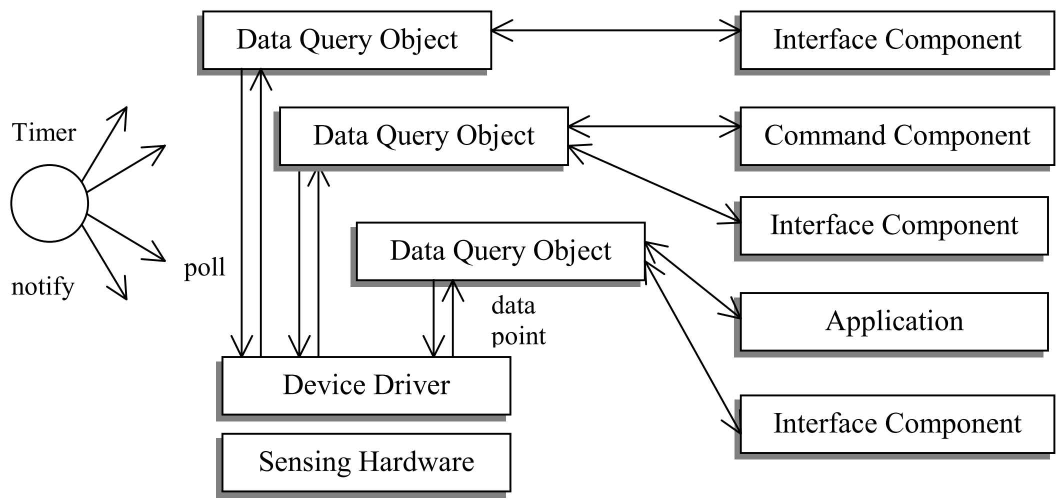


Figure 5.7 Model 3 - Each available *physiological signal* has its own dedicated data query object

In this final model, initialization of the DMC includes the definition of a list of physiological signals available on the associated sensing device. For each signal in the list, a new data query object is created. This data query object is interested only in which components want notification about its particular signal. If an upper layer component wants information about more than one physiological signal, it is registered separately with the data query objects corresponding to each of those signals in turn.

5.2.1.4 An Overview of the DMC

It is worth saying a little more about the functional sub-components encapsulated by the DMC. The role of the data query object has already been described as being to retrieve data from the sensing hardware about a particular physiological signal. The DMC maintains an array of data query objects, one for each physiological signal available from the particular sensing device. This is illustrated in Figure 5.8. When a higher level component such as an interface widget registers with the DMC to receive information about one or more physiological signals, the DMC registers that widget with each data query object associated with the retrieval of a signal of interest to it. The widget is duly added to the list of listeners maintained by each signal-specific data query object. As shown in Figure 5.8, each data query object in the array maintains itself a list of the data requirements associated with each higher level component registered with it. Each data query object notifies the higher level components in its list directly about new data points retrieved from the sensing hardware.

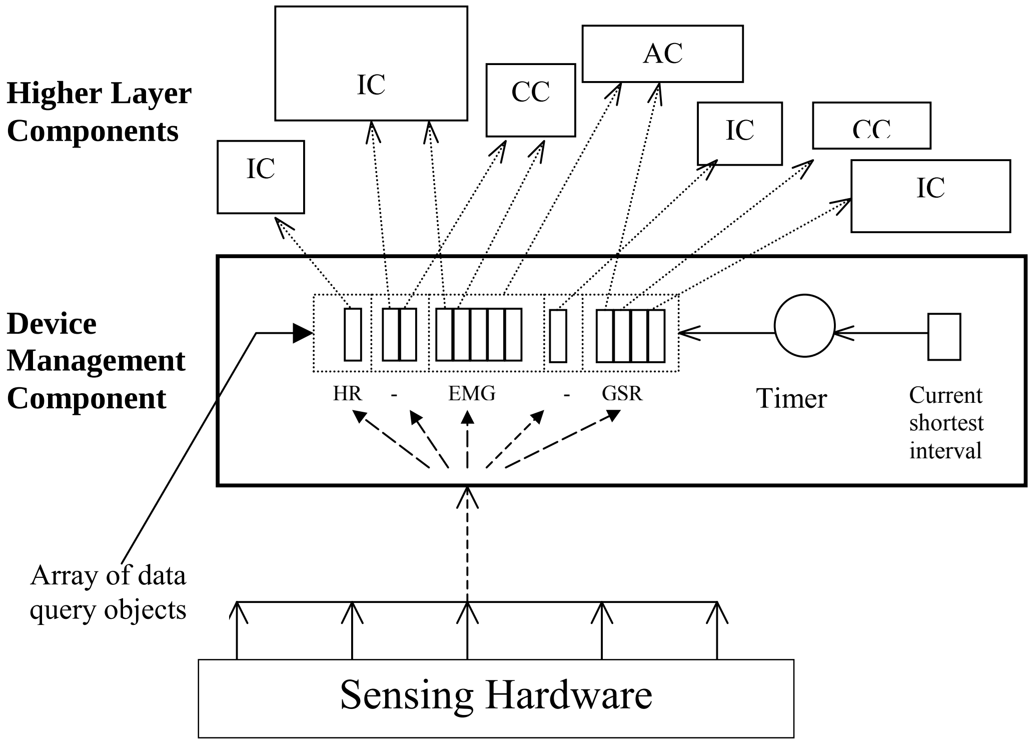

Higher Layer Components
IC
IC
CC
AC
IC
CC
IC
Device Management Component
HR
-
EMG
-
GSR
Timer
Current shortest interval
Array of data query objects
Sensing Hardware

Figure 5.8 DMC managing the distribution of data using physiological signal-specific data query objects

Whilst registering a new upper layer component, the DMC takes an opportunity to register *itself* with that component in order to listen for subsequent events which that component may generate. The events that the DMC is interested are related to either a change in data rate requirements or a change in physiological signals of interest.

5.2.1.5 Data Rates and Timer Intervals

As previously stated, it is envisaged that different command, interface and application layer components will have different requirements regarding the rate at which they can handle data from the sensing hardware. With the existing components of the toolkit this is not too much of an issue because, as will be shown, they are functionally quite simple. If we think about more complex sensing-based systems, however, it is reasonable to speculate that a component or set of components, which together make up a VR system for example, may need to load balance between input devices in order to concentrate processor time toward critical tasks. There will be periods when the response time for tracking the user via a head-mounted display and data gloves for example takes precedence over the incoming physiological data stream. In this case, the VR application may signal to the relevant data query objects within the physiological sensing device's DMC to turn the signal rate down.

Higher level components expose methods to the DMC for changing their associated timer interval preferences. It is possible to bind the value associated with a component's required timer interval in such a manner that any interaction with the component that causes this value to change creates a notification event. The DMC receives notification of this event by which it knows that the component's data requirements have changed. It can then retrieve the new interval value from the component.

If the new interval value is smaller than the timer's current shortest timer interval (which is the rate at which the timer is currently notifying data query objects), this value is replaced and the timer changes its behaviour accordingly. In this way, the timer is always trying to satisfy the data requirements of the most demanding component currently registered with the DMC.

The current timer interval value is available to each of the data query objects polling the hardware, which are in turn all subject to notifications at that interval. Each data query object holds a list of the timer intervals associated with the higher level widgets registered with it. This information is provided via the larger DMC - initially when a higher layer component registers with the DMC and subsequently when the DMC receives notification that a component's requirements have changed.

When a data query object polls the hardware for data about its particular signal of interest, rather than pushing the data out to the registered command/ interface/ application layer components, it notifies each registered component in turn that it has received a new data point. The onus is on the higher layer components to retrieve the new data point from within the data query object. Figure 5.9 illustrates an interface level component receiving notification from a physiological-signal specific data query object it is registered with. The notification is of a new data point retrieved from the sensing hardware by the data query object. The interface component responds to the event by polling the data query object for the new data point, which is duly returned to it.

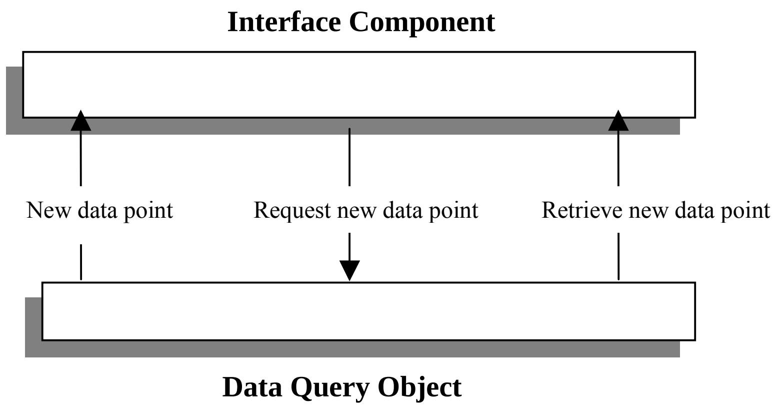


Figure 5.9 Series of events (left to right) associated with the retrieval of a new data point

There are two scenarios influencing the *frequency* at which data point notification events are generated by a particular data query object:

1. *The Timer's interval matches that of one of the data query object's currently registered components*. In this case, the data query object polls the hardware for data on receipt of each Timer notification event. If the data query object has other registered components whose time intervals are longer than the timer's current interval, it calculates how many timer notifications (and subsequent polls for data) should pass before it needs to notify each of those components that a new data point is available. This enables components with lower data frequency requirements to receive notification of new data points on every Nth timer notification, where:

   N = Component *f* / Shortest interval *f*

   For example, if the current timer interval is 0.1 seconds, and the requirements for a particular interface layer component is for notifications at 0.5 second intervals, then the interface component will only be notified of a new data point every 5th data query object hardware poll.

2. *The Timer's interval is shorter than that of any of the data query object's currently registered components*. In this case the data query object looks for the shortest interval associated with its registered components. It does a similar calculation to the one above in order to decide how many timer notifications to let pass before it actually polls the hardware for data. And as

in the previous case, it also calculates when to notify each of its registered components of the availability of a new data point.

5.3 Physiological Signal Processing in the Command Layer

It was stated in the introduction to this chapter that the second layer of physiological signal processing corresponded to advanced feature extraction from the physiological signal data stream with a view to translating features in the data stream into control commands. It was demonstrated that the complexity demanded for the second level of physiological signal processing was application-dependant and in a number of cases involved feature extraction from the data stream using a neural network. As the aim of this toolkit is to provide proof of the design concepts introduced here, a fairly simple second-level signal processing component has been created. This single command component extracts information from the physiological signal stream pertaining to the magnitude of a given signal, and acts like a switch, changing its own state depending on the magnitude of that signal.

Such a switching component can behave in either a latching or non-latching manner. In both instances a predefined threshold value is set within the component, which acts as the switching point. For a latching switch, if the value of a data point retrieved from the sensing hardware and corresponding to a given signal's magnitude is above that of the threshold value then the switch is set. If a subsequent amplitude signal value falls below the threshold value, nothing happens, the switch remains set. The next time a value is received above that of the threshold, the switch is unset. For a non-latching switch, a received data point value above the threshold value sets the switch and the next data point value below the threshold, unsets it.

In Chapter 3 Dewan's brain-controlled Morse code system relied on the amplitude of an *alpha* signal to switch between two states. Prosthetic limbs also incorporate switching based on the amplitude of one or more EMG signals. A software component that models a switching command may be used for either application. Alternatively, it could be linked to an animation, sound or video clip that runs only when the switch is set. Or it may be used as the equivalent of a single mouse button event. Latching switches are particularly useful for implementing 'sticky keys' which are useful to motor disabled individuals. The inclusion of a switch command between the DMC and an interface component enables the continuous physiological signal stream to be used as if it were a series of discrete events. So, for example, an interface component could present a button and the switching component in the command layer could facilitate discrete event driven operation of that button, but through a continuous data stream.

5.3.1 Encapsulating Common Interface Features

Early on in the design of the toolkit, a number of features were identified as being common to all electrophysiologically-responsive entities, be they command, interface or application level components. These common features included certain data values plus some common event handling functionality. In order to provide maximum flexibility (and consequent extensibility of the existing toolkit, as we shall see later) these common features have been encapsulated within a single generic software object. This software object effectively serves as the communications interface for any potential command, interface and application layer components. The composition of this generic object is described in Table 5.1.

Properties	
Name	*Description*
currentValue	A data point corresponding to the current value (amplitude) of a physiological signal
signalID	Identifier corresponding to one of a predetermined (by DMC) set of physiological signals
rewardThreshold	This property used to set a switching value (see Section 5.3.2 below)
maxScale	Formatting of some interface widgets requires information about the maximum and minimum potential values for a given signal
minScale	
eventInterval	This is the timer interval, which a widget makes available to Data Query Objects handling data about particular signals that widget is interested in.

Events	
Related Methods	*Description*
propertyChange addPropertyChangeListener removePropertyChangeListener	Other entities outside of a widget component will want to be notified of changes in the following properties: • signalID • eventInterval • currentValue

Table 5.1 Generic Object – the common interface provided by all components in receipt of pre-processed signals

The *Generic Object* provides the functionality required by a command, interface or application component to both retrieve and store a data point retrieved by a data query object it is registered with. When a component registers with the DMC, it exposes a series of properties, including an identifier corresponding to the signal it is interested in receiving information about, as well as information about the frequency at which it wants to receive notifications about new data points (the *eventInterval* property in Table 5.1). When a component first registers with the DMC, it does not know what signals are available on the particular sensing device that the DMC is abstracting over. For this reason, the *signalID* property initially stores a dummy value until it has registered with the DMC, when the list of available physiological signals that is held within the DMC is exposed to it. This is illustrated in Figure 5.10 (the variables labelled collectively as being *User-set values* are described in subsequent sections of this chapter).

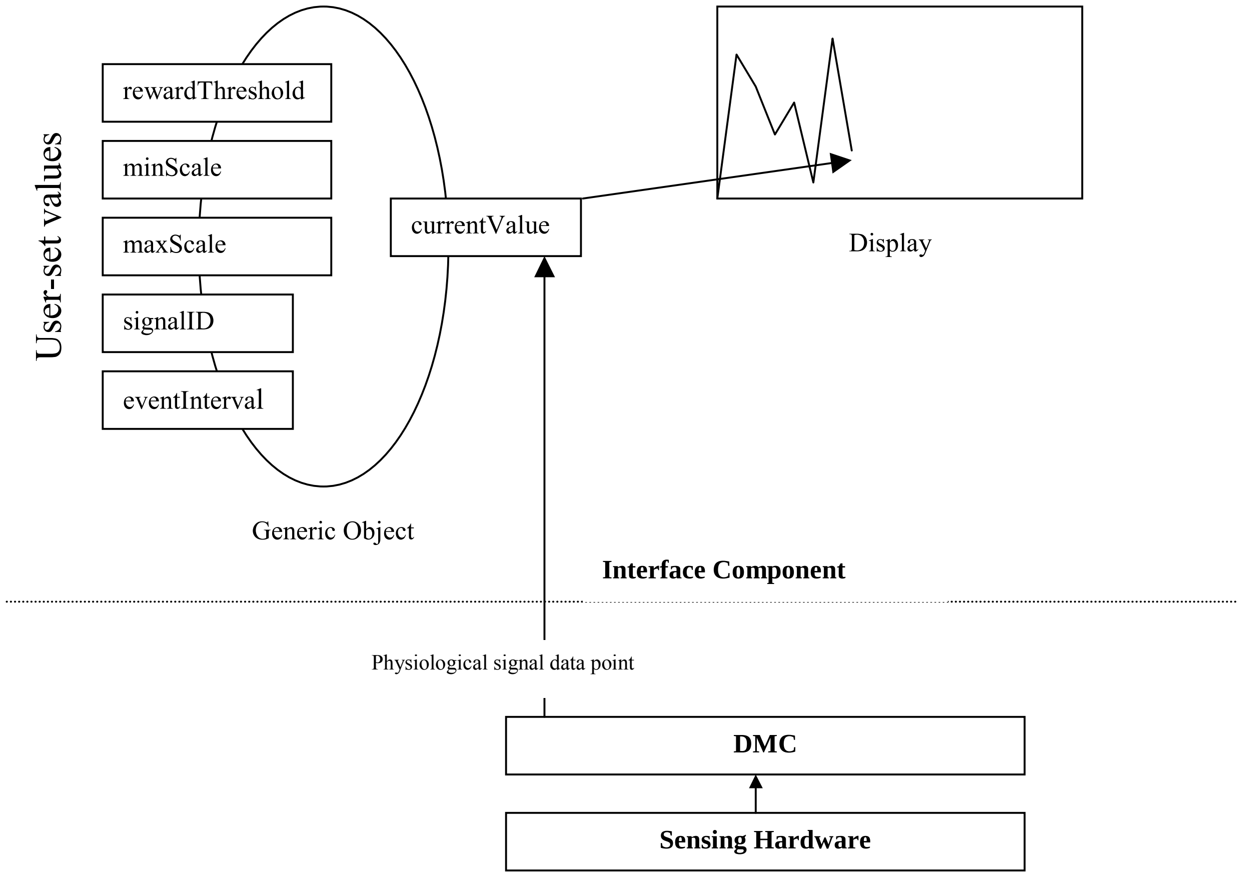
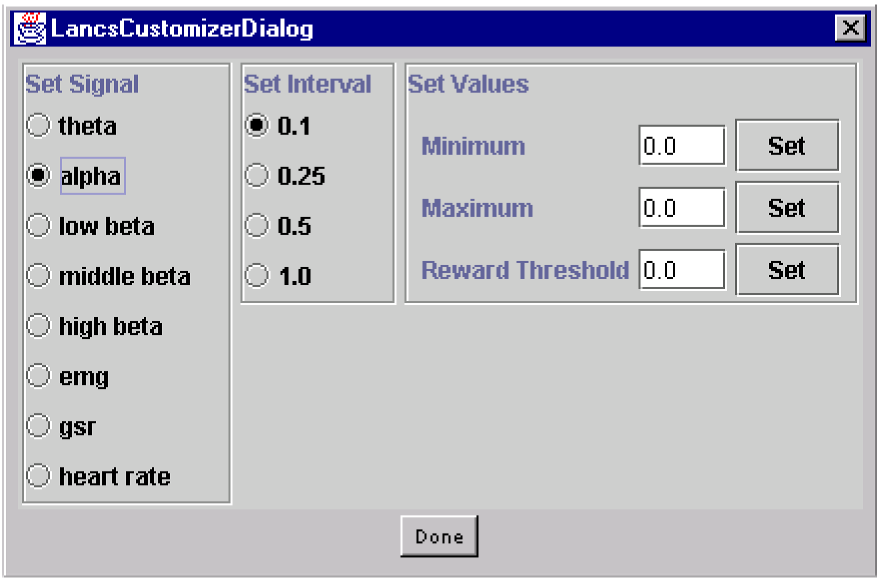


Figure 5.10 DMC using the generic object interface to send data to an interface layer component

5.4 Interface Layer Components

The main function of components in the interface layer of the model is to reflect the changing status of different physiological signals. Usually this involves directly translating each new physiological signal data point received by an interface component into an animation event within that component. Examples of potential animation events include drawing new points on a polygraphic animation or the changing pitch of a musical note. Due to the composite and encapsulated nature of components, an interface layer component could be an entire virtual environment or a computer game. In these cases an animation event triggered by receipt of a new data point could relate to an object, or the viewpoint of a subject moving within a 3D computer-generated environment. In other words *physiologically interactive interface components could be far more complex entities than those classified traditionally as interface widgets*.

At interface component-DMC registration, the user of the interface component explicitly selects a signal of interest for that component. In order to do this, a list of available signals must be made available to the user. This is achieved through the use of a component customizer dialogue. Every interface component has a copy of the customizer dialogue (Figure 5.11), which retrieves at initialisation a list of available signals and timer intervals from the DMC.



