A Speech Interface for Bedside Data Entry in an Intensive Care Unit

Marco Petroni

B. Eng., (McGill University), 1989

Department of Electrical Engineering

McGill University

Montréal

July, 1991

A thesis submitted to the Faculty of Graduate Studies and Research
in partial fulfillment of the requirements for the degree of
Master of Engineering

Abstract

This thesis presents the design and implementation of a speech interface for bedside data entry in an intensive care unit. A *speech interface* is a system comprised of a speech recognition system, for speech input, and a speech generation or speech synthesis system, for speech output. These interfaces allow the operation of a computer system using voice commands providing the user with feedback via speech output. Such systems permit users to perform "hands-free" and "eyes-free" data entry or system operation in circumstances where the use of traditional manual input devices, such as a keyboard, cannot be used. This thesis begins with a literature sampling of contemporary computerized medical information systems and speech interface systems followed by a description of the hardware and software architecture of the speech interface implemented. Test results are then presented and discussed followed by an outline of future extensions for the system.

Sommaire

Cette dissertation présente la conception et la mise en application d'une interface vocale pour l'entrée des aonnées prises au chevet du lit dans une unité de soins intensifs. Une *interface vocale* est composée d'un système de reconnaissance de la parole, pour répondre à des commandes vocales, et d'un synthétiseur de la parole, qui agit comme périphérique de sortie de paroles qui ont été numérisées ou générés de fa con synthétique. De telles interfaces permettent aux mains et aux yeux des usagers de rester libres pendant l'interaction avec le système informatique en utilisant des commandes vocales. Les interfaces vocales permettent au logiciels réagir avec des paroles. Cette thèse débute avec un exposé littéraire sur les systèmes informatiques médicaux et sur les interfaces vocales. La configuration machine et logicielle de l'interface vocale mise en application est décrite et les résultats des séances d'essais sont présentés et analysés. Avant de conclure, les grandes lignes des expansions futures sont discutées.

Acknowledgements

At the conclusion of this long and exciting process, my first thoughts go out to my parents, who provided me with the support and understanding I needed to get through this experience in graduate studies. The financial support of NSERC is also gratefully acknowledged.

Thanks must also be expressed to my supervisor, Alfred Malowany, for allowing me to discover the interesting and rewarding field of medical informatics, and for the inspiring discussions that assisted me to the very end. I would also like to acknowledge the help and input of Franco Carnevale and Dr. Ron Gottesman of the Montreal Children's Hospital for sensitizing me, and the rest of the PDMS team, to the issues and considerations specific to the intensive care unit environment; our collaboration has been both enriching and exciting.

An acknowledgment is also extended to the countless undergraduate students who have contributed in one way or another to the PDMS project. A special thanks goes out to the NSERC summer students Chantal, Han, and Lifang, who were responsible for implementing portions of the design.

Last but not least, a special thanks to all my fellow students in McRCIM, with whom I shared many adventures, and also a very special thanks to the "veterans" Nick, Christian, Mathieu, Jean, Kathleen, and Jean-François. They deserve praise for their help, patience, suggestions, and friendship.

Table of Contents

List of Figures

List of Tables

Chapter 1 Introduction

With the proliferation of computers as prices fell and their functionality grew, came their application in a myriad of diverse fields. Perhaps the most impressive and beneficial application of computers is in the medical domain where computers perform tasks ranging from accounting to 3-D imaging of parts of the human body, allowing surgeons a more detailed view than was previously possible.

Computerization has also had a large impact on hospital intensive care units (ICUs) allowing the continuous monitoring and display of the physiological parameters of a critically ill patient. The advent of these computerized monitoring systems in hospital ICUs, has removed the burden, on the health care staff, of having to continuously take patient readings allowing them to spend more time on patient care, and less doing tasks which can easily be handled by machines. In addition, many of these computerized monitoring systems perform some basic parameter analysis, generating alarms when either pre-set thresholds have been exceeded or when a combination of parameter changes may indicate an impending problem. These systems provide a continuous, watchful "eye" on the patient while the health care professionals are busy with other tasks at the patient bedside.