Set Signal: theta [✓ alpha] low beta middle beta high beta emg gsr heart rate

Set Interval: [✓ 0.1] 0.25 0.5 1.0

Minimum 0.0
Maximum 0.0
Reward Threshold 0.0

Figure 5.11 Interface component customizer

Once the signal selection has been made, the component is automatically added to the list of listeners on the data query object for that signal. A component's initial notification interval is a default value (0.0 = *do not notify me*) which also has to be set explicitly in order for the data query object to begin to notify the component of the retrieval of new data points. On notification of a new data point event from its referencing data query object, an interface component's *propertyChange* method retrieves the data point and stores it in the property *currentValue*. A class extending the generic object class, within which all of this functionality is encapsulated, would use the value of this property to influence its behaviour, usually updating the display of the encapsulating component.

5.4.1 Discrete and Continuous Interface Widgets

Some interface or application level components will behave in a manner analogous to that of traditional interaction techniques such as buttons and switches which are designed to be operated by discrete user events generated by discrete input devices. One example of a discrete interface component is a reward-based biofeedback application, such as the PacMan game described in Section 3.2. Another example would be a component driving a prosthetic hand that was only *on* (i.e. gripping) when a signal magnitude was above a certain value, and was *off* (i.e. release grip) otherwise. These components only act on data points they retrieve from a switch command component which they register with, and which in turn is registered with a data query object in the DMC, as shown in Figure 5.12.

The switch command is a continuous component, as it inspects every new value it retrieves from the DMC to see whether it is above or below its switching threshold value. A component registering with a switch component responds only to the discrete events produced by the switch and corresponding to a change in its state. These registering components can therefore be classified as discrete components.

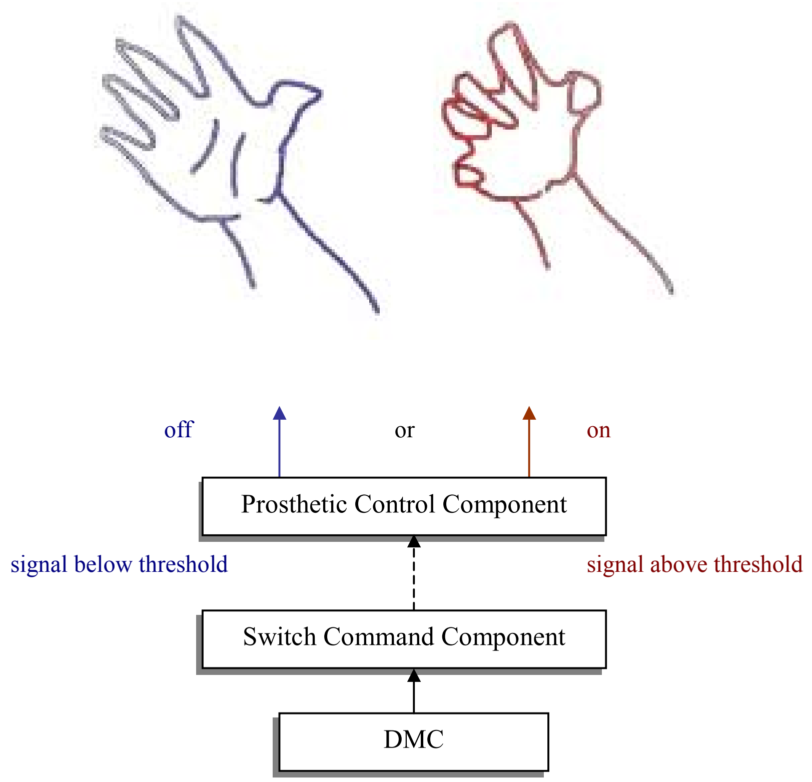


Figure 5.12 Discrete use of continuous physiological signals

In Chapter 2 it was shown that medical monitoring and biofeedback applications often display real-time physiological signals as dynamic polygraphic traces. An interface component that displays a physiological signal polygraphically is responding actively to every data point corresponding to a particular physiological signal of interest. Furthermore, a component such as a polygraphic display requires no further processing to be done on a signal beyond that carried out by the first level of pre-processing undertaken by the DMC. Therefore a component can register itself with the DMC, and retrieve signal data directly from a signal-specific data query object. Components that utilise every new data point in this manner can be classified as continuous components. We will now examine a number of such components.

5.4.1.1 Polygraph Component

The Polygraph component expects as input a stream of data relating to the changing characteristics of a single physiological signal. The X-axis of the Polygraph displays the amplitude value in microvolts. The range for this scale is set by the user of the component through the Polygraph's associated customizer dialogue. The amplitude of each new data point is plotted against the X-axis.

The Y-axis displays frequency information, each increment on the scale corresponding to the timer interval set via the customizer dialogue (which will be the time elapsing between new data point notification events). The Polygraph component stores the value of the last data point when it retrieves a new data point from a data query object. On retrieval of a new data point, it draws a line the between the last amplitude value and the new amplitude where the new value point is positioned at the next timer increment point.

Any interface component like the Polygraph that provides a numerical scale against which to display the current value of a physiological signal requires formatting information for that scale in the form of boundary values. So for example, a temperature gauge might be bounded at 0ºF and 40ºF, making the boundaries of the displayed scale 0 and 40 respectively. Two of the properties described in Table 5.1 - *minScale* and *maxScale* (accessible through the customizer dialogue fields *Minimum* and *Maximum*) are used to provide this formatting information to the component's display.

If a new data point contains an amplitude value that falls outside of the current range, either above the current *Maximum* or below the current *Minimum*, the line from the last point to the new point is drawn along the top or bottom of the screen. Whilst an application is running, the user can change the *Maximum* and *Minimum* values through the Polygraph's customizer dialogue. These changes will be reflected in the appearance of the Polygraph on the next update (new data point received/ redraw the Polygraph). When the Polygraph trace (the red line in Figure 5.13) reaches the right-hand side of the Polygraph, the display is subsequently shifted one timer interval to the left on every successive update event. Thus the Polygraph's panel scrolls beneath the trace line to ensure that the new line remains on the right of the screen.

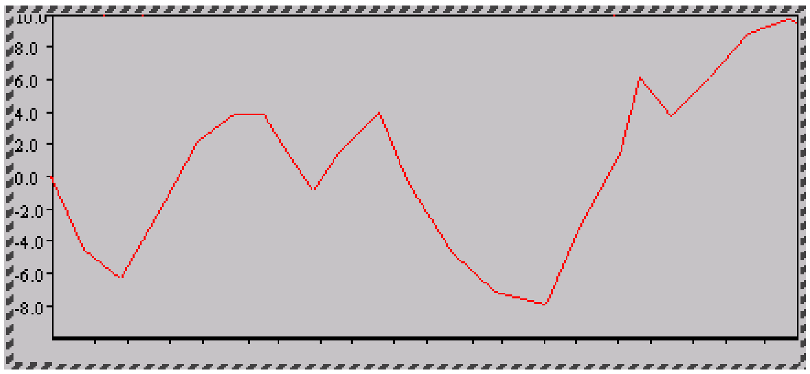


Figure 5.13 Polygraph component

5.4.1.2 Bargraph Component

The Bargraph interface component uses the same display panel as the Polygraph, but can accept and display data corresponding to multiple physiological signals. The Bargraph component creates a multiple signal data structure to hold an array of data points corresponding to one array cell for each signal it is interested in. On registration with the DMC, the Bargraph component registers a single timer interval request for all of the signals it is interested. On notification of new data events from associated data query objects, the Bargraph collects the new data points. When it has a full set, it redraws the graph. This is illustrated in Figure 5.14.

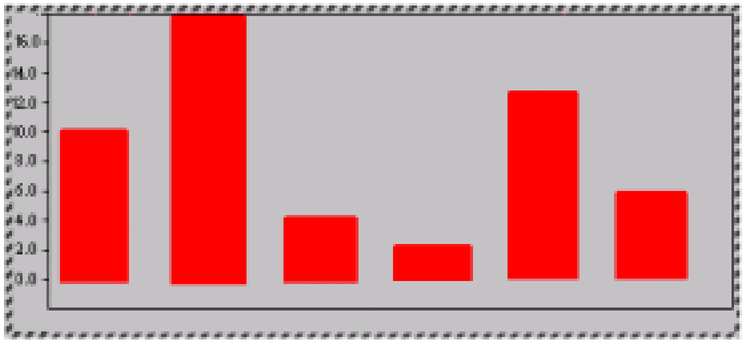


Figure 5.14 Bargraph component

5.4.1.3 Other 2-Dimensional Animation Components

Three additional widgets that give more abstract feedback about the amplitude of a signal are the Temperature widget, the Intensity widget and the Slider widget. Like the Polygraph, these are continuous widgets, in that every new data point causes their appearance to change. The Temperature widget is a square panel whose colour changes dynamically in relation to the current value of a signal. Its input is a single physiological signal. *Maximum* and *Minimum* values serve as the boundary values, with *Maximum* being a "hot" colour (red, for example) and *Minimum* being a "cold" colour (blue). Higher amplitudes produce hotter colours lower amplitudes produce colder colours (see Figure 5.15).

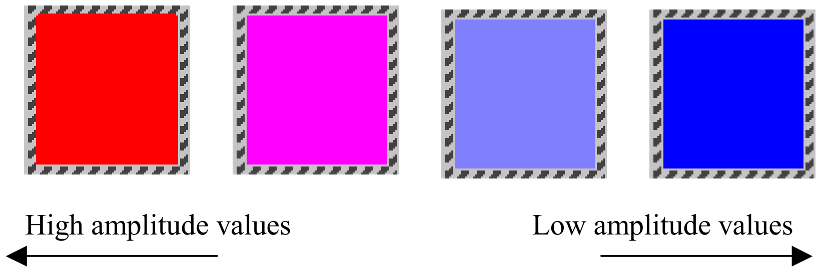


Figure 5.15 Coloured responses of a Temperature component

The Intensity component is almost identical to the Temperature component, except that the colour related to the *Minimum* value is white and the *Maximum* value is black, and the changing value of the signal's amplitude is displayed as corresponding shades of grey. The third animation widget incorporates the same graduated axial scale used by the Polygraph, and displays the current amplitude of a signal as a single sliding bar displayed against the scale (Figure 5.16).

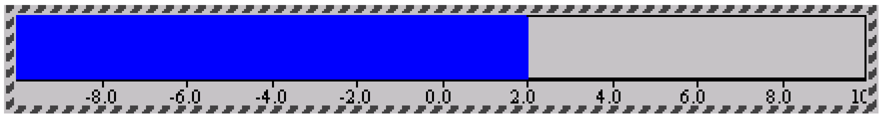


Figure 5.16 Slider component

5.4.1.4 Multimedia Components

The Video and Audio widgets are contrived examples of discrete interface components. Both register with a Switching command component to receive binary state information about the magnitude of a single physiological signal. The Video widget contains an embedded AVI video clip and the Audio widget contains an embedded WAV audio clip. In both cases, the clip runs only when the component receives notification from its Switching command component that the amplitude value of a physiological signal is above that of the Switching component's r*ewardThreshold* variable. The value of this switching threshold is set for an individual Switching command component through the interface components customizer dialogue (Figure 5.11) and passed onto the command component which consequently switches on that threshold. When the value drops below the threshold the video/audio clip pauses.

5.4.1.5 A VR Interface Component

A VR component is a good example of the complexity that can exist in an interface layer component. This interface widget is a 3D environment - effectively a whole virtual reality world (environment plus world model) albeit a small one - which takes as one of its inputs a physiological signal data stream.

The VR component is a continuous component that can use a physiological data stream in place of or alongside other continuous input devices of the kind described in Chapter 4. The changing amplitude of a physiological signal could be used to effect the appearance or behaviour of artefacts within the virtual environment. On the one hand physiological information could be used for hands-free navigation or interaction with particular artefacts the virtual environment. On the other hand, the physiological data could be used to provide *affective* information to the whole environment. Affective state information, like other environmental constraints such as gravity, would be used to effect some aspect of the entire environment.

The virtual environment shown in Figure 5.17 takes a single physiological signal as input and uses its changing amplitude to raise and lower the block on the right-hand side of the screen. This particular component was created for a specific application, which will be described fully in the following chapter.

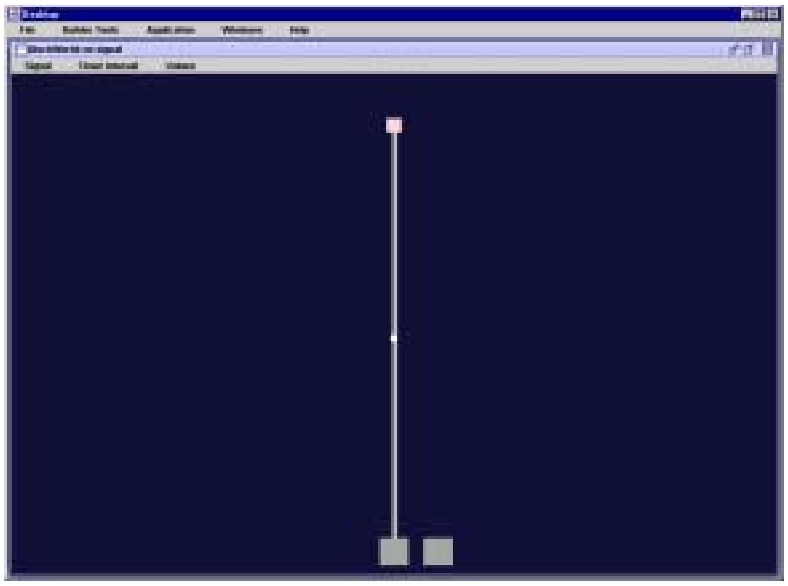


Figure 5.17 A VR interface component

5.5 Application Layer Components

Within the scope of our component-based system design strategy, Application Layer components constitute any modular parts of the system software that do not provide either advanced signal processing (which is a defining feature of components in the Command Layer) or signal presentation (a feature of components in the Interface Layer).
Our model of interaction from Figure 5.2 indicates that components found in the Application Layer could be the recipients of physiological data via at least two routes. In the first instance, an Application Layer component can register itself with the DMC to receive one or more physiological data streams directly from the first level of signal pre-processing. An Application Layer component that aims to be the direct recipient of physiological data from the DMC simply needs to utilize the generic object class (by extending it) in the same manner as Interface components described above. One example of such an Application Layer component would be a component whose role is to archive data from the sensing hardware for later analysis. We shall consider such a component in more detail in a short while.

Components in the Application Layer may alternatively wish to receive physiological data that has been more extensively processed. In such cases, the advanced signal processing would be carried out by a Command Component, which would in turn be the direct recipient of data from the DMC. An example of such a situation might involve an EPIC system that has built-in autonomous functionality to be utilized in the case of loss of normal human operator functioning. An Application layer component could be notified by the Command Component of distinct physiological changes in a system's human operator which may be indicative of that operator experiencing a hazardous state of awareness or loss of consciousness. The Application Layer component could therefore manage the system's response to such situations, by alerting the operator where appropriate or taking over control of the system if necessary.

For the purposes of completeness the following section will outline the one more component in the toolkit – a simple component for archiving physiological data received directly from the DMC.

5.5.1 Archiving Component

The final component in the existing toolkit provides archiving capabilities. The Archiving component registers directly with the DMC to be the recipient of a single physiological signal data stream. On notification of a new data point received by the physiological signal's data query object within the DMC, the Archiving component retrieves a value and writes it out to a text file. Archived signal data is built up into a list of comma-separated value pairs. These pairs correspond to a time stamp (added by the Archiving component) plus a floating-point number. The floating-point number corresponds to the magnitude value of the signal data point. An example of the data format for the archive file is illustrated in Table 5.2. The formatting of the text file is deliberate, as it enabled an archived file to be imported directly into a data analysis package such as Microsoft Excel.

```
"Time","Peak-to-Peak Amplitude"
0.125,30.77
0.250,71.30
0.375,103.83
0.500,32.07
0.625,28.25
0.750,53.12
```

Table 5.2 Data file format of the Archive component

5.6 Implementation Details

The components that make up the EPICS toolkit are implemented in the Java language and adhere to the JavaBean standard component specification (Englander 1997). A Java component, which consists of a set of Java classes and their associated resources, is referred to as a bean. A bean is written, compiled and packaged into a Java archive file, which includes any other precompiled resources that that bean requires. Once archived, a bean is ready for deployment. Java beans can be deployed within bean builder tools that incorporate containers called *bean boxes* that support the introspection of Java archives. Introspection involves the use of reflection services provided by the Java language to discover the methods properties and events exposed by a bean. Introspection relies on the use of standard naming conventions for the methods, events and properties that constitute the three defining features of Java beans.

The toolkit was developed on a 200Mz Pentium II computer using Sun's Java Development Kit (JDK) v1.2, Swing (Eckstein 1998) and the Java3D API. The sensing hardware over which the toolkit has been developed is the WaveRider by WaveAccess (WaveAccess 1997). The WaveRider is an 8-channel, multi-signal physiological sensing device, which is supplied with its own dedicated Windows application, called WaveWare. The development package for WaveRider consists of a native code DLL that can be used to retrieve physiological data from the WaveWare application. Unfortunately, this design effectively embeds an entire Windows application between the DLL and it is this application that is, in reality connected to the hardware and through which all requests for data are made. Having an entire Windows application running to do such a simple task produces an undesirable event-processing overhead. The design of the toolkit reflects systems where direct contact is established between the DMC and a simpler native code device driver.

The functionality for accessing the sensing hardware provided by the WaveRider-specific DLL has been mirrored by a single Java component, which is the DMC described above. In order to both establish a connection with the hardware and thereafter receive physiological signal data, a translation layer must exist between the DMC, which written in the Java language, and the DLL which is written in C code. This translation is facilitated through compiling Java Native Interface (JNI) functionality into both the DLL and the DMC. The JNI performs the translation between native code data types and Java data types, which is required before the data is suitable to be passed on to the command, interface and application components. The device-specific native DLL and the DMC together constitute the device layer of the EPICS toolkit. This is illustrated in Figure 5.18.

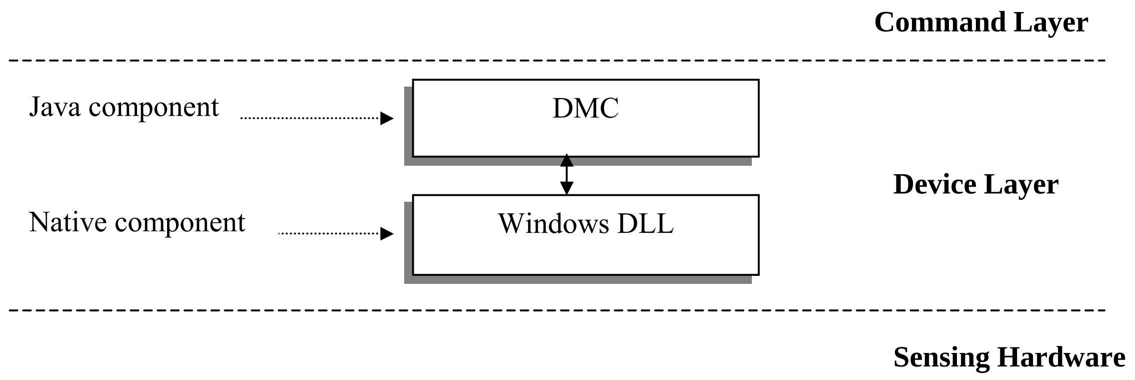


Figure 5.18 Two components making up the device layer

A connection is established between the DMC and the hardware through a single call to a native method held within the DLL. Another native call is available to break the connection with the hardware. Between calls to these two methods, a third method is called (on each DMC Timer notification) to return a signal-related data block from the WaveWare application. The format of this data block is not relevant, as it is a device-specific feature. What is important is that a single floating point value corresponding to the signal magnitude value of a particular physiological signal can be extracted from the retrieved data block by the relevant data query object within the DMC. If the WaveRider were replaced with an alternative device, and the DMC was rewritten to run with the new device, then it would still aim to provide to other components registering with it, a stream of floating point values corresponding to the magnitudes of signals detected by the new sensing device.

In order to provide a uniform interface to the DMC, all higher level components in the toolkit extend the generic object described by Table 5.1 above. As the signal identification and timer interval request functionality is embedded in the generic object, along with the ability to handle notification events, components extending this class need only worry about what to do with new signal data when they receive it.

Java beans inter-communicate by sending notification events and invoking method calls. An example is illustrated in Figure 5.9 above, where receipt of a new data point is shown to cause a data query object to notify an interface component that is registered with it. This notification event causes the interface component to call a method within the data query object, which returns the new data point to the interface component. As all Java beans maintain this communications model, so components can be easily composed together to produce multi-level pre-processing and multimedia signal presentation interfaces. The following chapter describes how this can be achieved through composition of the toolkit components.

5.6.1 Discussion

This chapter has described a set of components suitable for the rapid prototyping of simple EPIC systems. The lowest level of functionality is provided by two components that together make up a sensing device-dependant pre-processing layer. This first level of pre-processing involves the retrieval of data from the sensing device and conversion of this data into numerical values corresponding to the magnitude of a given physiological signal. Each numerical value can then be passed onto command, interface and application components. The second level of pre-processing is application-specific, and generally speaking its role is to extract features from the data stream and convert these into commands. Components that are designed for inclusion in the interface or application layers may or may not require second level pre-processing on the data stream depending upon whether they are discrete or continuous components. Continuous components receive a data stream directly from the device layer (first level of pre-processing). Discrete components respond to specific commands generated by the command layer (second level of pre-processing).

Due to limited device dependency within the toolkit, an EPIC application which was constructed using the components presented in this chapter would only require the replacement of the device layer in order to run with another physiological sensing device, regardless of the different signal sensing capabilities of that new device. This is because all of the details regarding physiological signals available with a given sensing device are encapsulated in the device layer. Limited device dependency satisfies one of the criteria for the toolkit laid out in the introduction to this thesis - that the interaction techniques of the toolkit should be independent of the sensing hardware. Device independence has been achieved by factoring out the first level of pre-processing functionality from subsequent pre-processing, and from both the interface and application entities within the interaction model.

Due to both the choice of standard development technologies (all components are Java beans) and the provision of a generic interface class which encapsulated the functionality required to exchange information with the DMC, the toolkit is easily extensible. When registering a new component, the DMC expects to be passed an object of the *Generic Object* type. Therefore, by simply extending the *Generic Object*, new command, interface and application level components may register with the DMC and exchange events with it and between each other in the manner described above.

Many toolkits designed for the creation of novel interactive application, such as those for VR introduced in Chapter 4 require the learning of special languages in order to build application and add new devices. Developers wanting to both create EPICS using the toolkit components and/or add custom components to suit the needs of their particular application need knowledge of only one standard programming language, namely Java. Furthermore, due to the increasing take-up of this popular high level language, support for the development of Java applications and beans in particular, is widely available. The following chapter will expound upon some of the features of Java beans, including their interpreted nature and the ability this provides to compose beans into runnable applications using drag and drop techniques provided by freely available composition tools.

5.7 Summary

In this chapter, a conceptual model of sensing-based interaction has been used to inform the design of a toolkit for building systems which sense, process and display electrophysiological signals. This has involved a shift from high-level design concepts discussed in Chapter 4 to relatively low-level implementation details. The rationale behind discussing low level implementation details in this chapter is that it provides useful information to the builders of subsequent tools for designing applications that incorporate other types of sensing equipment as well as for others wishing to both use and extend the toolkit presented here.

The existing toolkit consists of a number of Java components which can be combined to reflect the high level entities presented by McMillan's (McMillan 1995) sensing model of interaction. The components described in this chapter are in no way intended to be an extensive set, but rather serve to demonstrate the flexibility offered by a component-based toolkit for creating multimodal, multimedia interactive systems. The architecture of the electrophysiological sensing system illustrated at the beginning of this chapter can be finally re-presented here (Figure 5.19) envisaged as a component-based architecture achievable using the toolkit components.

The flexibility provided by a component-based solution (and created using a standard technology) will become increasingly obvious as, in the following chapter, a description of the toolkit in use is provided. The following chapter describes the toolkit components being composed in a bean box environment in order to demonstrate running simulations of simple EPIC systems. It also describes the construction of a traditional windows-style interactive application with embedded biofeedback capabilities provided by the toolkit components.

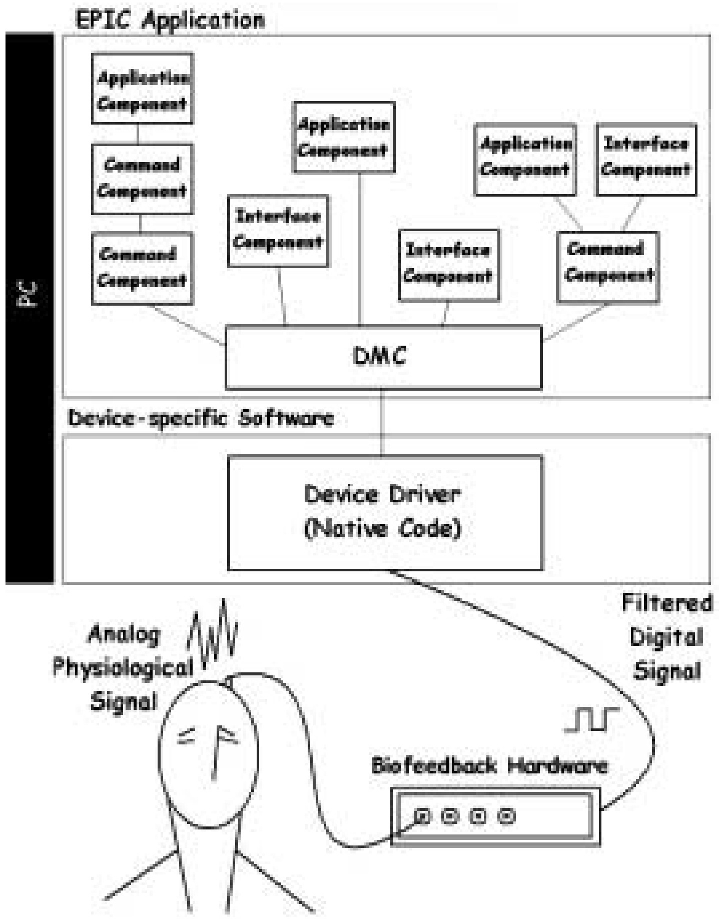


Figure 5.19 A component-based electrophysiological sensing system

Chapter 6

Evaluating the EPICS Toolkit

6.1 Introduction

The previous chapter described the design and implementation of a suite of software components that together constitute the current EPICS toolkit. The description of the toolkit also included an overview of the communications infrastructure that must exist between the device and components in the toolkit in order to support the use of multiple, continuous physiological sensing devices.

This chapter systematically evaluates the use of the toolkit in both the design and implementation of an electrophysiologically interactive system in terms of a biofeedback-based application. Before proceeding with this however, it will be useful at this point to revisit the objectives of this thesis and review our progress so far toward meeting these.

6.2 Thesis Objectives Revisited

In the introductory chapter, three major objectives were laid out for the work to be presented within this thesis. These were:

1. Identify the requirements for building interactive systems that incorporate physiological sensing technologies
2. Identify a suitable model of interaction for systems based on sensing
3. Based on the satisfaction of the first two objectives, design and implement a device-independent software toolkit suitable for building interactive systems that incorporate human physiological information.

The first objective was fulfilled within Chapters 2 and 3, where an assessment was made firstly of the existing sensing technologies and then secondly of the use of physiological information within a wide range of interactive applications. By examining the requirements of these interactive applications, in terms of physiological signal detection, processing and presentation, it was possible to identify common features suitable for support through provision of code fragments (the traditional format for software toolkits).

Chapter 4 fulfilled the second objective by presenting an exploration of existing high-level models of human-machine interaction. This began with an examination of the current most popular model, which is based upon windows, icons, menus and pointers (WIMP). This model was used to reason about the relationship between high level models of interaction, and the tools that subsequently emerged to support development of interactive systems adhering to those models. Moving on to explore models of interaction that incorporate multiple human modalities we identified a model of human-machine interaction that is based on *sensing*. This model was shown to reflect the interaction that takes place between a user with a range of continuous computer input devices including those required for voice- and gesture-based exchanges. We identified at this point that electrophysiologically interactive computer systems are a subset of sensing-based interactive technologies. It followed then that this high level model of interaction, which is based upon McMillan's (McMillan 1995) Structural Coupling Paradigm, was the model we require in order to describe human-EPIC system interaction.

The third and final objective was fulfilled in Chapter 5, where the sensing model of interaction was modified to more closely represent the architecture of an interactive EPIC system. This refined model was used in conjunction with information gleaned from previous chapters to design a suite of software components. Together, these software components demonstrate the basic functionality seen to be common across the range of EPIC applications described in Chapter 3.

The modified model of interaction based on sensing is constructed in such a way as to separate application-specific functionality from the underlying sensing hardware. This is mirrored in the toolkit components, which are with a single exception, implemented in such a way as to be totally independent of the sensing hardware. The single exception is a sensing device-specific component that communicates with higher-level signal processing and presentation components through an interface that is both common and transparent. As an additional feature to those laid out in the objectives for this work, it was indicated that expansion of the toolkit to include other physiologically responsive entities simply requires the utilization through extension of the common interface (which is itself a separate software component) by any new component(s).

We can say categorically then that from Chapters 2 through to Chapter 5 we have fulfilled the objectives laid out for this work. What we need to do now is to find some manner of assessing just how good a job we have made of designing our toolkit. Therefore, identification and completion of a suitable method of assessment is the aim of this penultimate chapter.

6.3 Method of Assessment

The most obvious method of assessing the suitability of the EPICS toolkit for supporting the development of EPIC systems is the practical application of the toolkit to creation of such a system. This can be broken down into three distinct stages:

1. Demonstrate the suitability of the toolkit for prototyping a real EPIC system
2. Demonstrate the suitability of the toolkit for building a real EPIC system
3. Assess the implemented EPICS system in use

This empirical approach to evaluation allows us to assess the ease with which the toolkit components can be composed to create runnable prototype EPIC applications. As the cost of prototyping interactive systems can be high, we want to demonstrate that prototyping EPIC applications using the toolkit components is not only easy, but also rapid. Similarly, after the evaluation of an EPIC prototype is completed, we want to demonstrate that the toolkit components can be used in the final implementation of a system, effectively reducing the overall cost of developing that system. Finally we require someone unrelated to the design and development of the toolkit to use an EPIC application built using the toolkit components. In this way we are establishing that the toolkit components are robust and suitable for use by EPIC system developers.

The following sections describe the toolkit components being used for both the rapid prototyping and implementation of an EPIC application. What follows will hopefully demonstrate not only the utility of the toolkit components in both the design and implementation of a final system, but also how amenable the existing toolkit is to customization and extension.

6.4 Building a Real EPIC Application

The true test of any toolkit is its use in the creation of a real-world application. In the case of the EPICS toolkit the real-world application turned out to be a biofeedback system which was to be used in a controlled psychological investigation into the role of signal presentation in EMG biofeedback (Dyson 1999). The biofeedback system was to form part of an ongoing study by members of the Psychology Department at Lancaster University into strategies of avoiding the onset of migraine headaches.

The initial request by the study psychologist was for a biofeedback application suitable for use in training individuals to reduce tension in the muscles of the frontalis or forehead muscles. The aim of the study was to evaluate whether providing a subject with information about their current level of forehead muscle tension helped them to learn how to relax those muscles more quickly and/ or more deeply than if they had received no feedback information.

This simple description constituted the initial requirements for a suitable EPIC system to support their work. The first step in the development process was to demonstrate the different functional components available in the toolkit for presenting EMG information.

6.4.1 Prototyping Using the EPICS Toolkit

Chapter 4 highlighted the role that user interface toolkits play in supporting the requirements capture process through their use for rapidly prototyping interfaces for evaluation. In order to refine the somewhat vague requirements laid down for the biofeedback system, the toolkit was therefore used to present prototype interface configurations to the study psychologist. A real-time demonstration of the presentation capabilities of the toolkit helped him to decide on modes of presentation most suitable for the study. In order to explicate the ease with which the toolkit can be used for prototyping EPICS interfaces, we will now step through the exact process used in this particular case.

Step 1 - Making the Components Available for Demonstration

Each of the different components that make up the toolkit (described in the previous chapter) are effectively a collection of useful code fragments that together make up some functionally distinct interactive entity. So, for example, the Polygraph Component (see Section 5.4.1.1) is a collection of code fragments that together make up a graphical user interface entity which takes as input a stream of electrophysiological data and whose function is to display in real time changes inherent in that data stream. So the line on the display rises and falls as a direct reflection of the rising and falling amplitude of a particular electrophysiological signal. The configuration of code fragments that makes up the Polygraph Component is shown graphically in Figure 6.1.

The actual polygraphic display is made up of three separate code entities. The first one represents the background panel of the polygraphic display. The second code entity represents the numerical scale (which incidentally is instanced twice in the Polygraph Component – once for the x-axis and once for the y-axis). The third and final code fragment receives physiological information from the DMC (see Section 5.2.1) via a copy of the Generic Object (see Section 5.3.1) and uses this information to animate an embedded graphical display.

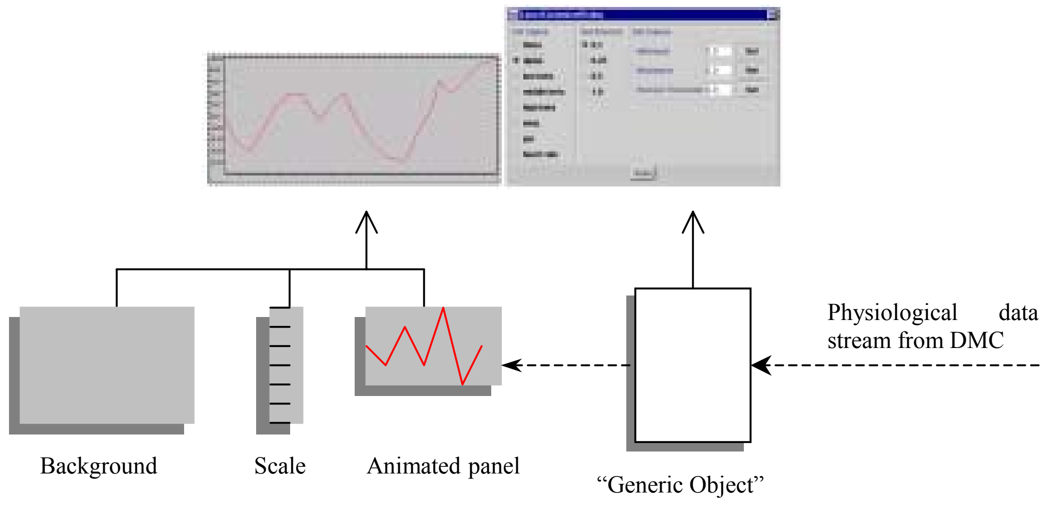


Figure 6.1 Individual parts of the Polygraph Component and it's associated Customizer Dialog

As the toolkit components conform to a particular component technology standard – namely JavaBean technology, it is a simple task to make each composite toolkit component (such as the Polygraph Component) available for evaluation. This is due to the availability of growing number of JavaBean compatible environments, each of which can both display and introspect (see Section 5.6) software components that adhere to the JavaBean standard. Many of these are commercial integrated development environments (IDEs) such as Visual Café, Visual J++ and JBuilder which in themselves provide far more functionality than is required for constructing a prototype EPIC environment. A simpler alternative to making the toolkit components available for evaluation and manipulation is to use a freely available bean box application.

A bean box is a simple container application within which the functionality and appearance of JavaBean components can be tested. The best known of these is Sun's own *Bean Box*, which comes as part of their Bean Developer Kit (BDK). By adding the archived (see Section 5.6) toolkit components to a bean box's archive directory, the toolkit components appear as menu items within the running bean box environment. This can be seen in Figure 6.2.

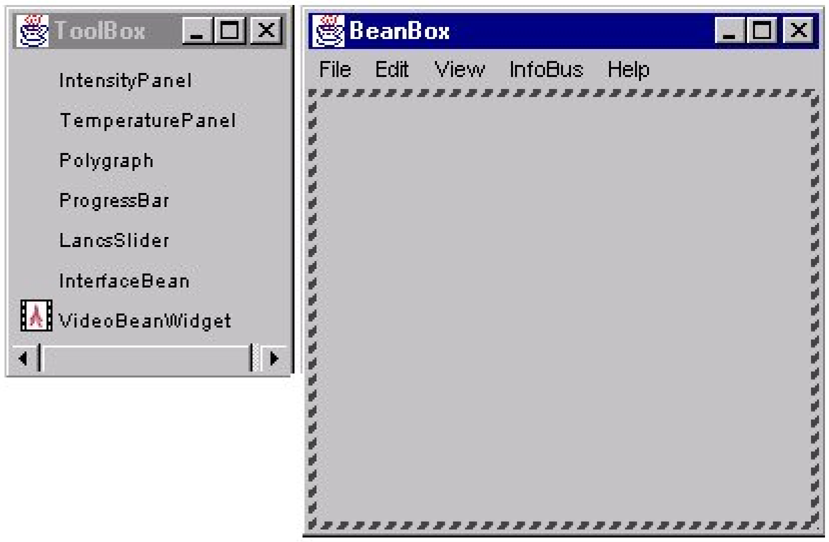


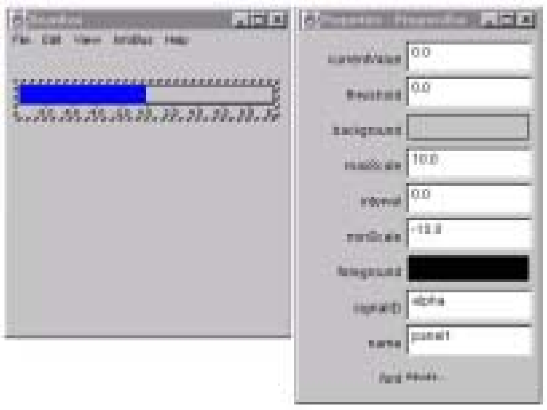
Figure 6.2 Toolkit components available via the menu of a bean box

For the remainder of this section we are going to demonstrate manipulation of the toolkit components using Sun's Bean Box, although all of the activities described here can be carried out in a similar manner in any JavaBean compatible IDE or bean container application.

Step 2 - Demonstrating a Single Toolkit Component

Demonstrating the functionality of a component within a bean box is simply a matter of selecting that component from the menu and dropping it into the main container panel of the bean box. Figure 6.3a shows the Progress Bar Component as it appears within the bean box's container panel. As described above, the bean box introspects the component and makes its properties available for viewing and editing. Generally containers like the bean box display these properties via a menu which appears to the right hand side of the main display panel as shown (Figure 6.3a). However, as our toolkit components each have their own customizer dialog, we can call this up (via a menu option in the bean box environment) to provide an alternative method of both viewing and editing the properties of a component. This is shown in Figure 6.3b.