As in most other applications, computerization and automation in the ICU has lead to an increase in the amount of data which is available to be viewed and interpreted by the nurses and physicians. This creates the need for a means of accumulating, presenting, and also assisting in the interpretation of this data in a manner which is meaningful to the health care professionals. If a data management system is not provided, the extra data generated by automation will not serve any useful purpose. In turn, computerized systems which require extensive human interaction, must be designed both with the human operator in mind, and with an in depth understanding of the application in which a given system will be used. Otherwise, computerized data management systems will be doomed to be abandoned and shelved in favour of the current manual charting methods.

According to an article by Kriewall and Long [Kriewall and Long, 1991], computer-based medical systems can be categorized as being either *controllers* (artificial hearts, drug infusion systems), *information managers* (networks, interfaces, neural networks), or *Diagnostic Tools* (clinical lab equipment, blood pressure monitors, ICU monitors, positron emission tomography, etc.). This first chapter will mainly focus on the diagnostic tools and information manager components of computer-based medical systems.

The first section will present some background information on computers in intensive care units, focusing on the different applications and interaction modalities that have been used. The following section will present some other applications of computers in the medical field ranging from imaging to the use of computers in teaching. Following that section will be a presentation of other non-medical applications where computer automation has lead to an increase in the amount of data available for interpretation and viewing by human users, and how computers and interaction modalities have helped users cope with such a large influx of data.

1.1 Computer-Based Medical Systems

1.1.1 Historical Background

In the late 1970's, as the cost of computers began to come down and their computational power and functionality increased, it became possible for hospitals to place computers in many departments [Abrami and Johnson, 1990]. Software specialized for different specialized departments, such as the laboratory, the radiology department, and pharmacy began to appear. As the prices of computers continued to fall in the 1980's, it was possible for more hospitals to computerize increasing numbers of departments.

The computerization of medical instrumentation, in particular, presented health care professionals with enormous amounts of data to process. The overwhelming majority of this data was lost, however, as nurses would still continue to manually take readings from the monitors at half-hour or hourly intervals for accumulation, charting, and interpretation [Collet *et al.*, 1989]. Computerization at this stage had not yet eliminated either the transcription and entry errors inherent in the manual charting process, or the fact that potentially important variations in the patient's condition, between intervals when readings were taken would not be recorded.

In hospital intensive care units, where patients can have numerous physiological parameters being monitored at a given time, tremendous pressure was placed on hospital administrators to have the data acquisition and logging process automated [Carnevale, 1986] [Nolan-Avila and Shabot, 1987]. Despite the high technical standard of the computerized monitors, the complaint of many users was that the units lacked flexibility. In addition, the range of possible applications was often determined by the manufacturer with little or no user influence [Alesch *et al.*, 1991]. In order to cater to the increasing needs of the hospital ICU community, many computerized monitor vendors provided a means of getting at the data generated by their equipment, either by networking a group of monitors and providing an interface to the network to acquire the data, or by providing a direct interface to the monitor itself [Hewlett-Packard, 1987] [EMT, 1988]. This allowed development of automated computer systems which would acquire, process, display, and archive data relevant to a particular ICU ward, and thus, patient data management systems were born.

The use of computers in critical care gave rise to issues such as data storage and integrity, security, and the usability of these systems by novice users – questions which are still being addressed today.

1.1.2 Data Acquisition Systems

For the most part, Personal Computers, or PCs, are used for the collection, storage, analysis, presentation of data acquired from monitoring equipment. This section will present some of the PC-based systems as well as describing some of their functionality.

Alesch [Alesch *et al.*, 1991] describes a system based on a PC / AT platform which acquires, stores, and presents data which allows the determination of intercranial compliance, a feature which was not available from their patient monitor and had to be synthesized from data acquired from the monitor. Such data is essential in present day neurosurgery.

In order to present relevant information to physicians appropriate for decision making, a PC-based system for the collection and graphical display of patient data for comatose and head injured patients is being developed at the University of Louisville [Stickland Jr., 1991]. The information presented by the system consist of graphical trends and of data presented in a spreadsheet format.

Important information gained from viewing physiological patient parameters simultaneously is mentioned by Moser [Moser *et al.*, 1990] which describes another PC/AT-based system for computer-aided background patient monitoring in an ICU. This system, which was developed in-house, uses Personal Computer local area networks (LANs) to allow the information to be viewed at the patient bedside. This article also mentions that the archiving and "off-line" processing of this data may give some insight into the effects of new drugs on a patient.

Dasta [Dasta, 1990] mentions the important role that computers play in ICU environments, assimilating, processing, and displaying large amounts of data in easy-to-understand formats for the users. This article mentions that, although computers have been in the ICU for some years now, paper medical records are still being used. Three prototype patient data management systems are presented, and the use of knowledge-based programs which provide help to the clinician on such topics as acid-base management, hemodynamic monitoring, and shock management, is discussed.

The use of computers in the emergency room and in the intensive care unit to capture, and display biomedical signals is discussed by Aguglia [Aguglia *et al.*, 1990]. Computers are also used to provide support to the solution of medical problems with diagnostic and medical decision making software.

Another system described by James et al presents a PC/AT system which records, stores, and analyses graphically and statistically, neurophyiological and cardiovascular recording during an experiment [James *et al.*, 1990]. The system is composed of physiological amplifiers which connect to the PC/AT and whose output data can also be written on a laser printer.

The evaluation and comparison of computerized systems versus the manual data management systems is discussed by two articles by Tachakra et al [Tachakra *et al.*, 1990], and by Kari et al [Kari *et al.*, 1990]. In these articles, studies demonstrated that the computerized method of data acquisition and storage yielded significant savings in time, staff workload, and in accuracy. Both studies also showed that patient information was faster to access and to interpret by the physicians using the computerized systems than by using the manual paper charts.

In two studies of computer-based patient monitoring and data management systems done by Hammond et al [Hammond *et al.*, 1991a] [Hammond *et al.*, 1991b], results showed that the use these systems not only reduced non-nursing related work, but also improved the quality, quantity, and recall of the clinical information by the staff. This leads to a significant improvement in documentation flowsheets both in terms of quantity and accuracy which can in turn have medicolegal and quality assurance impacts, as well as enhancing patient care.

The use of computers to acquire and store anesthesia and ICU information is crucial for legal issues, states Gibbs [Gibbs, 1989]. The failure of meaningful, supportive, or exculpatory documentation raises serious questions about the quality of the care rendered. For this reason, Gibbs states that the use of computers for automated data acquisition and record keeping in anesthesia and ICUs is as crucial as the use of both the cockpit voice data recorders and flight data recorders in aviation.

In an excellent article by Holtermann et al [Holtermann *et al.*, 1990], the authors present their experience with computer-aided acquisition, processing and documentation of patient data in an intensive care unit. In their study, one PC was used for the data acquisition and processing. The authors determined that one central PC was impractical for this task, and advocate the use of many PCs networked to the central PC, which would act as a server. The distributive nature of the system would also allow the physicians to view patient information in their offices.

On-line data gathering and accumulation can also be used for "off-line" data analysis and research. An article by Schwid et al discusses the uses of data acquisition for the purposes of off-line data analysis and clinical research [Schwid *et al.*, 1990].

Another example of acquired data which is also processed "off-line" is a microcomputer communicating to a measuring interface via a serial RS-232C connection for the monitoring of heat and fluid balance in severely burned patients [Ferguson *et al.*, 1989]. The computer is not only used to monitor patients, but the data acquired is archived in a data base and used for future predictions of heat and fluid losses, given burn area and severity.

While many of the systems presented above boast of the advantages that computerized data acquisition provides, it is imperative for user acceptance that the system provide an effective person-machine interface. The next section discusses the different user interfaces used in computer-based medical systems.

1.1.3 User Interaction

Manual Input Techniques

While functionality and power are important in any application, what will inevitably and eventually determine the acceptance of a system by its end users is the quality and effectiveness of its user interface. Once thought of as simply "a string to tie the box with" for application programs, user interfaces have only recently become a "box" in their own right, requiring as much thought and as much consideration as the actual computational and functional aspects of an application program [Edmonds, 1990]. In addition, person-machine interaction devices are also an important consideration. The most common user interface on computers is, no doubt, the keyboard. While remarkable improvements have been achieved in computers and in their processing speed, only modest improvements have appeared in input and output devices. In the interest of reducing the time required and the errors made in entering or retrieving data many new devices and variants have appeared on the market. The devices range from variations on the standard "QWERTY" keyboard, to devices such as touch screens, and speech recognition and generation.