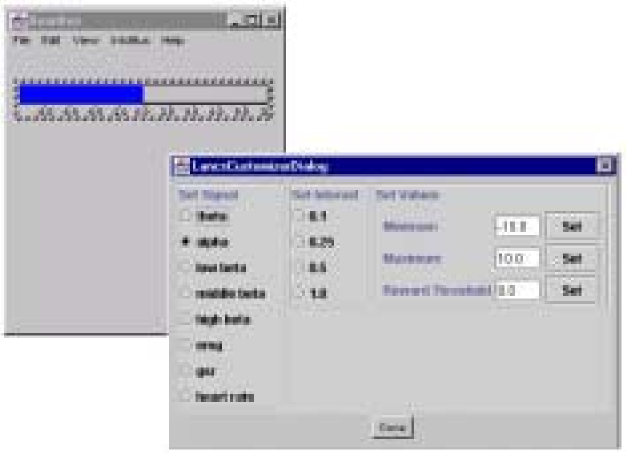
The bean container also provides a menu-based access to a toolkit component's methods. We will say more about these in a moment, but for now it is worth stressing again that the introspection features which enable both the properties and methods of a given component to be made available for manipulation are found in *all* JavaBean compatible environments.



currentValue	0.0
threshold	0.0
background	
maxScale	10.0
interval	0.0
minScale	-15.0
foreground	■
signalID	alpha
name	panel1
font	Abcde...

Figure 6.3a The Progress Bar Component in the bean box

It was stated in Section 5.4 that the initial notification timer interval for any toolkit component that is to receive physiological data is set to a value 0.0, which means *do not notify me*. It is pointless to set this value until that component has registered itself with the DMC to receive data about one or more physiological signals. Therefore the next step in demonstrating the functionality of any such toolkit component is to connect it to the DMC, which in turn connects to the particular physiological sensing device being used.



Set Signal: theta, [✓ alpha], low beta, middle beta, high beta, emg, gsr, heart rate

Set Interval: 0.1, 0.25, 0.5, 1.0

Set Values:

Minimum	-15.0
Maximum	10.0
Reward Threshold	0.0

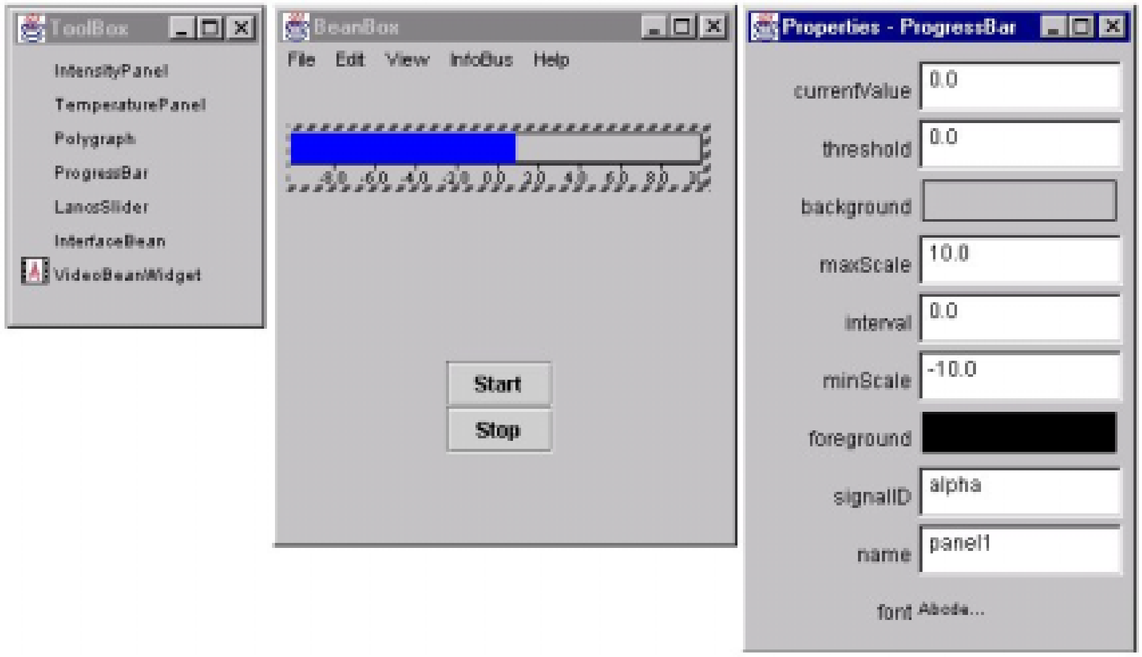
Figure 6.3b The Progress Bar Component's Customizer Dialog

Step 3 – Making a Connection to the Sensing Hardware

Even though the DMC (device management component – see Section 5.2.1) is not something we would expect to find displayed in the interface for a running EPICS application, for demonstration purposes we still need a way to manipulate its major activities. The only activities we need to explicitly manipulate are:

(a) Make a connection to the sensing hardware and
(b) Break the connection with the hardware.

Thus we have created a simple graphical interface to the DMC which consists of two buttons, one marked *Start* to make the connection, and one make *Stop* to break the connection. These are shown in Figure 6.4.



currentValue	0.0
threshold	0.0
background	▭
maxScale	10.0
interval	0.0
minScale	-10.0
foreground	■
signalID	alpha
name	panel1
font	Abcde...

Figure 6.4 The DMC (labelled InterfaceBean in the menu) added to the bean box

At this point we have within our demonstration environment the sensing hardware-specific Device Management Component plus one higher level component (in this case an Interface Level component representing a Progress Bar). The next step is to establish a connection between these two toolkit components.

Step 4 – Establishing a Connection between the DMC and other Toolkit Components

An additional advantage of choosing the JavaBean component standard as the standard within which to develop the toolkit is the ease with which these components can be manipulated. As Java beans can be introspected to make their methods explicit within a graphical environment such as a bean box, it means that connections between components of the toolkit can be established *graphically*. In other words, toolkit components can simply be wired together.

The stages involved in this process are shown in Figures 6.5a, 6.5b and 6.5c. The stage in the process involves selecting the component to be connected to the DMC by clicking on it once with the left mouse button. In Figure 6.5a we have selected the Progress Bar Component. After making this component the focus for our activities, we can go to the bean box environment's *Edit* pull down menu and select a method belonging to the Progress Bar Component. The method we are choosing corresponds to the *addPropertyChangeListener* method which Progress Bar inherits from Generic Object. As described in Section 5.3.1 this Generic Object is instanced by all higher level toolkit components in order to provide the interface between them and the DMC so the description included here of registering the Progress bar with the DMC applies to any of the components described in Chapter 5.

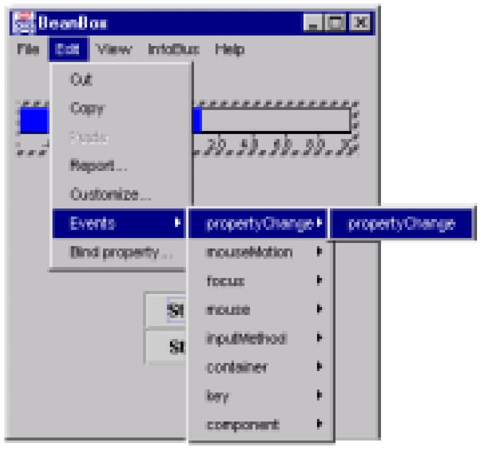


Figure 6.5a Selecting one of the Progress Bar Component's properties

Once selected, this property change method is shown visually within the container environment as a wire (the red line shown in Figure 6.5b) protruding from the Progress Bar Component. The wire can be dragged over another component, which in turn displays a list of methods it has available to be called when the event later fires. So for our purposes, the wire is to be dragged to the graphical representation of the DMC, where the mouse button is release, causing the DMC to automatically display a list of its available methods (see Figure 6.5c).

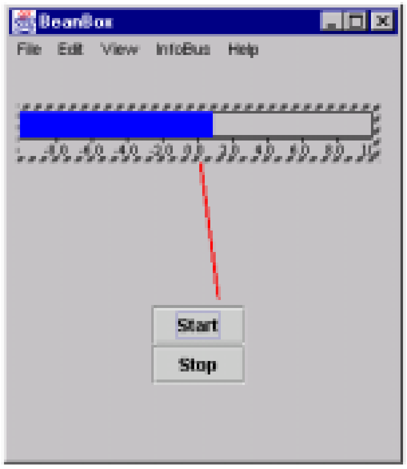


Figure 6.5b Wiring the Progress Bar Component to the Device Management Component

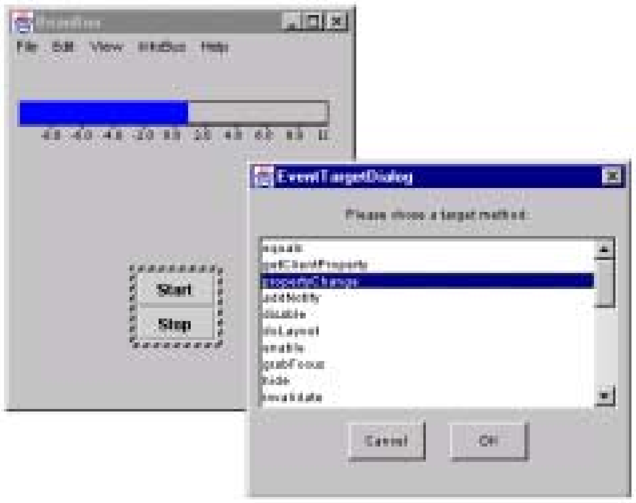


Figure 6.5c Selecting the DMC’s property to be called on generation of a new data point

As stated previously in Chapter 5, the DMC takes this opportunity to register itself automatically with the Progress Bar Component in order to be notified of changes in the bound properties set out in the Generic Object (see Section 5.3.1).

Step 5 – Establishing a Connection between the DMC and the Sensing Hardware

At this stage it is necessary to establish the connection between the DMC and the sensing hardware. This is simply a matter of pressing the Start button with the left-hand mouse button. At this point the sensing device which is connected to the serial port of the computer is switched on though a method call by the DMC to the sensing device's DLL. This is all hidden from the user.

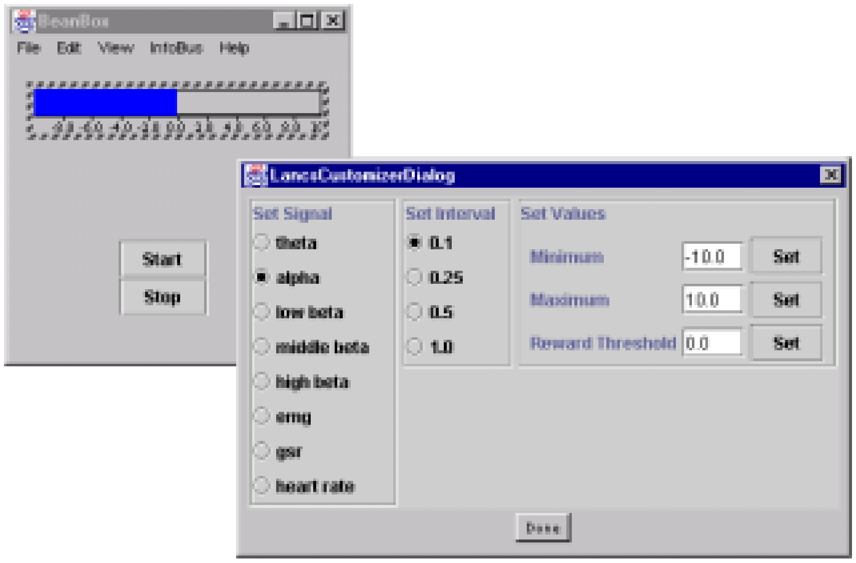
Step 6 – Setting Properties within a High Level Toolkit Component

It was stated previously that the timer notification property of the Progress Bar Component is initialised to 0.0 which means *don't notify me*. Also, the signalID which tells the DMC what electrophysiological signal the Progress Bar Component is interested in is initialised to a null value. This is because at the point of initialisation, and before a higher level toolkit component has registered with the DMC it cannot know what electrophysiological signals are available from the given sensing hardware. At the point of connection between the Progress Bar and the DMC, the DMC makes available a list of available electrophysiological signals. This can now be seen if we open the customizer dialog associated with the Progress Bar by selecting the customizer dialog option from the bean container's *Edit* menu[29] (see Figure 6.6). There are now two things that we must do in order for the DMC to begin polling the sensing hardware for data.

The first thing we can do is select a physiological signal from those displayed in the customizer menu. This selection causes an event to be generated, which the DMC is notified of. The DMC then retrieves from the Progress Bar the identifier of the physiological signal selected within its customizer dialog. Now the DMC can register the Progress Bar with an internal data query object that is associated with that particular signal. At this point the DMC is still not polling the sensing hardware for data as it does not yet have a timer interval at which to carry out the polling activity. So the next thing to do is to select a timer interval from the list provided by the Progress Bar's

[29] Customizer dialogs are a standard optional feature of the JavaBean component specification and are thus supported by all JavaBean containers and IDEs

customizer dialog. Carrying out this activity again generates an event, which the DMC receives and which causes it in turn to request this timer interval from the Progress Bar. With this information the DMC automatically begins to poll the hardware for data about the requested signal at intervals corresponding to the Progress Bar's timer interval. A final note at this stage – a high level component's timer interval and physiological signal identifier can be set in either order but the DMC only begins to pool the hardware when both properties have been set.



Set Signal: ( ) theta [✓ alpha] ( ) low beta ( ) middle beta ( ) high beta ( ) emg ( ) gsr ( ) heart rate

Set Interval: [✓ 0.1] ( ) 0.25 ( ) 0.5 ( ) 1.0

Set Values:
Minimum -10.0 Set
Maximum 10.0 Set
Reward Threshold 0.0 Set

Figure 6.6 Using the Progress Bar's Customizer to set the physiological signal identifier and notification interval.

Step 7 – Adding Further Components to the Container for Demonstration

Steps 1, 3 and 6 should subsequently be repeated for each new high level component that is to be demonstrated. It does not matter that the DMC is now running, but in each case a new component will not receive data from the DMC until its timer interval and physiological signal identifiers have been set through that components particular customizer dialog. Figure 6.7 shows the bean box displaying (clockwise from top) a Progress Bar Component, a PolyGraph Component, a graphical representation of the single, sensing device-specific DMC, a Video Component, a Temperature Component and an Intensity Component. Each of the components in the display may well be receiving information about a different physiological signal and at different timer intervals[30].

[30] See Section 5.2.1.5 for details on how the DMC manages different timer interval requirements

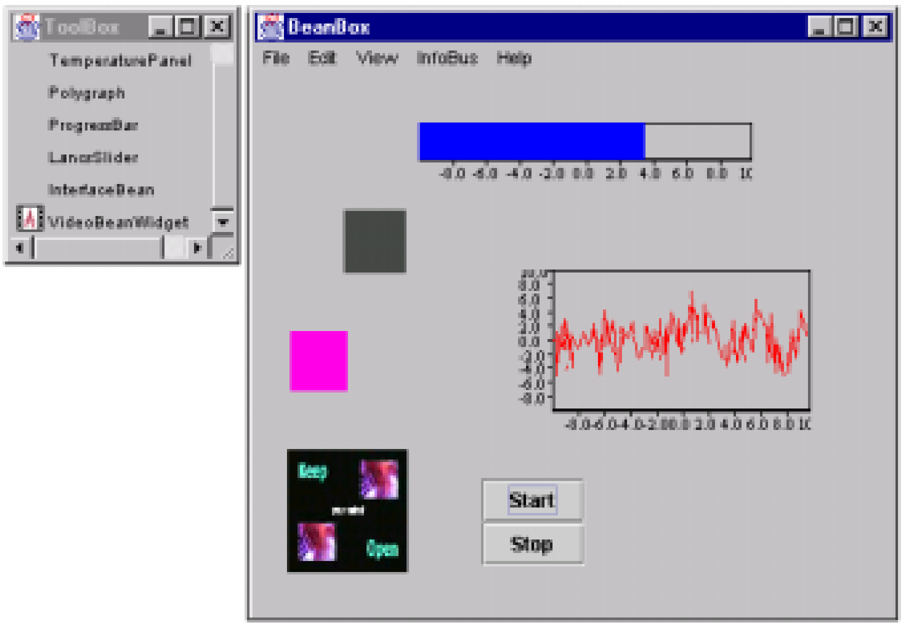


Figure 6.7 A running biofeedback simulation

Using the approach outlined above, a runnable simulation of an EPICS interface was constructed. This could then be evaluated by the study psychologist in order for him to decide on the best methods of signal representation for the EMG evaluation. This prototype consisted of a configuration much like that shown in Figure 6.7, using the bean box and with each component receiving EMG signals at different rates.

6.4.2 Assessing the Prototype

Three particular aspects of the prototyped interface configuration were of interest to the study psychologist. These were:

1. The display options for feeding back EMG signal data to a subject
2. The effect of different timer intervals on the different display options
3. Support for archiving and data analysis

The study psychologist was present as the prototype interface was built. Following the process laid out above it took approximately 5 minutes to construct a functional prototype of a biofeedback interface. The presence of a running simulation fuelled discussion about different potential interface configurations and further requirements for the display were established.

It had already been established that the study would involve three groups of individuals, two receiving feedback information about changes in EMG tension and one control group who would receive no feedback. As already mentioned, the aim of the study was to assess what effects, if any, the availability of physiological signal information had on the process of conscious forehead muscle relaxation.

The first group to be studied would receive no physiological signal feedback information. This group served as a control group for the study and obviously the no feedback condition meant no specific interface presentation requirements. The second group where to receive only aural feedback about the current level of tension in the frontalis muscle group. The psychologist's choice was to present the amplitude of the frontalis EMG signal as a continuous tone, the pitch of which was to increase in relation to an increase in muscle tension, and decrease as muscle tension decreased. This was demonstrated within the prototype using an Audio Component (see Section 5.4.1.4).

Individuals in the final group were to receive visual feedback about the changing amplitude of their forehead EMG. It was important for the sake of the study to ensure that the different feedback conditions, despite employing differing sensory modalities, had as few other variable conditions as possible. After assessing a number of the available visual feedback components, the study psychologist decided to avoid a traditional polygraphical representation or any other representation that provided a scale against which a user could actively measure changes in signal magnitude. He chose instead the more abstract signal representation provided by the VR Interface Component, where EMG signal amplitude would correspond to the height of a levitating block presented in a three-dimensional space (see Figure 5.17). A summary of the characteristics of the two feedback conditions provided for the study is shown below in Table 6.1.

Individuals in all groups were required to have the changing characteristics of their frontalis EMG monitored throughout a session, with the data being archived for later analysis. This required access in all cases to the toolkit's Archiving Component, the functionality of which we were also able to demonstrate using the bean box.

The format for a training session was thus laid out to be:

- A 15-second[31] period where the subject would be asked to frown as hard as possible in order to produce a measure of maximum tension in the forehead muscles. This data was to be recorded and archived. A signal value averaged over the whole 15-second period would be representative of maximal forehead muscle tension.

- A 6-minute period where the subject would be asked to stare at a fixed dot on a screen and try to relax their forehead muscles. This data would again be archived, the data subsequently being averaged of the 6-minute period serving as a measurement of what is known as a *resting baseline*.

- A 10-minute training session. The data from this was also to be archived for later data analysis using an Excel spreadsheet package.

At the start of each session the details of the study would be explained to a subject and the biofeedback system introduced. Subjects would be told that the feedback they were to receive through the system corresponded directly to their current level of forehead muscle tension. It would also be explained that the height of the aural tone or visual block corresponded to the current level of their forehead muscle tension. The higher the tone/ block, the higher the tension, the lower the tone/ block, the lower the tension. The aim for the 10-minute training period was to keep the tone/ block as low as possible.