The primary mode of data entry is still done by keyboards. While still an effective mode of entering data for users who have mastered it, people who aren't familiar with the layout and location of the keys often resort to the familiar two-finger "hunt-and-peck" routine for typing. While this may be acceptable for people who are interacting with a computer at an information booth, it is definitely unacceptable for environments where computer data entry and retrieval must be done as quickly and as efficiently as possible.

One variation on the keyboard specifically for medical data management systems is discussed by Solingen and Shabot [Soligen and Shabot, 1988]. This keyboard has 32 keys, which consists of both numeric key entry and multiple function keys which allows single-key operation of a patient data management system.

System operation and effective entry of information can also be achieved by touchscreen as is shown by King and Smith [King and Smith, 1990]. The addition of a touchscreen to their system has caused a dramatic increase in the ability of casual users input data into the system effectively. For normal operations, this system could be made to function under touchscreen input only, removing the necessity of a keyboard.

Since many of the patient data management systems must also handle significant volumes of data input manually, input modalities other than the keyboard were sought. For electronic capture of data at the patient bedside, Hammond and Stead [Hammond and Stead, 1988] state that the presence of a permanent bedside terminal is not required. Medication usage of a given patient could be recorded by a portable bar-code reader if the nurse had a pre-printed paper medication sheet with the equivalent human-readable and bar-code-readable equivalents. They also state that numerical data, such as blood pressure and pulse could be entered through a portable keyboard device similar to those used in industrial inventory systems.

The use of portable computers to facilitate the calculation of hemodynamic parameters and to interpret arterial blood gas samples is discussed by Large [Large, 1990]. In this article, pocket computers are seen as a cheap and powerful means of allowing useful information to be presented at the patient bedside. The problem with these systems is the lack of dedicated software for clinical tasks.

A couple of articles review the interaction technologies which are available and which are in development and are useful for the medical field. Cohen [Cohen, 1990a] reviews the latest technologies which will enhance the medical professional's ability to interface with computer systems which include bar-coding, graphics, and expert systems. Murchie and Kenny [Murchie and Kenny, 1988] compare the input modalities of keyboard, light pen, and voice recognition for speed, errors, and the number of correction required to enter patient admission data. In this study, keyboard was shown to be the fastest, most accurate, easiest, and most preferred method of data entry.

Speech Recognition and Speech Generation

Perhaps the most discussed input and output modalities in the medical literature is the use of speech recognition and speech generation. As the recognition technology and its functionalitv improves and becomes more affordable, more users are considering the use of speech in their applications. Speech is useful in situations when at least one of the following conditions are met [Schneiderman, 1987]:

- the speaker's hands are busy,
- mobility is required,
- the speaker's eyes are occupied,
- harsh or cramped conditions preclude the use of a keyboard.

Speech interfaces have two components, the speech recognition component, and the speech generation component. The process of recognizing speech is a more difficult task than of generating speech. Speech recognizers must be able to deal with the numerous possible ways different people can say a given word as well as handling the way that the same user can say the same word. Due to the pseudo-random nature of the speech signal, basic pattern recognition techniques are not very good for speech recognition. The most successful speech recognition algorithms employ a statistical technique, well suited to the nature of the speech signal, known as hidden Markov models (HMMs) [Rabiner, 1989]. States in the HMMs can be viewed as corresponding roughly to acoustic events in the speech signal. While other methods such as dynamic time warping (DTW), vector quantization (VQ), and expert systems have also been attempted for performing speech recognition, HMMs are still the most successful method for performing speech recognition. An excellent treatment and comparison of the aforementioned speech recognition techniques is presented by O'Shaughnessy [O'Shaughnessy, 1987]. Research is currently ongoing to apply artificial neural networks to speech recognition, but they have yet to be used in commercial systems. An example of a neural net architecture for speech recognition is described in an article by Dalsgaard and Baekgaard [Dalsgaard and Backgaard, 1990]. Another very interesting example of neural networks which uses the anatomical and physiological findings on the afferent pathway from the ear to the cortex is presented by Kurogi [Kurogi, 1991].

Three styles of speech, in order of increasing difficulty, can be distinguished: *isolated-word* or *discrete utterance* speech, *connected-word* speech, and *continuous speech* [Reddy, 1976]. Continuous speech recognition allows natural conversational speech with little or no adaptation of speaking style imposed on system users. Most commercially available systems are *speaker-independent*, demonstrating good performance for those who trained the system. Since the training time and memory required in such systems grow linearly with the number of speakers, less accurate *speaker-independent* are useful if a large population is to be served.