Information gathered during the resting and frowning periods would be used to calculate two threshold amplitude values. The first threshold value was to correspond to maximum EMG amplitude obtained during frowning. The second was to correspond to the minimum EMG amplitude obtained during the resting baseline period. These thresholds would need be set within the biofeedback system in such as way as to correspond to the highest and lowest possible pitch in the aural feedback condition and to the highest and lowest position in the block feedback condition.

[31] The reason for such a short period is because frowning can quickly become extremely uncomfortable and is difficult to sustain.

	Tonal Feedback	**Visual Feedback**
Surface characteristics of the feedback signal	Disembodied tone	Solid grey block
Lower μV value	50% if mean μV from resting baseline	50% if mean μV from resting baseline
Upper μV value	Mean μV from maximal tension	Mean μV from maximal tension
Sampling interval	Approx. 0.5 seconds	Approx. 0.5 seconds
Signal presentation dependent upon μV change?	No	No
Signal present when current μV below lower threshold value?	Yes	Yes
Signal present when current μV above upper threshold value?	Yes	Yes
Current μV relative to range?	No	Yes

Table 6.1 Characteristics of the alternative feedback conditions (after Dyson 1999)

If, during a feedback session, the value of detected muscle tension fell between these values it would be converted into to a suitable pitch or block position value providing immediate, real-time feedback to the subject. If, on the other hand, a data point were received by the Audio or VR interface components, where the EMG signal amplitude value fell outside of this range, no change in tone or animation was to occur. In this way the subject would receive feedback of one kind (tonal or positional) if they were maintaining their level of forehead muscle tension between the pre-specified threshold values. They would receive feedback of a different type (fixed tone or no animation) if their level of forehead muscle tension fell outside of the threshold values.

Initial explorations into the appropriateness of these measurements were carried out using the mocked-up feedback application employing the VR Interface Component (Section 5.4.1.5) that converted EMG signal magnitude into the changing height of a graphical block. Discussion during this investigative session lead to the conclusion that there existed the potential for reduction in forehead muscle tension to occur below that achieved during the resting baseline measurement period. If this were to occur, using the initial design with the baseline as the lower threshold value, the user would receive the wrong type of feedback (a fixed tone, no animation), as the signal would fall outside of the useable range. For this reason, the study psychologist established that the lower threshold value would need to be set at 50% of the resting baseline measurement (see Table 6.1).

An additional feature for the final system that was established during the evaluation of components running in the bean box was the rate at which the sensing hardware should be sampled for information regarding the level of EMG tension. During the initial demonstration, the sampling rate interval was adjusted to demonstrate the feedback provided through different update rates. With the time interval set at 0.1 seconds, the tonal feedback produced *"…a frantic and unsettling barrage of sound…"* (Dyson 1999, page 45). Similarly with the block feedback, the moving block jumped around erratically as the VR Interface Component tried to satisfy a refresh rate of 10 frames per second. This made it difficult to get a feel for what was happening with the EMG signal. An agreeable update rate for both the tone and block feedback was, after some experimentation, found to be a 0.5-second interval.

6.4.3 Discussion

There are several things that can be established from this description of the toolkit components being used to prototype an EPICS application. These can be categorised as follows:

1. **The existing toolkit components lend themselves to the speedy construction of prototype EPIC systems.** As described above, the creation of a prototype interface for a biofeedback application was constructed in around five minutes[32].

2. **A prototype can be constructed using any one of a number of widely available JavaBean-compatible environments.** Although the description given above concentrates on the use of a particular JavaBean container, namely Sun's Bean Box, it is to be stressed that all development and demonstration environments that support JavaBeans can be used to manupulate Java components in the manner described.

   Access to such tools means that the toolkit components can be manipulated within a drag and drop graphical environment. As described in Chapter 4, toolkits which consist of high-level

[32] By an expert user

code fragments are often only of use to experienced programmers as a means to construct final applications more quickly. Whereas those toolkits which are supported by environments which afford graphical manipulation of the high-level code fragments are more likely to be accessible to users who do not have programming expertise but who have instead expertise of the application domain. Although in the description above, the study psychologist, who was the domain expert in terms of the biofeedback application, did not manipulate the toolkit components himself, with more appropriate (read: more intuitive) JavaBean manipulation environments[33] this would certainly be possible.

3. **The availability of a runnable simulation of an EPIC system fuels discussion between the domain expert and the implementor of the final system, leading to the establishment of further requirements for a final system.** This is after all one of the main reasons for prototyping an application.

Having established the requirements for the actual biofeedback application, the next step was to build that application, utilizing the toolkit components in the final implementation.

6.4.4 Implementing an EPIC Application

Due to the fact that the system was to be used by the study psychologist without technical support or assistance, it was important to package the toolkit components within a familiar and easy to use environment. It was decided that the most appropriate mode of presentation for the biofeedback application was within a windows-style interactive environment. This meant that interaction with components from the toolkit would be eased for anyone familiar with WIMP-based interaction. Also, as stated in Section 3.5.1, the windows-style of presentation lends itself ideally to biofeedback applications which often require the simultaneous presentation of multiple physiological signals.

The core functionality of the application had been decided upon through the prototyping stage described above. Most of this functionality was already available in a number of the toolkit's components. During evaluation of the existing components however, changes had been suggested

[33] The need for more useable JavaBean manipulation environments is discussed in the final chapter of this thesis under the heading of further work.

by the study psychologist as to the total representation of information within the VR Component in particular. This required a small amount of re-coding in order to modify that component for the following reasons.

Firstly, it was mentioned in Chapter 2 that eye movement produces distinct electrical signals known as electro-occulograms or EOG. When trying to measure other electrical activity from either the face or scalp (or as in our case, the forehead) EOG signals appear as electrical noise. Known generally as EOG artefacts, these signals contaminate the other signals being detected. The most common method of eliminating eye movement artefacts is by detecting them explicitly with electrodes positioned around the eyes, and subtracting the EOG signals from the other signals being detected. A simpler alternative is to encourage the subject to restrict their eye movement as much as possible. This second approach was the one favoured by the study psychologist and in order to facilitate this a 'stare' point was built into the feedback system. For the tone feedback condition a sheet of black card with a single central white dot on it was to be used to provide a stare point[34].

In order to provide a similar stare point within the VR Interface component, which became referred to as BlockWorld, the 3D world had to be modified to include an object on which the subject could fixate. In order that the subject could see the movement of the moving feedback block without moving their eyes away from the stare point, it was necessary to position the fixation object proximal to the moving block and roughly half way between its highest and lowest potential positions.

The BlockWorld feedback component uses the two threshold values, corresponding to the maximum and minimum signal values, to automatically calculate the position for the stare point object. The stare point object (a small white cube) and visual representations of both the maximum and minimum signal values (two small pink cubes) are positioned along a vertical pole parallel to the path of the signal feedback block. This is illustrated in Figure 6.8. The addition of visual cues as to the current maximum and minimum threshold values corresponded to the second modification that needed to be made to the original VR Interface Component.

[34] This interface was considered more appropriate that an electronically presented equivalent due to potential of screen flicker. The cardboard interface was also used for the baseline measurement taken at the start of the session.

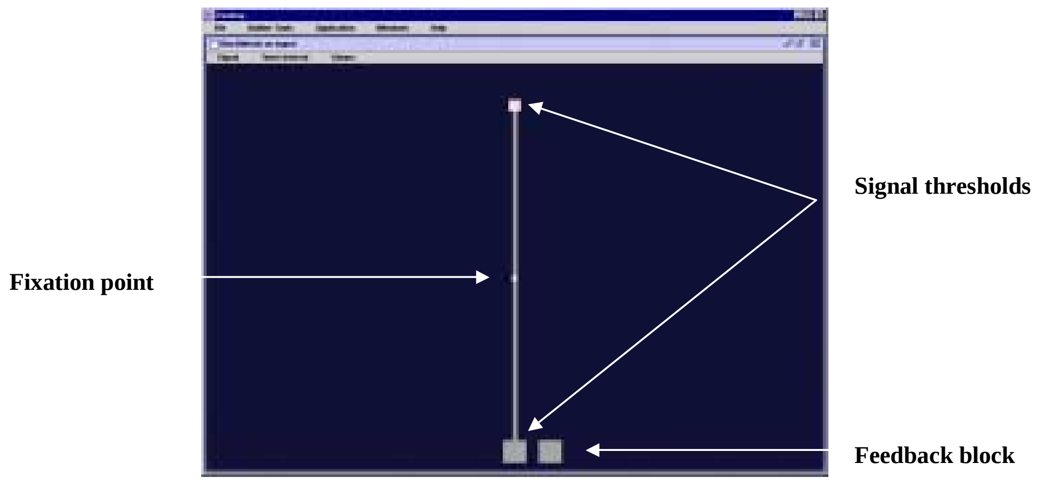


Figure 6.8 The modified VR Interface Component known as BlockWorld

The overall windows-style environment for the final biofeedback application was created using standard Java interface components from the Swing libraries. A frame-like container was created that was suitable for holding instances of any of the toolkit's Interface components[35]. The container component included its own pull down menus enabling the study psychologist to set both the signal identifier for the physiological signal that the contained component was to listen for and the timer interval required between notifications of new signal data points[36]. Once a signal is chosen from the pull down menu, that signal's identifier is displayed in the title bar of the toolkit component's container. This enables the study psychologist to see at a glance which signal a particular interface entity is responding to. An example of the component container containing an instance of the BlockWorld Interface Component (itself a modified version of the VR Interface Component from the toolkit) and presented within the windows environment built for the study is shown in Figure 6.9.

The DMC became an integral part of the overall windows-style environment being automatically initialised when the application is first started. Whenever a new Interface Component, such as BlockWorld, is selected from the application's pull down menu and added to the windows-style environment it is automatically registered with the embedded DMC. This registration is analogous to the wiring together of Interface Component and DMC described in Step 4 of the

[35] A method in the container class takes as its parameter an instance of the generic object class (which all components extend).
[36] These are features previously demonstrated in the customizer dialog used for manipulating toolkit component properties in a demonstration environment, such as in the bean box.

prototyping scenario outlined above. When a signal identifier has been chosen for an Interface Component, that component is automatically registered with the signal-specific data query object within the DMC. At any point after the windows-style application is opened, the DMC-sensing hardware connection can be made by choosing the Start option from the Application menu (top left in Figure 6.9). As soon as a signal identifier and timer interval have been selected for the BlockWorld or Audio Component, the DMC begins to poll the sensing hardware box for data.

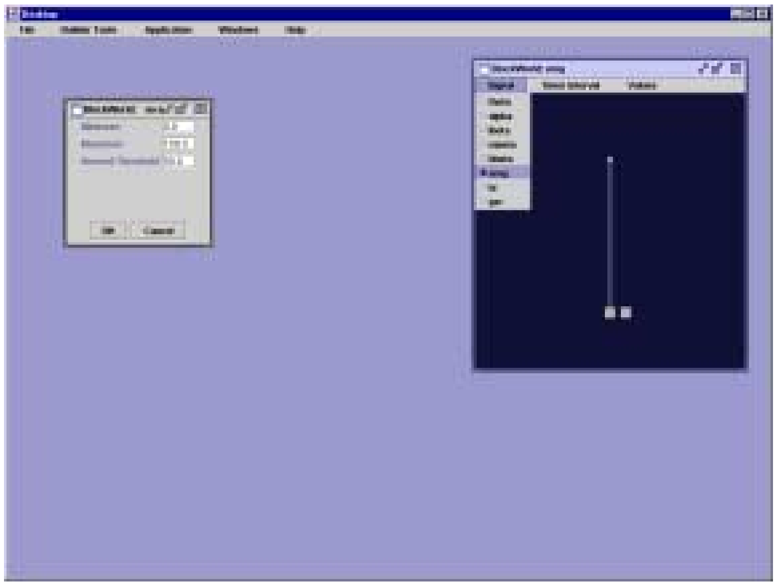


Figure 6.9 The BlockWorld feedback component presented within a frame-like container and presented within a Windows-style application

After the two baseline measures where taken in the manner described in Section 6.4.2, the study psychologist would require a means to input the calculated threshold values. As these values correspond to the maxScale and minScale variables found in the Generic Object (see Section 5.3.1) all that was required was to make the setting of these values available within the windows environment. In order to achieve this, a frame was created through which the variables could be set textually, in a manner similar to that employed in the customizer dialog described in the previous chapter. This frame, shown on the left side of figure 6.9, opens automatically when an interface component is first selected from the main menu's Builder Tools menu.

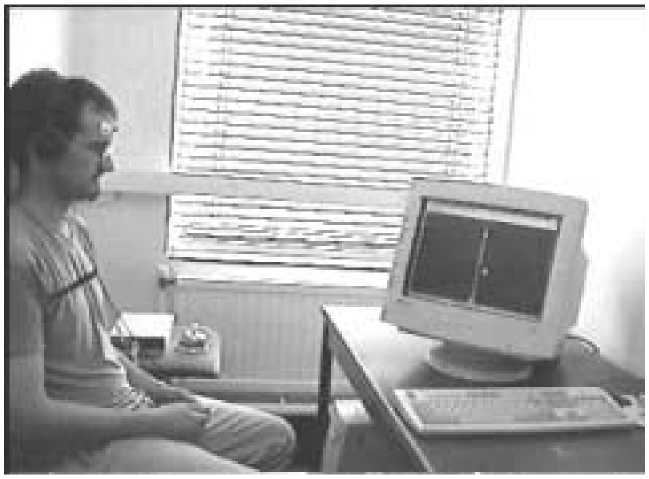


Figure 6.10 A subject undergoing EMG biofeedback training using the BlockWorld feedback component

The other feature required in the final application was the ability to archive data for later analysis. This was provided using an unmodified instance of the toolkit's Archiving Component. Having no specific interface of its own, a dialog was created which enabled the study psychologist to select the signal ID for the physiological signal of interest, and to set a name for the file into which the archived data was to be saved. For analysis purposes, the study psychologist required that the timer interval for receipt of new data points corresponding to current EMG magnitude was to be set to 0.125s. As the Archive Component and the feedback presentation components (BlockWorld and Audio) each retrieve data independently from the DMC, the different data requirements of these components was easily satisfied. Figure 6.10 shows the BlockWorld feedback component in use. It is shown here maximized to fill the screen during biofeedback training in order to lessen distraction for the subject undergoing training.

6.5 Assessing the Application

The final biofeedback application described above was used in controlled study to investigate the effects of EMG signal presentation in the learning of forehead muscle relaxation. The study involved 18 subjects, 6 of whom received real-time EMG signal feedback aurally thorough the toolkit's Audio Component, and 6 of whom received visual signal feedback via the BlockWorld interface[37]. The study continued over a period of 3 weeks during which each subject participated in 5 identical feedback training session using the biofeedback system.

During a period of biofeedback training using the system, subjects were left alone, wired to the system[38] whilst they tried to use the feedback information provided by the biofeedback system in order to learn to control the level of tension in the frontalis muscles of the forehead. No subject experienced any difficulties either understanding the point of the biofeedback system or using the system.

Throughout the period of the study the psychologist in charge of the investigation used the biofeedback application unsupervised and without technical assistance. Data was collected from all subjects taking part in the study using the biofeedback application. This data was archived by the biofeedback application and analysed off-line using a standard spreadsheet package.

At the end of the study the main psychologist involved indicated his satisfaction with the system. He liked in particular the flexibility in signal presentation, which the toolkit afforded. He was especially pleased with the ability to present signal feedback within a 3D virtual environment, but commented more generally as the requirement for flexibility in signal presentation. Another proponent of biofeedback (Gaudett 1983) acknowledges that individuals have definite preferences for the ways in which physiological information is presented which may lead to differences in biofeedback training outcomes. This is a subject still awaiting investigation however.

6.6 Discussion

The aim of this chapter has been to evaluate in some way the ability of the EPICS toolkit to fulfil its role in supporting the development of future EPIC systems. In order to do this, we have followed the process of prototyping and building a fully working EPIC application. In order to establish the utility of the EPICS toolkit is worth reiterating (and in some cases expanding upon) a number of points from what has been described above.

[37] The other 6 participants were a control group who received no feedback information during EMG relaxation training
[38] Via the WaveRider physiological sensing hardware, which was the device from which the DMC within the biofeedback application was retrieving EMG data

As support for manipulation of the toolkit components within drag and drop environments has been established as being widely available, there is no reason why this task could not have been carried out by anyone who is familiar with the toolkit components. Although the task of prototyping the biofeedback interface was carried out by the creator of the toolkit, as is generally the case with the support for graphical manipulation of high level code entities, there is less of a requirement for the users of such environments to have extensive coding experience.

The availability of a runnable prototype fuelled discussion as to overall requirements for a final biofeedback application. The prototyping process was carried out in the presence of an expert in the domain of biofeedback training. The domain expert was able to evaluate both the appearance and behaviour of various signal presentation components. This lead to the direct selection of some toolkit components to be included in the final system, as well as suggested improvements to other components to be included also.

Implementation of a fully operation, windows-style biofeedback system took roughly two working days to complete. Subsequent to the prototyping session, the study psychologist came back with further details as to how the overall biofeedback-based study was to be carried out. This information was also fed into the design of the final system. The implementation period included modifications to existing toolkit components. It should be noted however that implementation of a fully blown EPIC system was a coding task that necessarily had to be undertaken by an experienced programmer.

The choice of standard component technology increased the ease of integration of the toolkit components with other pre-existing high-level code libraries. The easiest way to manipulate both the toolkit components and the traditional WIMP interaction techniques, such as pull down menus, that are provided by the Java AWT and Swing libraries is by importing those libraries and the toolkit components into a commercial integrated development environment. The final application was constructed using Symantec’s Visual Café development environment. Implementation, including the described modifications to the VR Component, took approximately two days to complete.

The final system was used in a controlled study of the role of signal presentation in EMG relaxation carried out by the Psychology Department at Lancaster University. They used the system over a 3-week period, in isolation and without the need for technical support from the system implementor.

The overall conclusion we can draw from the description of the prototyping, implementation and use of an EPIC system described in this chapter is that the existing EPICS toolkit components are suitable for supporting the creation of robust EPIC applications.

6.7 Summary

This chapter began by revisiting of the overall objectives for this thesis laid out in the introductory chapter. It was established that Chapters 2 through to Chapter 5 have fulfilled this thesis's three main objectives. The aim of this particular chapter has been an empirical evaluation of the EPICS toolkit.

Firstly, we described the EPICS toolkit being used for rapidly prototyping a popular form of EPIC application. Using the toolkit components the act of constructing a runnable simulation of a biofeedback interface was seen to take a desirably short period of time[39]. Due to the implemented toolkit components adhering to a standard component technology, namely JavaBeans, the possibility was discussed for the act of prototyping an EPIC application to be carried out using any one of a growing range of JavaBean compatible demonstration and development environments. In this instance, the act of prototyping the biofeedback interface was described using Sun's freely available Bean Box environment.

The prototype was constructed in the presence an expert in the domain of biofeedback training. The presence of the prototype served to demonstrate to the domain expert the signal presentation capabilities of the existing toolkit components. The aim of the evaluation was to establish requirements for a final biofeedback application that was to be used in a controlled psychophysiological study. The process of evaluation using the running biofeedback simulation stimulated discussion between the domain expert and the implementor of the final system, during which requirements for the final system were established.

[39] 5 minutes for an experienced user who was familiar with both the toolkit components and the construction environment

Subsequent to the prototyping experience, the final windows-style interactive biofeedback application was implemented in a period of roughly two working days. The final implementation included a number of standard EPIC toolkit components plus one modified EPIC toolkit component, as well as standard interactive components from the Java and Swing libraries. These were brought together using a commercially-available integrated development environment.

The final application was used in a controlled investigation into the role of signal presentation in EMG biofeedback. The system was used by the study's psychologist to train 18 individuals in order to gain conscious control of EMG activity. The system not only enabled the presentation of EMG signal information in a couple of abstract feedback formats, but it also archived each subject's EMG signal data for later analysis. Over the period during which the evaluation took place, the system was operated by the study psychologist with no requirement for technical intervention or support.

The overall conclusion that we can draw from this description of the existing EPIC toolkit being used to rapidly prototype and then construct a biofeedback application is that the toolkit fulfils the requirement to support the development process for future EPIC systems.

Chapter 7

Conclusions and Future Work

7.1 Summary

This thesis has presented a toolkit that supports the rapid prototyping and implementation of systems that incorporate human electrophysiological information. The toolkit takes the form of a suite of Java components that conform to the JavaBean standard.

Chapter 2 offered an introduction to the range of human physiology that can be detected through the use of special electronic sensing equipment. It provides a generic overview of the principle features of sensing technologies and describes how integrated sensing devices can be used to make physiological signals available to a computer. Also in Chapter 2 the principle of biofeedback was introduced as a mechanism by which to train individuals to learn conscious control of various physiological parameters though the real time presentation of signals relating to those parameters.

In order to ascertain what common interaction techniques would be desirable in a development support toolkit, a review of systems that incorporate electrophysiological information was presented in Chapter 3. This review brought together work from a number of fields including clinical biofeedback, prosthetic technologies for the disabled, research into hands-free human-machine control, psychophysiological task analysis and the development of emotionally responsive technologies. These systems where identified collectively as being EPICS – electrophysiologically focussed interactive computer systems – due to their shared requirements for detecting, processing and presenting physiological information. Chapter 3 concluded by identifying the shared features among these systems that could be provided by suitable support tools.

Chapter 4 examined the support that has emerged for the development of more traditionally interactive systems, looking in particular at the range of tools available for rapid prototyping, design and development of systems based on the WIMP interaction paradigm. It was demonstrated that the existence of conceptual models of interaction aids tool design by providing, in the first instance, a set of high level components. These components can be used for thinking about the communication that takes place between a user and an application and how this can best be facilitated through a suitable user interface. The second benefit of conceptual models is that they demonstrate well-defined architectures, which can be reflected in the tools conceived after them. Finally, it was shown that the success of interactive systems development depends on the input of a range of experts, many of whom are not software developers. For this reason a necessary consideration in the design of tools is their usability by non-programmers.

Chapter 4 also considered the development of non-WIMP interactive systems and looked in particular at VR applications, as these share with EPICS features of both multimodality and the requirement to sense and make sense of continuous streams of user generated input data. A conceptual model of interaction based on sensing was identified and modified to suit the wide application base of EPIC systems identified in Chapter 3. This modification was based on the requirement for the continuous data streams produced by physiological sensing hardware to be pre-processed in two distinct stages in order to satisfy the differing data requirements of various types of EPIC system.

Chapter 5 described the design and implementation of a suite of Java components whose design and conception reflect the architecture of the refined sensing model of interaction identified in Chapter 4. The pre-processing of physiological signals takes place across a number of Java components, which in turn can be combined in different ways to suit the requirements of different applications. The choice of components for implementation in the interface set is based on methods of physiological signal presentation identified in Chapter 3 as being common across a number of different types of EPIC system.

A description of the Java component suite in use provided the focus for Chapter 6. The choice of JavaBean software components as the basis for developing the toolkit was justified by demonstration of the ease with which these precompiled entities can be composed using freely available builder tools such as Sun's Bean Box. Chapter 6 also described the use of the toolkit components in both the design and rapid implementation of a biofeedback application that was used in a study carried out by the Psychology Department at Lancaster University.

This final chapter discusses the implications of the work described in this thesis. It begins by reviewing the aims and objectives that were laid out for this work in Chapter 1, stating in turn how each of these has been satisfied. A summary of the contributions of this work is followed by a description of future improvements and extensions to the existing toolkit. This chapter closes with a discussion of the implications of the work presented in this thesis.

7.2 Review of Project Aims

The overall aim of the work presented in this thesis has been to develop a software toolkit to support the creation of interactive systems that incorporate electrophysiological information. In Chapter 1 this overall aim was broken down into a number of distinct objectives, which will be reiterated (in italics) here.

Identify the requirements for building interactive systems that incorporate physiological sensing technologies.

By providing a broad, comprehensive look at what are in terms of human-machine interaction unconventional systems, it has been possible to classify those systems into two broad groupings:

1. Systems that detect and feedback physiological information to a subject for training purposes.
2. Systems that detect and process physiological information in order to monitor the physiological and emotional state of system users.

Emergent requirements for training systems focussed on the detection of signals and their real time presentation to a user. This constitutes the setting up of a feedback loop between the system and the user, which enables the user to see the effects that different thoughts, feeling and/or situations have of a particular physiological parameter, such as heart rate, blood pressure, even signals detected from the brain.

Monitoring systems also required signal detection, although feedback information was of less importance. Traditional polygraphical signal representation is useful to the interested parties in monitoring physiological signals (psychophysiologists in the main) as they convey information about both the magnitudinal and temporal aspects of these signals. The major provision in monitoring systems however was for capturing information for later assessment.

In more complex monitoring systems, such as those being used to investigate the potential for direct brain-computer communication, a complex level of pre-processing was required and this was shown to be most often provided by neural networks and other forms of learning algorithm.

Together, these sensing, processing and presentation requirements served as the basis for the design of the pre-processing and interface components implemented in the final toolkit.

Identify a suitable model of interaction for systems based on sensing

Chapter 4 demonstrated how existing software development toolkits reflect high level models of interaction. Identification of a suitable model of interaction for sensing-based systems was shown to depend upon reconsideration of the physical relationship that exists between a computer and its user. By thinking beyond mechanical interaction, it becomes possible to think about the requirements of systems whose job it is to sense the user's actions. A non-mechanical human-machine coupling model, called the Structural Coupling Paradigm, was identified as providing the ideal basis for thinking about and designing systems that support human-machine interaction through sensing.

Based on satisfaction of the first two objectives, design and implement software toolkit suitable for creating systems that incorporate human physiological signals.

Subsequent to the identification in Chapter 3 of a set of useful EPICS functionality and armed with a suitable conceptual model of interaction, it was possible to design a set of components that matched the conceptual model and provided the required EPICS functionality. These components were implemented as JavaBeans, the details of which were presented in chapter 5

Implement the toolkit in such a manner as to facilitate easy substitution of one sensing hardware device for another.

By subtly modifying the conceptual model and dividing the pre-processing of signals into two distinct stages, it was possible to limit the amount of device dependent code to one[40] component that together provide the first level of pre-processing. All other components in the toolkit are implemented in a manner that provides an abstraction over the physical details of the sensing device used. This means that all other toolkit components are suitable for use with different sensing hardware devices, it has not been able to demonstrate this as part of this study due to the unavailability of another physiological sensing device. However, the structuring of the toolkit components satisfy the criteria for the toolkit to be device independent.

[40] Conceptually one but physically two in the case of the existing toolkit

7.3 Summary of Contributions

The work in this thesis has introduced to computer science a new classification of interactive computer systems one based on the detection, processing and presentation of electrophysiological information. It has done this in the following ways:

By providing a review of existing systems that include physiological signal detection, processing and presentation.

The review in Chapter 3 is, to the author's knowledge, the first occasion on which this diverse range of interactive systems has been brought together and classified according to the shared requirement to sense human physiology. Indeed, during the course of this project the author had cause to talk with many practising clinicians who apply biofeedback methods with their patients daily, yet had never thought about the wider implications of the technologies they used. This was also found to be true of manufacturers of some of the sensing devices who were contacted as part of this work. Electrophysiologically focussed interactive computer systems or EPICS as we have named them constitute a new class of interactive system.

By producing the first software toolkit that addresses the needs of both designers and developers of electrophysiologically focussed interactive systems.

This toolkit is, to the author's knowledge, the first available support for the development of this new class of interactive system. By making this work and the ideas associated with it available within the wider computing community (Allanson 1999b), it is hoped that more people will take up the challenge of exploring of this technology. It is proposed that the EPICS toolkit provides an ideal jumping off point for anyone interested in exploring the potential of incorporating physiological information into the human-machine relationship.

Where it would have been perfectly possible to develop the toolkit in any one of a number of suitable high level programming languages, the choice of Java, and more specifically the JavaBean software component standard was deliberate for a number of reasons. Firstly, Java is an increasingly popular choice for the designers of interactive systems as it offers all of the common interaction techniques as classes within its AWT and SWING libraries. This means that the

EPICS toolkit components can be easily integrated with interaction techniques for standard WIMP-style interactive systems.

Furthermore, from a developer's perspective, Java is well supported in terms of the range of commercially available graphical tools, like JBuilder and Visual Café, which provide management and support for development of complex Java applications. The toolkit components can be imported into any of these environments and can thereafter be modified and incorporated into larger applications. In the same manner, new components can be created using the interface class *Generic Object*. This supports the requirement for extensibility laid out in Chapter 1.

Undoubtedly the most important reason for choosing to implement the toolkit as Java beans is that software components, such as beans, support the notion of *composition not coding*. This means that the toolkit components are precompiled units, which can be composed into a runnable application using any one of a number of freely available bean builder tools. This is ideal for non-programmers, who can construct a running application without the need to understand the working of the underlying system, only needing to know the rules of composition. This last point is elaborated on in the section regarding future work, below.

Identification of a high-level interaction paradigm that serves a wide range of sensing-based interaction and modelling of same using component-based approach

Chapter 4 illustrated how existing conceptual models such as Seeheim, MVC and PAC have influenced the design of tools to support the development of WIMP-style interactive applications. This work has served to identify the high-level model that best describes the interaction taking place in systems that incorporate sensing. Other interactive systems whose input modalities rely on sensing, such as VR systems which incorporate gesture recognition or voice recognition, already apply the principles of pre-processing signals in order to extract features which can be translated in to commands (this was illustrated in the architecture diagram for MR Toolkit shown in Figure 4.10). What the design of the EPICS toolkit demonstrates is that signal pre-processing can be packaged in such a way as to enable the incremental structuring of a number of layers of signal pre-processing functionality to produce a solution suitable to a given application. It is envisaged that the availability of suitably packaged complex processing units, such as neural network classifiers or hand-gesture recognition software (or in fact any type of gesture recognition technology) will enable relatively simple composition-based construction of modally complex interactive systems.

7.4 Future Work

It was stated in the introduction to Chapter 5 that the suite of components presented did not offer a comprehensive set of components for building EPICS applications. Indeed, it is not possible to identify all of the components that may be required by future EPIC systems. Nor is it practical to attempt to construct components to match all of the functional parts of all currently available EPICS systems. All that can be achieved within the scope of a piece of work of this nature is the proposal of a model and implementation of a sufficient number of components to demonstrate and evaluate the validity of that initially proposed model. This requirement has been fulfilled by the components described in Chapter 5.

Nevertheless, there are features that have not been implemented that will be invaluable for the further investigation of a number of aspects of electrophysiological human-machine interaction. These correspond to the 2nd level of pre-processing as described in chapter 5.

7.4.1 Extend the Pre-processing Facilities

A generic view of McMillan's (McMillan 1995) Structural Coupling Paradigm is presented in Figure 7.1, with the interior of the controller exposed as consisting of 3 stages – *sense*, *think* and *act*. The second layer pre-processing component described in Chapter 5, namely the *switching* component, provides the simplest possible form of the "thinking" stage prescribed by the model, about which McMillan writes:

> "*…thinking may involve the recognition and classification of syntactically formed patterns that require more complex processing [than that required for issuing commands on the basis of simple signal characteristics]. Even more sophisticated thinking may require semantic-based interpretation and symbolic reasoning to create the controller's internal knowledge in real-time, perhaps through the use of internal models*." (page 764)

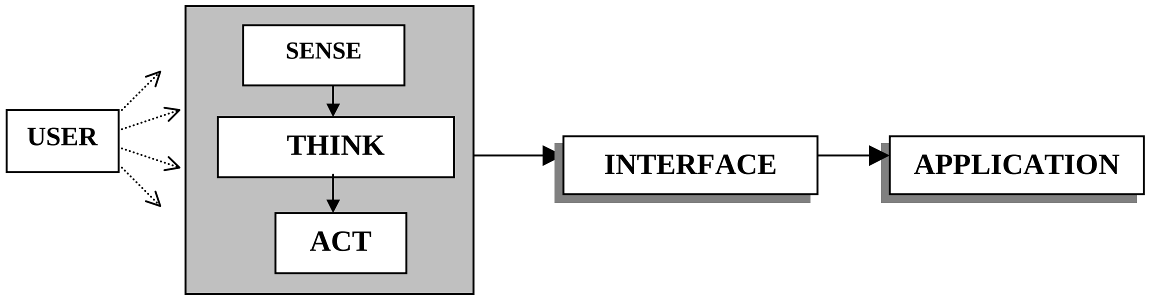


Figure 7.1 Generic model of the Structural Coupling Paradigm

McMillan speculates that ultimately the controller in sensing-based systems will take the form of a software agent working with the user that is familiar with their user's voice/ gestures/ physiological responses. Creating such technology is work for another project.

7.4.2 Embed the Toolkit Components in Applications for Mobile Devices

With the emergence of increasingly powerful and affordable mobile computing devices, it is may be useful to embed the signal pre-processing functionality within a mobile device that is worn about the person of a user. Mobile sensing technologies have been developed for athletes and can detect and assess information gathered during training. With the toolkit components running on suitable mobile devices it would be possible to have the mobile sensing hardware pass data to a mobile device that pre-processes it then transmits the data onto an third party (perhaps the computer of the athlete's coach).

One further reason for running the pre-processing software on a mobile device is that it would enable controllable physiological signals to be used to drive any one of a range of prosthetic devices, including wheelchairs, soft keyboards and limbs. Work that has been undertaken in this area has either involved rudimentary control of devices with embedded sensing and switching technology, for example prosthetic arms, or more complex control, such as brain signal-driven soft keyboards which require proximity to a desktop or laptop machine.

In recent years, researchers at University College, London (UCL) have been experimenting with implanted electrodes in the leg muscles of paraplegic patients (Hunt 1997). By passing electrical signals into these electrodes, leg muscles can be stimulated to move. They have been experimenting with walking algorithms, which attempt to stimulate muscles in the right order to effect walking-like motion in the legs. Though this is incredibly exciting work, the control of motion resides within a computer algorithm over which the subject has only rudimentary control (on or off). It may be possible to use information detected from other parts of the body (ideally, this would eventually be directly from the brain) in order to control the manner in which the leg muscles are stimulated, giving control back to the subject. This may sound like science fiction, but the availability of cheap, portable, intelligent physiological sensing devices means that the work that needs to be done to establish the feasibility of this (i.e. much research into conscious physiological control) could proceed far more quickly.

7.4.3 Produce Application-specific JavaBean Assemblers for Different Domains

Sun's Bean Box is a very simple, but generally useful example of the type of component construction tool that would be useful for non-programmers. In order to support fully the requirements of this group of users, application-specific construction environments are required. Take, as an example, the BioGraph software that runs on the ProComp device (both of which were described in Chapter 2). Biofeedback as a discipline supports a *prototype as final interface* strategy to user interface development. That is, for a given biofeedback session with a particular patient a graphical interface containing multimedia representations of one or more physiological signals of interest is constructed by selecting those multimedia entities from a menu and setting the attributes via a dialogue. Through provision of a suitable bean builder, the toolkit components themselves could be the multimedia entities that the user selects from a menu and whose attributes are manipulated. This would preclude the requirement, demonstrated in Chapter 6, of having to explicitly package the components in a manner as to make them usable by a non-programmer who is comfortable with windowing applications.

In a similar manner, a specific builder could be provided that included only that functionality required by psychophysiologists interested in monitoring the physiological condition of a user for either health and safety reasons, or as part of the systems evaluation process. Here again, we are not talking about using the components to design a system that would be useful for monitoring applications, only to re-implement that application incorporating the toolkit components within a more common windows-style interactive interface. Rather, the idea is to throw away the need to create an explicit windows-style application for a particular domain, and instead create application-specific bean builders to provide the actual interface to a running system which is custom built by the user for each new session of monitoring, or biofeedback training, and so on.

The use of standard JavaBean technology for the development of the components toolkit presented in this thesis means that the toolkit is immediately interoperable with other Java beans, including all of the elements of the SWING foundation classes for Java. As demonstrated through the description of the biofeedback application provided in Chapter 6, this means that traditional WIMP-style interactive systems can be built which have embedded physiological sensing capabilities. This implies that systems such as NASA’s Closed Biocybernetic System, which detect and dynamically respond to changing user state, could be constructed easily and cheaply at this time.

7.5 Implications of this Work

The research contained in this thesis shows that our technological and physiological knowledge has reached a point where it is realistic to envisage mobile integrated EPIC devices being commercially available within the next 3 to 5 years. Initially these may be functionally quite simple, involving the monitoring of one or two physiological parameters in individuals with known medical conditions. This information will be used to help them manage their conditions, perhaps warning them of the potential onset of particular episodes such as epileptic seizures or heart attacks. There is every possibility that in medical EPICS part the system will be physically implanted into the user.

Developments in computer cognition will thereafter enable the development of the types of intelligent and responsive systems introduced in Chapter 3. These will be the first truly *personal* computers maintaining a physiological link to their users, watching and learning their responses to different situations, and perhaps using this knowledge in turn to affect the users’ environment. During the current epoch in human-computer interactivity the major research focus is on the notion of *ubiquity*. EPIC systems are inherently ubiquitous, ultimately blending into the environment and blurring the lines between user and machine.

There is much work to be done in order to understand the implications of our bodies’ physiological responses to their environment. Sensing technologies such as those described in

Chapters 2 and 3 are allowing the scrutinization of this information and thus we have the opportunity to adjust our environments in order to lessen their detrimental emotional and physiological effects. This is a reflective process however, in that by applying these same technologies in order to examine how we respond to our environment, it may be possible to gain control over those responses. By these opposite yet complementary methods, a marriage between humans and suitably enabled computing devices may result in more calming, intuitive and supportive computer-based technologies.

Realisation of electrophysiologically-aware systems depends on more researchers taking on the challenges of electrophysiologically interactive computer systems development. Hopefully, the work contained in this thesis will serve to convince the reader that rudimentary EPICS are feasible now and that working to evolve these opens the door to many exciting possibilities for the future of human-machine interaction.

References

(Aasman 1987) Aasman, J., Mulder, G., Mulder, L.J.M (1987). “Operator effort and the measurement of heart-rate variability”, *Human Factors*, **29**: pp 161-170.

(Allanson 1999a) Allanson, J. (1999). “Mind Over Virtual Matter: Using Virtual Environments for Neurofeedback Training”, *Proceedings of IEEE VR99*, Houston, Texas: pp 270-274.

(Allanson 1999b) Allanson, J., Rodden T., Mariani, J (1999). “A Toolkit for Exploring Electrophysiological Human-Computer Interaction”, *Proceedings of Human-Computer Interaction (Interact ‘99)*, Edinburgh: pp 231-237

(Ames 1996) Ames, A. L., Nadeau, D. R., Moreland, J. L (1996). *VRML 2.0 Sourcebook*, John Wiley & Sons.

(Apple 1998) Apple Computer, Inc. (1998). *HyperCard Script Language Guide: The Hypertalk Language*, Addison-Wesley.

(Archer 1999) Archer, D. (1999). “I Know How You Feel”, *New Scientist*, **162:** pp 26-31.