Several articles present the issues involved with using speech as an input modality. Bergeron and Locke [Bergeron and Locke, 1990] present a good overview of current speech technology and applications. Martin [Martin, 1989] discusses the usefulness of speech in user interfaces versus the traditional keyboard and mouse input modalities. The pros and cons of speech as an input modality are discussed by various researchers, who have investigated its use in applications and present their results [Brennan *et al.*, 1991].

Speech generation can be either playback of digitized speech, or the generation of synthetic speech. Here, the arrival of special purpose digital signal processing (DSP) chips has allowed this field to grow rapidly in recent years. Currently, the tradeoffs in speech generation are among the conflicting demands of maximizing sound quality and minimizing memory usage. The vocabulary size of the speech synthesis system directly determines the amount of memory required. Speech synthesis involves the conversion of an input text into a speech waveforms using algorithms and some form of previously coded speech data. *Voice response* systems handle a input text of a limited vocabulary, while *text-to-speech* systems handle all input text. O'Malley [O'Malley, 1990] discusses the issues involved in text-to-speech conversion, and how it can be done.

A few applications which combine speech recognition and generation are documented in literature. The most common application of speech recognition is in the generation of diagnostic reports and of physician observations, where speech is seen as a viable alternative to the manual typing of reports.

Ikehira et al [Ikehira *et al.*, 1990] describe the entry of 580 of bone scintigram reports using a speech recognition system. Since speech recognition systems have a limited vocabulary, it was found that the use of the system decreased the complexity of the reports without excluding information, and the resulting reports entered by voice were more concise and clearer than the hand written equivalent.

A study on the application of speech recognition to record dental examination data is discussed by Feldman and Stevens [Feldman and Stevens, 1990]. The time and errors differential between the traditional method of tape recording the observations and using speech recognition were measured. Examinations typically required slightly more time to enter observations using speech recognition, while no difference in the error rate was found between the two methods.

Another application of speech recognition to reporting systems is mentioned by Matumoto et al [Matumoto *et al.*, 1987]. Here, automatic radiologic reports were generated with the aid of an automatic speech recognition system. Since the observations must first be dictated and then typed into a system, the use of speech recognition allowed the observations to be entered directly into the computer system. In this application, common observations were "coded" into one-word or multi-word representations, in order to avoid having to dictate common and long observations by voice. The system required more time to enter reports by voice than by typing. The vocabulary size used in this application consisted of about 340 words.

A larger, 1000-word vocabulary was used in another radiological report generation system by Robbins et al [Robbins *et al.*, 1987] which also used speech recognition. This group split their vocabulary up into five sections according to anatomical or subspecialty application (gastrointestinal radiology, neuroradiology, etc.). According to their results, 70% of a radiology report could be generated with the aid of this system. As with the other system described in the previous paragraph, this system had code words which would generate either whole lines, sentences, or even complete reports. Dictation time with voice entry was longer than typing in the reports by 20%.

Although both systems described above did not present any time savings in entering the reports, both teams stated that the systems were far from complete, and still required improvements, but that the preliminary results were encouraging them to continue to use speech as an input modality.

Other articles discuss other systems which uses speech recognition for medical reporting. An article by Cohen [Cohen, 1990b] describes a system which uses speech recognition for the voice entry of PAP smears. Joseph [Joseph, 1989] describes applications of medical reporting systems which use speech recognition for emergency medical doctors and pathologists. The systems described in these two articles closely resemble the ones discussed above which generate radiological reports.

An interesting application of speech recognition for "eyes-free" and "hands-free" operation is to the control of a microscope in a surgical unit. Liang and Narayanan [Liang and Narayanan, 1988] describe a system in which voice control eliminates the need for cumbersome foot pedals or proximity switches. This system also allows programmable controls to the remember certain points in the depth of focus, magnitude of zoom, or movement along the X-Y axis. These specified points can be revisited when the correct words are spoken by the surgeon.

Applications of speech generation in medical applications are not as numerous as the applications of speech recognition. An article by McIntyre and Nelson [McIntyre and Nelson, 1989] presents a system which uses human voice messages over non-verbal alarm signals to deliver warning messages in an ICU. Studies performed at the University of Alberta hospital showed that the use of non-verbal warnings were perceived to be threatening by some patients, and that there was also significant opportunity for error in interpretation by fatigued or anxious personnel. Human voice messages were shown to have significantly less interpretation errors over the non-verbal signals which led to the conclusion that taped voice messages merit more trials in ICUs.

Diethrich [Diethrich, 1988] describes a system in the cardiac ICU of the Arizona Heart Institute in which both speech recognition and speech generation is used to allow the nurse to enter data at using voice, freeing the nurse to attend to the patient while documenting the treatment. The speech recognition system used is capable only of recognizing isolated words. Voice synthesis is used to provide feedback of entered data, when visual communication via the computer screen, located at the patient bedside, is not possible.

Natural Language Interaction

Since speech recognition systems often supply either discrete-word or connected-word speech, which some have suggested may be unnatural. To address this issue, some have suggested the use of *natural language* interaction for the input of information or for equipment operation. The goal of natural language systems is to respond properly to arbitrary natural language sentences or phrases. Natural language interaction seldom provides enough context for issuing the next command, unless so designed, and "clarification dialogs" between the system and the user are often required. When the scope of a task domain is limited, opportunities exist for natural language systems.

An article by Rukos [Roukos, 1989] describes in depth, the issues involved in a natural language system for speech understanding. In this system, speech recognition and natural language knowledge sources must be combined. As in any natural language system, the language knowledge sources consist of a unification formalism for describing the syntax of English, with other higher-order logic language for representing the meaning of an utterance, and a framework for interfacing syntax and semantics.

Considerations of natural language interfaces and application experiences are related in a paper by Brunner et al [Brunner *et al.*, 1990]. This article also presents arguments for and against the use of these systems for commercial applications and the primary issues involved in the evaluation of these systems.

A proposed natural language application in the medical domain is described by Guerrouad [Guerrouad, 1989]. In this article the use of natural language for the control of operating room equipment is seen as a way of liberating the surgeon's concentration from the manipulation of equipment so that it can be focused on other tasks. Natural language input would not constrain the surgeon to use a constrained vocabulary as would a speech recognition system.

Another natural language system for dermatological diagnosis is described by Landau et al [Landau *et al.*, 1989]. The article first discusses why computer-aided diagnosis has not gained wide-spread acceptance in the medical community, and how this can be addressed through the use of natural language interfaces which would allow the physician to "converse" with a diagnostic system.

Graphics

All of the data acquisition systems discussed in section 1.1.2, and most of the speech recognition and generation systems discussed previously, display their information using graphical displays. The real innovative and impressive use of graphical displays is in medical imaging systems.

The interactive visualization of three dimensional cardiac images is described by Smith et al [Smith *et al.*, 1990]. This system, implemented on a Sun 3/280, allows the viewing, translation, rotation, and scaling of 3-D heart images in real time.

Zhang et al [Zhang *et al.*, 1990] present an impressive system for the viewing of 3-D brain images for surgery in this article. Simultaneous display of brain images from different imaging devices obtained in different orientations is possible. A multi-image environment allows the simultaneous display of magnetic resonance, computed tomography, digital subtraction angiography, and positron emission tomography images in multiple windows adjusted for common coordinates with reference to markers. Points and regions marked in one image can be dynamically projected to other images yielding exceptional advantages over previous imaging procedures.

Multimedia

Another user interaction technique which is gaining popularity in the medical domain is multimedia. Multimedia combines graphics, video pictures, and audio sound to display information. Issues in the design and application of multimedia interfaces are discussed by Laurel et al [Laurel *et al.*, 1990]. In medical applications, radiology has once again been the pioneer in multimedia applications.

Hickey et al [Hickey *et al.*, 1990] describe a multimedia radiological system which contains digitized radiographic images, voice reports, and other patient data which is supplemented by a network which allows the simultaneous sharing of this multimedia information among several users. Information can also be shared through what is known as a *shared visual workspace* allowing physicians to consult with each other accessing the same information even though they are in different physical locations.

Another similar system which communicates information between a radiology department and an emergency department is discussed by Goldberg et al [Goldberg *et al.*, 1989]. All the data, images, and graphical information is sent through the network from the radiology department to the attending physician in the emergency room. Voice data is relayed using twisted pair connections to telephone sets.