(Ark 1999) Ark, W., Dryer, D. C., Lu, D. J. (1999). “The Emotion Mouse”, *8th International Conference of Human Computer Interaction*, Munich, Germany, pp 818-823

(Baecker 1995) Baecker, R. M., Grudin, J., Buxton, W. A. S., Greenberg, S. (1995). “Development Tools”, *Readings in Human-Computer Interaction: Toward the Year 2000*. Eds. R. M. Baecker, J. Grudin, W. A. S. Buxton, S. Greenberg. Morgan Kaufmann Publishers.

(Baehr 1997) Baehr, E., Rosenfeld, J.P., Baehr. R. (1997). “The Clinical Use of An Alpha Asymmetry Protocol in the Neurofeedback Treatment of Depression: Two Case Studies”, *Journal of Neurotherapy* **2**(3): pp 10-23.

(Basmajian 1977) Basmajian, J. V. (1977). “Learned control of single motor units” *Biofeedback: Theory and Reseach*. G. E. Schwartz, Beatty, J. New York, Academic Press.

(BAT 1998) Brain Actuated Technologies (1998) “BAT Inc - Makers of Alternative Human-Computer Interfaces”, http://www.brainfingers.com/

(Bell 1979) Bell, J. S. (1979). “The use of EEG theta biofeedback in the treatment of a patient with sleep-onset insomnia”, *Biofeedback and Self-Regulation* **4**(3): pp 229-236.

(Bennett 1981) Bennett, W. A. (1981). “History of Amputation Surgery and Prosthetics”, *Atlas of Limb Prosthetics: Surgical and Prosthetic Principles*. American Academy of Orthopedic Surgeons. C.V. Mosby Company, St Louis, MO

(Bentley 1994)	Bentley, R. M. (1994). "Supporting multi-user interface development for cooperative systems". *Ph.D Thesis*. Computing Department, Lancaster University, UK.
(Bergman 1995)	Bergman, E., Johnson, E. (1995). "Toward Accessible Human-Computer Interaction", *Advances in Human-Computer Interaction*. Ed. J. Nielsen. Norwood, NJ, Ablex Publishing Corporation**:** pp 87-113.
(Birk 1972)	Birk, L. (1972). *Biofeedback: Behavioural Medicine*. London, Brune & Stratton.
(Blattner 1996)	Blattner, M. (1996). "Multimedia Interfaces: Designing for Diversity", *Multimedia Systems and Techniques*. Ed. Furht, B. Boston, Kluwer Academic Press**:** 87-122.
(blaxxun 1999)	blaxxun (1999). "Internet Multimedia Communication" http://www.blaxxun.com/
(Boothroyd 1997)	Boothroyd, D. (1997). "Mind Over Matter" *New Electronics on Campus***:** pp 6-8.
(Bowman 1997)	Bowman, T. (1997). "VR Meets Physical Therapy". *Communications of the ACM* **40**(8): pp 59-60.
(Brainmaster 1999)	Brainmaster.com (1999) "A Better Way to Look at Your Brain" http://www.brainmaster.com/
(Buxton 1990)	Buxton, W. (1990). "There's more to interaction than meets the eye: Some issues in manual input", *Human-Computer Interaction*. J. Preece & L. Keller. Prentice Hall**:** pp 122-137.
(Card 1983)	Card, S. K., Moran, T. P., Newell, A. (1983). *The Psychology of Human-Computer Interaction*. Hillside, NJ, Lawrence Erlbaum Associates.
(Carlsson 1993)	Carlsson, C., Hagsand, O. (1993). "DIVE - A Platform for Multi-User Virtual Environments", *Computers and Graphics* **17**(6) pp 394- 400
(Carnegie 1999)	Carnegie Mellon Research Institute. (1999). "Alert at the Wheel" http://www.cmu.edu/cmri/adm_alert.html
(Ciarcia 1988)	Ciarcia, S. (1988). "Computers on the Brain", *Byte*, June 1988: pp 273 - 285.
(Colebourne 1998)	Colbourne, A. (1998). "VR-Mog: Designing 3D worlds using 2D UI methods" http://www.comp.lancs.ac.uk/computing/research/cseg/projects/vrmog/.
(Colebourne 1999)	Colbourne, A. (1999). "AC3D Modeller" http://www.comp.lancs.ac.uk/computing/users/andy/ac3d.html.

(Cosmo 1999a) Cosmo (1999a). “Cosmo Worlds 2.0” http://www.cosmosoftware.com/products/worlds/.

(Cosmo 1999b) Cosmo (1999b). “Cosmo Player” http://cosmosoftware.com/download/player.html.

(Coutaz 1987) Coutaz, J. (1987). “Pac, an object-oriented model for dialog design”, *Human-Computer Interaction (Interact '87)*, Amsterdam, North-Holland, pp 431-436

(Dannenberg 1992) Dannenberg, R. B., Joseph, R. L. (1992). “Human-Computer Interaction in The Piano Tutor” *Multimedia Interface Design*. Eds M. Blattner, & R. B. Dannenberg. New York, ACM Press**:** pp 65-78.

(Delmas 1993) Delmas, S. (1993). *XF: Design and Implementation of a Programming Environment for Interactive Construction of Graphical User Interfaces.* Berlin, Technische Univesitat.

(Denney 1991) Denney, M. R., Baugh, J.L., Hardt, H.D. (1991). “Sobriety outcome after alcoholism treatment with biofeedback participation: a pilot inpatient study”, *International Journal of Addiction* **26**(3): pp 335-341.

(Dewan 1967) Dewan, E. M. (1967). “Occipital Alpha Rhythm Eye Position and Lens Accomodation”, *Nature* **214**: pp 975-977.

(Dix 1998) Dix, A., Finlay, J., Abowd, G., Beale, R. (1998). *Human-Computer Interaction*. London, Prentice Hall.

(Dyson 1999) Dyson, B. (1999). “The Role of Signal Presentation in Frontalis Electromyographic Biofeedback”, *Masters Thesis*. Psychology Department, Lancaster University.

(Eckstein 1998) Eckstein, R., Loy, M., Wood, D. () Java Swing. O'Reilly, UK.

(EEG Spectrum 1999) EEG Spectrum (1999) “Neurofeedback Research and Clinical Services” http://www.eegspectrum.com

(Edelberg 1972) Edelberg, R. (1972). “Electrical Activity of the Skin”, *Handbook of Psychophysiology*. Eds. N. S. Greenfield, Sternbach, R. A. New York, Holt, Rinehart and Winston, Inc. pp 367-418

(Engelberg 1994) Engelberg, M. (1994). “An Introduction to VR at NASA Johnson Space Centre” http://www.jsc.nasa.gov/cssb/vr/Equipment/tools.html.

(Englander 1997) Englander, R. (1997). *Developing Java Beans*, O’Reilly, UK.

(Fahrion 1992) Fahrion, S. L., Walters, E.D., Coyne, L., Allen, T (1992). “Alterations in EEG amplitude, personality factors, and brain electrical mapping after alpha-theta brainwave training: a controlled case study of an alcoholic in recovery”, *Alcoholism, Clinical and Experimental Research* **16**(3): pp 547-552.

(Farewell 1988)	Farewell, L. A., Dochin, E. (1988). "Talking off the top off your head: toward a mental prosthesis utilizing event-related brain potentials", *Electroencephalography and Clinical Neurophysiology* **70**: pp 520-523.
(Fernandez 1997)	Fernandez, R. (1997). "Stochastic Modelling of Physiological Signals with Hidden Markov Models: A Step Toward Frustration Detection in Human-Computer Interfaces". *Masters Thesis*. Media Laboratory, MIT.
(Flanagan 1997)	Flanagan, D. (1997). *Java in a Nutshell*, O'Reilly.
(Gaudette 1983)	Gaudette, M., Prins, A.,Kahane, J. (1983). "Comparison of auditory and visual feedback for EMG training", *Perceptual and Motor Skills*, **56**: pp 383-386
(Gips 1996)	Gips, J., Olivieri, P. (1996). "EagleEyes: An Eye Control System for Persons with Disabilities" http://www.cs.bc.edu/gips/EagleEyes/EEReport.html
(Goldberg 1983)	Goldberg, A., Robson, D. (1983). *Smalltalk-80: The language and its implementation*, Addison-Wesley.
(Goldstein 1972)	Goldstein, I. B. (1972). "Electromyography: A Measure of Skeletal Muscle Response", *Handbook of Psychophysiology*. Eds. N. S. Greenfield & R. A. Sternbach, Holt, Rinehart and Winston, Inc**:** pp 329-365.
(Goleman 1995)	Goleman, D. (1995). *Emotional Intelligence*, Bloomsbury Paperbacks.
(Graham-Rowe 1998)	Graham-Rowe, D. (1998). "Think it and it's done", *New Scientist*, 17th October: p 5.
(Greenhalgh 1999)	Greenhalgh, C. (1999). *Large Scale Collaborative Virtual Environments*. London, Springer.
(Gregory 1987)	Gregory, R., L. (1987). *The Oxford Companion to the Mind*, Oxford University Press.
(Hagan 1995)	Hagan, J., Peterson, R. (1995). *The Engineering of Social Control: The Search for the Silver Bullet. In Crime and Inequality*, Stanford University Press.
(Hare 1982)	Hare, J. F., Timmons, B.H., Roberts, J.R., Burman, A.S. (1982). "EEG alpha-biofeedback training: an experimental technique for the management of anxiety", *Journal of Medical Engineering and Technology* **6**(1): pp 19-24.
(Hart 1990)	Hart, S. G., Wickens, C. D. (1990). "Workload assessment and prediction", *Manprint: An approach to systems integration*. Ed. H. R. Booher. New York, Van Nostrand Reinhold**:** pp 257-296.

(Hassett 1978) Hassett, J. (1978). *A Primer of Psychophysiology*. W. H. Freeman

(Healy 1997) Healy, J., Picard, R. (1997). “Digital Processing of Affective Signals”. *Technical Report* No. TR#444, Massachusetts, MIT.

(Henderson 1986) Henderson, D. (1986). “The Trillium User Interface Design Environment”, *Proceedings of SIGCHI'86*, New York, ACM Press. pp 221-227

(Hinckley 1999) Hinckley, K., Sinclair, M. (1999). “Touch-Sensing Input Devices”, *Proceedings of CHI'99*, ACM Press.

(Hockey 1989) Hockey, G. R. J., Briner, R. B., Tattersall, A. J., Wiethoff, M. (1989). “Assessing the impact of computer workload on operator stress: The role of system controllability”, *Ergonomics* **32**: 1401-1418.

(Horst 1984) Horst, R. L., Ruchkin. D. S. (1984). “Event-related potential indices of workload in a single task paradigm”. *Human Factors Society 28th Annual Meeting*, Santa Monica, CA, Human Factors Society.

(Howard 1996) Howard, T. (1996). “Beyond the Big Barrier” http://www.cs.man.ac.uk/staff/toby/bluesky/bci/bci.htm.

(Hunt 1996) Hunt K.J., Munih M. & Donaldson N. de N. (1997). “Feedback control of unsupported standing in paraplegia. Part I: Optimal control approach”. *IEEE Transactions on Rehabilitation Engineering* **5**(4), , pp.331-40.

(I^3 2000) Institute of Interventional Informatics (2000) http://www.pulsar.org/aboutpulsar/i3.html

(IBVA 1998) IBVA Technologies Inc. (1998) http://ibva.com/

(Isreal 1980) Isreal, J. B., Chesney, G. L., Wickens, C. D., Donchin, E (1980). “P300 and tracking difficulty: Evidence for multiple resources in dual-task performance”, *Psychophysiology* **17**: pp 259-273.

(Jones 1991) Jones, S., Downtown, S (1991). “Windowing systems: high- and low-level design issues”, *Engineering the human-computer interface*. A. Downtown. London, McGraw-Hill**:** pp 246-292.

(Junker 1995) Junker, A., Berg, Christian., Schnider, Patricia. (1995). “Evaluation of the Cyberlink Interface as an Alternative Human Operator Controller”. *Technical Report* No. AC/CF-TR-1995-0011, Wright-Patterson Air Force Base, OH.

(Kamiya 1969) Kamiya, J. (1969). “Operant control of the EEG alpha rhythm and some of its reported effects on consciousness”, *Altered States of Consciousness*. C. T. Tart. New York, Wiley.

(Keirn 1990) Keirn, Z. A., Aunon, Jorge I. (1990). “Man-Machine Communications

Through Brain-Wave Processing", *IEEE Engineering in Medicine and Biology Magazine* March 1990: pp 55-57.

(Kelly 1990) Kelly, M. F., Parker, P. A., Scott, R. N. (1990). "The Application of Neural Networks to Myoelectric Signal Analysis: A Preliminary Study", *IEEE Engineering in Medicine and Biology* **37**: pp 221-229.

(Kirkup 1997) Kirkup, L., Searle, A., Craig, A., McIsaac, P., Moses, P. (1997). "EEG-Based System for Rapid On-Off Switching Without Prior Learning". *Medical and Biological Engineering and Computing* **35**: pp 504-9

(Knapp 1990) Knapp, R. B., Lusted, H. S. (1990). "Bioelectric controller for computer music applications". *Computer Music Journal* **14**(1): pp 42-47.

(Kramer 1987) Kramer, A. F., Sirevaag, E. J., Braune, R. (1987). "A Psychophysiological Assessment of Operator Workload During Simulated Flight Missions", *Human Factors* **29**(2): pp 145-160.

(Kramer 1991) Kramer, A. F. (1991). "Physiological metrics of mental workload: A review of recent progress", *Multiple-Task Performance*. L. Damos. London, Taylor and Francis.

(Krasner 1988) Krasner, G., Pope, S. (1988). "A Cookbook for using the Model-View-Controller User Interface Paradigm in Smalltalk 80", *Journal of Obejct-Oreinted Programming (JOOP)* **1**(3): pp 26-49.

(Kübler 1999) Kübler, A., Kotchoubey, B., Hinterberger, T., Ghanayim, N., Perelmouter, J., Schauer, M., Fritsch C., Taub, E., Birbaumer, N. (1999). "The thought translation device: a neurophysiological approach to communication in total motor paralysis", *Experimental Brain Research* **124**(2): pp 223-232.

(Kuhlman 1978) Kuhlman, N. (1978). "Functional Topgraphy of the Human Mu Rhythm", *Encephalography and Clinical Neurophysiology* **44**: pp 83-93.

(Lake 1997) Lake, C. (1997). "Effects of Prosthetic Training on Upper-Extremity Prosthesis Use", *Journal of Prosthetics and Othotics* **9**(1): pp 3-9.

(Lecolinet 1996) Lecolinet, E. (1996). "XXL: A Dual Approach for Building User Interfaces", *Proceedings of the ACM Symposium on User Interface Software and Technology (UIST'96)*, Seattle, WA. pp 99-108

(Lipson 1998) Lipson, E. (1998). "AnyWear" http://www.mindtel.com/mindtel/anywear.

(Lubar 1995a) Lubar, J. F. (1995a). "Neurofeedback for the Management of Attention Deficit/ Hyperactivity Disorders", *Biofeedback: A Practitioner's Guide*. M. S. Schwartz. New York, Guilford Press: pp 493-525.

(Lubar 1995b) Lubar, J. F., Swartwood, M. O., Swartwood, J. N., Timmermann D. L. (1995b). "Quantitative EEG and Auditory Event-Related Potentials in

	the Evaluation of Attention-Deficit/Hyperactivity Disorder: Effects of Methylphenidate and Implications for Neurofeedback Training", *Journal of Psychoeducational Assessment (ADHD Special)*: pp 143-160.
(Lubar 1997)	Lubar, J. F. (1997). "Neocortical Dynamics: Implications for Understanding the Role of Neurofeedback and Related Techniques for the Enhancement of Attention". *Applied Psychophysiology and Biofeedback* **22**(2): pp 111 - 126.
(Lusted 1996)	Lusted, H. S., Knapp, Benjamin R. (1996). "Controlling Computers with Neural Signals", *Scientific American*: pp 58-63.
(Macedonia 1995)	Macedonia, M. R., Yoshida, C., Nishimura, T., Ishida, T. (1995). "Exploiting Reality with Multicast Groups: A Network Architecture for Large-scale Virtual Environments", *Virtual Reality Annual International Symposium (VRAIS'95)*, North Carolina, IEEE Computer Society, pp 2-10
(Martin 1980)	Martin, I., Venable, P. H. (1980). *Techniques in Psychophysiology*, John Wiley & Sons, Inc.
(McDaniel 1997)	McDaniel, R. G., Myers, B. A. (1997). "Gamut: Demonstrating Whole Applications", *Proceeding of the ACM Symposium on User Interface Software and Technology (UIST'97)*, Banff, Canada, ACM Press, pp 81-82
(McKenna 1996)	McKenna, P. (1996). "Look Ma, No Hands: Researchers harness brain power to control jets". http://www.af.mil/news/airman/0296/look.htm.
(McMillan 1995)	McMillan, G. R., Eggleston, Robert G., Anderson, Timothy R. (1995). "Nonconventional Controls". *Handbook of Human Factors and Ergonomics*. G. Salvendy, John Wiley and Sons, Inc.: pp 729-771.
(Metz 1997)	Metz, S. M., Hoffman, Beth (1997). "Mind Operated Devices", *Cognitive Technology* **2**(1): pp 69-74.
(Mickish 1990)	Mickish, A., Marchal, P. (1990). "Garnet: Comprehensive Support for Graphical, Highly Interactive User Interfaces", *Readings in Human-Computer Interaction: Toward the Year 2000*. Eds. R. M. Baecker, J. Grudin, W. A. S. Buxton, S. Greenberg. Morgan Kaufmann Publishers: pp 357-371
(Microsoft 1998)	Microsoft Corporation. (1998). *Microsoft Visual Basic 6.0 Programmer's Guide*, Microsoft Press.
(Mulder 1987)	Mulder, L. J. M., Mulder, G (1987). "Cardiovascular reactivity and mental workload", *The beat-by-beat investigation of cardiovascular function*. Eds O. Rompelman, R. I. Kitney, Oxford, Oxford University Press.
(Myers 1989)	Myers, B., Vander Zanden, B., Dannenberg, R (1989). "Creating

Graphical Objects by Demonstration", *Proceedings of User Interface Software and Technology (UIST'89)*, ACM Press. pp 95-104

(Myers 1990) Myers, B., Guise, D.A., Dannenberg, R.B., Vander Zanden, B., Kosbie, D.S., Pervin, E., Mickish, A., Marchal, P. (1990). "Garnet: Comprehensive Support for Graphical, Highly Interactive User Interfaces", *Readings in Human-Computer Interaction: Toward the Year 2000*. R. M. Baecker, J. Grudin, W. A. S. Buxton, S. Greenberg. San Francisco, CA, Morgan Kaufmann Publishers, Inc: pp 357-371

(Myers 1995) Myers, B. A. (1995). "State of the Art in User Interface Software Tools", *Readings in Human-Computer Interaction: Toward the Year 2000*. R. M. Baecker, J. Grudin, W. A. S. Buxton, S. Greenberg. San Francisco, CA, Morgan Kaufmann Publishers, Inc: pp 323-343.

(NASA 1999a) NASA, L. R. C. (1999). "Human Error in Aviation" http://chem.larc.nasa.gov/HumanFactors/error.htm.

(NASA 1999b) NASA, L. R. C. (1999). "Crew Hazards and Error Management" http://chen.larc.nasa.gov/.

(NASA 1999c) NASA, L. R. C. (1999). "Attention" http://chem.larc.nasa.gov/HumanFactors/attn.htm.

(Natani 1981) Natani, K., Gomer, F. E. (1981). "Electrocortical activity and operator workload: A comparison of changes in the electroencephalogram and in event-related potentials". *Technical Report*, McDonnell Douglas.

(Nelson 1996) Nelson, T. W., Hettinger, Lawrence J., Cunningham, James A., Roe, Merry M., Lu, Liem G., Haas, Michael W., Dennis, Leon B., Pick, Herbert L., Junker, Andrew, Berg, Christian B. (1996). "Brain-Body Actuated Control: Assessment of an Alternative Control Technology for Virtual Environments". *1996 Image Conference*, Scottsdale, Arizona. Pp 225-232

(Norman 1986) Norman, D. A. (1986). "Cognitive Engineering", *User Centered System Design*. D. A. Norman, Draper, S. W. Hillside, NJ, Lawrence Erlbaum Associates: pp 31-61.

(Norman 1992) Norman, D. A. (1992). *Turn Signals Are the Facial Expressions of Automobiles*, Addison-Wesley.

(Olsen 1995) Olson, P. R. (1995). "Definitions of Biofeedback and Applied Psychophysiology", *Biofeedback: A Practitioner's Guide*. M. S. Schwartz. New York, The Guilford Press: pp 27-31

(Olsen 1997) Olsen, D. R. J. (1997). "Interactive Software Architecture", *Handbook of Human-Computer Interaction*. Eds M. G. Helander, T. K. Landauer, P. V. Prabhu, Amsterdam, North-Holland: pp 1147-1158.

(Other90 1999) Other90.com (1999). "The MindDrive" http://www.other90.com/

(Ousterhout 1994) Ousterhout, J. (1994). *An Introduction to Tcl and Tk*, Addison-Wesley.

(Palfreman 1991) Palfreman, J., Swade, D. (1991). *The Dream Machine*. London, BBC Books.

(Patmore 1994) Patmore, D., Putnam, W.L., Knapp, R.B. (1994). “Assistive Cursor Control for a PC Window Environment: Electromyogram and Electroencephalogram Based Control”, *Online Proceedings of the Center on Disabilities Virtual Reality Conference*, California State University, Northridge CA. http://www.csun.edu/cod/94virt/wec~1.html

(Peek 1995) Peek, C. J. (1995). “A Primer of Biofeedback Instrumentation” *Biofeedback: A Practitioner's Guide*. Ed. M. S. Schwartz, New York, The Guilford Press: pp 45-95.

(Peniston 1989) Peniston, E. G., Kulkosky, P.J. (1989). “Alpha-theta brainwave training and beta-endorphin levels in alcoholics”, *Alcoholism, Clinical and Experimental Research* **13**(2): pp 271-279.

(Peniston 1991) Peniston, E. G., & Kulkosky, P.J. (1991). “Alpha-theta brainwave neuro-feedback therapy for Vietnam veterans with combat-related post-traumatic stress disorder”, *Medical Psychotherapy* **4**: pp 47-60.

(Peniston 1993) Peniston, E. G., Marrinan, D.A., Deming, W.A., & Kulkosky, P.J. (1993). “EEG alpha-theta brainwave synchronization in Vietnam theater veterans with combat-related post-traumatic stress disorder and alcohol abuse”, *Advances in Medical Psychotherapy* **6**: pp 37-50.

(Penny 1996a) Penny, W., Roberts, Steve. (1996). “Baysian neural networks for detection of imagined finger movements from single-trial EEG”, Dept. of Electrical Engineering, *Technical Report*, Imperial College, London

(Penny 1996b) Penny, W., Roberts, S., Stokes, M. (1996). “Predicting side of finger movement from EEG”. *Technical Report*, Dept. of Electrical Engineering, Imperial College, London.

(Penny 1998) Penny, W. D., Robert, S. J., Stokes, M. J. (1998). “Imagined Hand Movement Identified for the EEG Mu-Rhythm”. *Journal of Neuroscience Methods* (submitted).

(Pfaff 1985) Pfaff, G., ten Hagen, P. J. W. (1985). *Seeheim Workshop on User Interface Management Systems*. Berlin, Springer-Verlag.

(Pfurtscheller 1993) Pfurtscheller, G., Flotzinger, D, Kalcher, J. (1993). “Brain-Computer Interface: A new communication device for handicapped person” *Journal of Microcomputer Applications* **16**: pp 293-299.

(Pfurtscheller 1994) Pfurtscheller, G., Flotzinger, D., Neuper, C. (1994). “Differentiation between finger, toe and tongue movement in man based on 40Hz EEG”, *Electroencephalography and Clinical Neurophysiology* **90**: pp 456-460.

(Picard 1997) Picard, R. W. (1997). *Affective Computing*, MIT Press.

(Pittman 1990) Pittman, J., Kitrick, C. (1990). “A Visual User Interface Management System”, *Proceedings of the ACM Symposium on User Interface Technology (UIST’90)*, Utah, ACM Press. pp 36-46

(Pope 1993) Pope, A. T., Bogart, E. H. (1993). “Identification of Hazardous Awareness States in Monitoring Environments”. *SAE Technical Paper* No. 921136 SAE 1992 Transactions: Journal of Aerospace, vol 101, pp 449-457

(Pope 1995) Pope, A. T., Bogart, E. H., Bartolome, D. S. (1995) “Biocybernetic System Evaluates Indices of Operator Engagement in Automated Task”, *Biological Psychology, Special Edition: EEG in Basic and Applied Settings*, vol. 40, pp 187-195

(Posner 1997) Posner, M., Raichle, M. E. (1997). *Images of Mind*, Scientific American Library, New York.

(Pressman 1994) Pressman, R. S. (1994). *Software Engineering*, McGraw-Hill, London.

(Rosenberg 1998) Rosenberg, R. (1998). “Computing without Mice and Keyboards: Text and Graphic Input Devices for Mobile Computing”. *Ph.D Thesis*. Computing Department, UCL, London.

(Rugg 1995) Rugg, M. D., Coles M. G. H. (1995). *Electrophysiology of Mind*, Oxford Science Publications.

(Salgado 1999) Salgado, R. (1999). “NeatTools” http://www.pulsar.org/neattools/index.html.

(Saridis 1982) Saridis, G. N., Gootie, T. P (1982). “EMG Pattern Analysis and Classification for a Prosthetic Arm”. *IEEE Engineering in Medicine and Biology* **29**: pp 403-411.

(Saxby 1995) Saxby, E., Penniston, E. G. (1995). “Alpha-theta brainwave neurofeedback training: an effective treatment for male and female alcoholics with depressive symptoms”, *Journal of Clinical Psychology* **51**(5): pp 685-693.

(Scheifler 1986) Scheifler, R., Gettys, J. (1986). “The X Window System”, *ACM Transaction on Graphics* **5**(2): pp 79-109.

(Schoon 1998) Schoon, N. (1998). “The computer that can hack into your emotions”, *Independent Newspaper*. 17th February 1998. London. pp 1

(Schmucker 1986) Schmucker, K. (1986). “MacApp: An application framework”, *Byte*, August 1986: pp 189-193.

(Schwartz 1995) Schwartz, N. M., Schwartz, M. S. (1995). “Definitions of Biofeedback

	and Applied Psychophysiology", *Biofeedback: A Practitioner's Guide*. M. S. Schwartz. New York, The Guilford Press: pp 32-42.
(Shagass 1972)	Shagass, C. (1972). "Electrical Activity of the Brain", *Handbook of Psychophysiology*. N. S. Greenfield, Sternbach, R.A., Holt, Rinehart and Winston, Inc: pp 263-328.
(Siddle 1980)	Siddle, D. A. T., Turpin, G. (1980). "Measurement, quantification and analysis of cardiac activity", in *Techniques in Psychophysiology*, Martin, I., Venables, P. H., John Wiley & Sons, Inc: pp 139-246.
(Singh 1990)	Singh, G., Green, M. (1990). "Druid: A system for demonstrational rapid user interface development", *Proceedings of the ACM SIGGRAPH Symposium on User Interface Software and Technology (UIST'90)*, Utah, ACM Press. pp 167-177
(Smith 1995)	Smith, G. B. (1995). "A Shared Object Layer to Support Cooperative User Interfaces", *Ph.D Thesis*, Lancaster University, UK.
(Sommerville 1992)	Sommerville, I. (1992). *Software Engineering*. Addison-Wesley.
(Sowizral 1997)	Sowizral, H., Rushforth, K., Deering, M. (1997). *The Java 3D API Specification*, Addison-Wesley.
(Springer 1993)	Springer, S. (1993). *Left Brain, Right Brain*. New York, W. H. Freeman.
(Strayer 1986)	Strayer, D. L., Kramer, A. F. (1986). "Psychophysiological indices of automaticity and attentional resources", *Psychophysiology* **23**: p 466.
(Superscape 1999)	Superscape (1999). "Superscape Interactive 3D Software" http://www.superscape.com/.
(Sutter 1992)	Sutter, E. E. (1992). "The brain response interface: communication through visually-induced electrical brain responses", *Journal of Microconputer Application* **15**: pp 31-45.
(Tansey 1993)	Tansey, M. A. (1993). "Ten-year stability of EEG biofeedback results for a hyperactive boy who failed fourth grade perceptually impaired class", *Biofeedback and Self-Regulation* **18**(1): pp 33-44.
(Tattersall 1980)	Tattersall, A., Hockey, G. (1980). "Dynamic decisions and workload in multitask supervisory control", *IEEE Transactions on Systems, Man and Cybernetics* **10**: pp 217-232.
(TDC 1996)	Transportation Development Canada (TDC). (1996). "Commercial Motor Vehicle Driver Fatigue and Alertness Study" http://www.tc.gc.ca/TDC.publicat/summ.htm
(Took 1990)	Took, R. (1990). "Surface Interaction : A paradigm and model for separating application and interface", *Proceedings of CHI'90*, ACM Press.

(UltraMind 2000)	UltraMind (2000). http://www.ultramind.com
(University of Alberta 1999)	University of Alberta, C. G. R. G. (1999). “MR Toolkit: Virtual Reality and 3D and 2D User Interface Software Tools”. http://www.cs.ualberta.ca/~graphics/MRToolkit.html.
(Vincente 1987)	Vincente, K. J., Thornton, C., Moray, N (1987). “Spectral Analysis of Sinus Arrhythmia: A Measure of Mental Effort”, *Human Factors* **29**(2): pp 171-182.
(Walter 1953)	Walter, W. G. (1953). *The Living Brain*, W. W. Norton and Company.
(Warner 1999)	Warner, D. (1999). “The Pulsar Project” http://www. pulsar.org.
(Wastell 1999)	Wastell, D. (1999). Personal Communication.
(Waters 1996)	Waters, R. C., Anderson, D. B., Barras, J. W., Brogan, D. C., Casey, M. A., McKeowan, S. G., Nitta, T., Sterns, I. B., Yerazunis, W. S. (1996). “Diamond Park and SPLINE: A Social Virtual Reality System with 3D Animation, Spoken Interaction and Runtime Modifiability”, *Technical Report*, MERL
(Watson 1978)	Watson, C. G., Herder, J., Passini, F.T. (1978). “Alpha biofeedback therapy in alcoholics: an 18-month follow-up”, *Journal of Clinical Psychology* **34**(3): pp 765-769.
(WaveAccess 1997)	WaveAccess (1997) *WaveRider User Manual*, WaveAccess, P.O. Box 1764, Sebastobol, CA 95472 USA
(Wickens 1983)	Wickens, C. D., Kramer, A. F. Vanassew, L., Donchin, E (1983). “The reciprocity of primary and secondary task resources: Evidence from the P300 component of the ERP”, *Science* **221**: pp 1080-1082.
(Wiener 1948)	Wiener, N. (1948). *The Human Use of Human Beings: Cybernetics and Society*. Boston, Houghton, Mifflin and Company.
(Wiener 1961)	Wiener, N. (1961). *Cybernetics: or Control and Communication in the Animal and the Machine*. Cambridge, MA, MIT Press.
(Wolpaw 1990)	Wolpaw, J. R., McFarland, Dennis J., Neat, Gregory W., Forneris, Catherine A. (1990). “An EEG-based brain-computer interface for cursor control”. *Electroencephalography and Clinical Neurophysiology* **78**: pp 252-259.
(Wolpaw 1994)	Wolpaw, J. R., McFarland, Dennis J. (1994). “Mulitchannel EEG-based brain-computer communication”, *Electroencephalography and Clinical Neurophysiology* **90**: pp 444-449.

Appendix A

Physiological Sensing Hardware Manufacturers

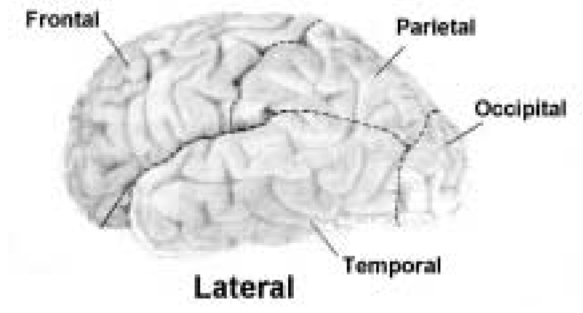
Device Name	Manufacturer	Details
GSR2	Thought Technology 2180 Belgrave Avenue Montreal, Quebec, Canada H4A 2L8 1-800-361-3651 – Tel (514) 489-8255 – Fax	❑ Single-channel, stand alone GSR detection and presentation device.
EMG Retrainer	Chattanooga Group 4717 Adams Road P.O. Box 489 Hixson, TN 37343 (423) 870-2281 – Tel (800) 242-8329 – Fax	❑ Two-channel, stand alone EMG signal detection and presentation device.
MyoTrac	Thought Technology (see above)	❑ Single-channel, stand alone EMG signal detection and presentation device.
BioGraph	Thought Technology (see above)	❑ Multi-channel, multiple signal detection device. ❑ Serial peripheral. ❑ Signal display via proprietary computer application (ProComp available from Thought Technology).
AnyWear	Institute of Interventional Informatics 500 University Place, Syracuse, NY 13210 (315) 423-4585 – Tel (315) 423-4676 – Fax	❑ Multi-channel EMG signal detection device ❑ Serial peripheral ❑ Signal display via proprietary computer application (NeatTools available from Inst. Inter. Inf.).
BioMuse	BioControl Systems, Inc. 2555 Park Blvd. Palo Alto, CA, 94306 (415) 329-8494 – Tel (415) 329-8498 – Fax	❑ Multi-channel, multiple signal detection device. ❑ Serial peripheral. ❑ Signal display via proprietary computer application.
MindDrive	Discovogue Infotronics 411000 Modena Italy (39) 059-220-060 – Fax	❑ Single channel GSR sensing device ❑ Serial peripheral ❑ Signal display via proprietary computer application.
RelaxPlus	www.ultromind.co.il	❑ Single channel GSR sensing device ❑ Serial peripheral ❑ Signal display via proprietary computer application.
Neurocybernetics II	EEG Spectrum Neurofeedback Research and Clinical Services	❑ Multi-channel, multiple signal detection device.

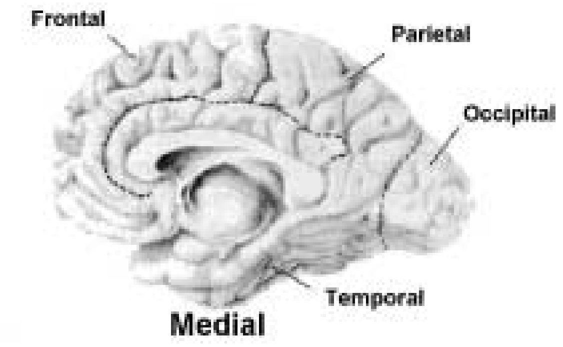
	16500 Ventura Blvd, Ste 418 Encino, CA 91436-2505 (818) 789-3456 – Tel	❑ Serial peripheral. ❑ Signal display via proprietary computer application
CyberLink	Brain Actuated Technologies, Inc. 139 E. Davis Street Yellow Springs, Ohio, USA 45387 937-767-2674 – Tel 937-767-7366 – Fax	❑ Multi-channel, multiple signal detection device. ❑ Serial peripheral. ❑ Signal display via proprietary computer application.
IBVA	IBVA Technologies, Inc. PMB 164 25-13 Old Kings Hwy, North Darien, CT 06820 203 838-4728 – Fax	❑ Multi-channel EEG sensing device ❑ Serial peripheral ❑ Signal display via proprietary computer applications
Brainmaster	BrainMaster Technologies P.O. Box 538 Chesterland, OH 44026-0538 (440) 350-9822 – Tel (440) 423-0090 – Fax	❑ Single channel EEG sensing device ❑ Serial peripheral ❑ Signal display via proprietary computer applications

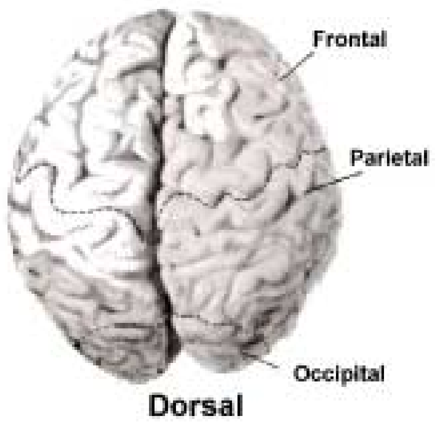
Appendix B

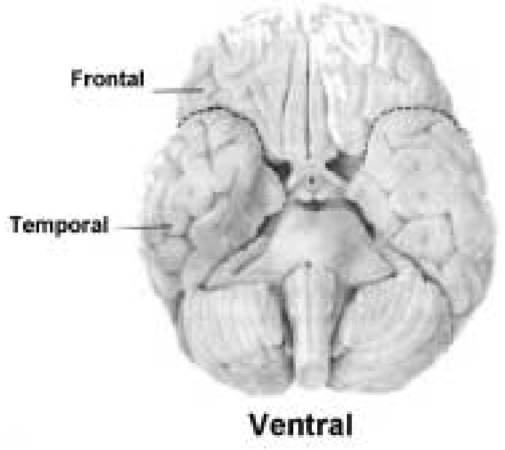
Physiological Appendix

B.1 Lobes of the Brain

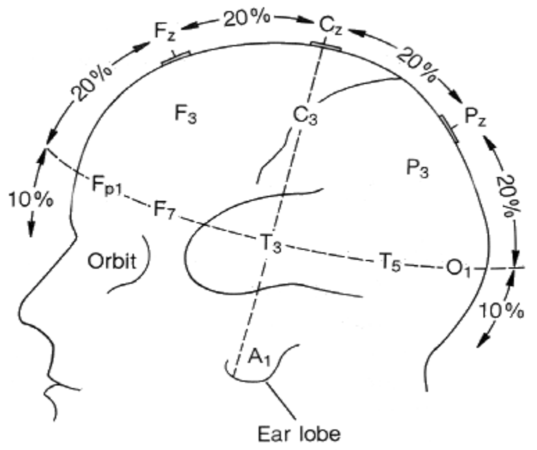




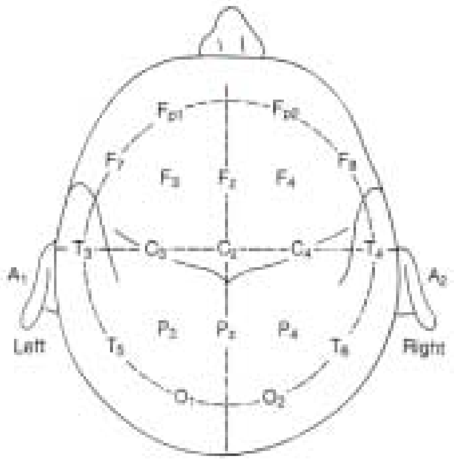




B.2 The International 10-20 System of Scalp Electrode Placements



Left Side of Head



Top of Head