An ambitious integrated medical workstation, which also features speech input using speech recognition, is described by Kuhn et al [Kuhn *et al.*, 1990]. This workstation includes multimedia for on-line documentation of images using text, graphics, or digitized speech comments. This information is then made available to other departments where additional comments could be added. The images and comments gathered over the years could eventually be used in a multimedia teaching system.

1.2 Other Fields With Similar Issues

The issues of data acquisition and effective user interaction is definitely not particular to the field of medicine. In this section, some applications of data acquisition and user interaction methods used in other fields will be presented.

1.2.1 Data Acquisition

One area in which there is an urgent need to reduce overcrowding and operator saturation is air traffic control (ATC). New systems being developed today for ATC will see the implementation of expert systems which will assist in collision avoidance and bad weather landings [Perry and Adam, 1991]. Also, new ATC displays will help controller to deal with massive amounts of information as the traffic in the skies overhead increases. These new ATC systems, coupled with digital flight management systems on board the current generation of aircraft, will bring air traffic control in a new era of improved efficiency and safety.

ATC is just one example of where computers will play an ever increasing role in managing large amounts of data in real-time, and presenting this data to many users in a meaningful manner, with computers also playing a role in the interpretation of the acquired data.

1.2.2 User Interaction

Speech Interfaces

For the training of prospective air traffic controllers, systems exist which provide a speech interface with the trainee [Matrouf *et al.*, 1990] [Matrouf *et al.*, 1988] [Harrison *et al.*, 1986]. Speech generation systems simulate pilot dialogue and a speech recognition system translates a student air traffic controller's spoken commands into actions which are interpreted by the simulator.

The military have also taken interest in the use of speech interfaces for the purposes of decreasing pilot saturation due to the large amounts of visual information that must be processed. A study done by Moore and Moore [Moore and Moore, 1986] compare speech input and output versus other pilot input and output channels in order to show which applications interactive voice is best suited for.

Laporte [Laporte, 1989] describes a system which uses speech recognition and synthesis for the selection of radio frequencies in military aircraft. Results in this article show that tracking a target while using voice interaction for frequency selection is more precise than manual frequency selection method.

Miliatry systems which combine speech and graphics are the new airborne warning and control system terminals (AWACS) which are described by Salisbury et al [Salisbury *et al.*, 1990]. A previous study done by Salisbury and Chilcote of Boeing [Salisbury and Chilcote, 1989] investigated the feasibility of using speech in the AWACS systems with very encouraging results.

Other military studies have noted that speech interfaces are useful for many applications which can help reduce operator overload and decrease stress and fatigue from using the keyboard [Cupples and Beek, 1990].

Speech interfaces are especially useful for handicapped people for the operation of workstations [Clark, 1989] as well as for enabling the blind to interact with computers [Fellbaum, 1987].

To facilitate the operation of machinery in process control applications, speech recognition was used as a substitute to pressing buttons and shifting gears in a system described by Bitzer and Domer [Bitzer and Domer, 1989].

An application in which "hand-free" and "eyes-free" data entry was crucial was for meat inspection [Shozo, 1990]. In this application, speech recognition allow the meat inspector to record the inspection data while using both hands to treat the internals of cattle.

Speech recognition has also been used in robot control applications, an example of which is described by Foster and Bryson [Foster and Bryson, 1989]. This article describes an IBM 7535 industrial manufacturing robot which is controlled by voice. A PC/AT serves as an interface between the robot and the speech unit. A program running on the PC/AT serves to interpret words from the recognizer into a real-time sequence of movements for the robot.

Other Interaction Techniques

A recent article by Kitano [Kitano, 1991] presents new and interesting developments in naturai language interfaces. This article describes a system which intergrates natural language processing, speech recognition, and speech generation in a real-time Japanese to English translation system. The sentence parsing algorithm used predicts the next possible word in an input utterance thus enabling translation to begin before the end of a sentence.

New and innovative interaction techniques are always appearing. The one which directly relate to manual data entry problems are pen-based systems which use hand written character recognition. The issues involved in recognizing hand written characters in "notepad" computers is described in an article by Pittman [Pittman, 1991]. Another article by Helm et al [Helm *et al.*, 1991] describes issues involved in parsing "visual" language.

An ambitious project to define and present the architectural principles of a user interface for the 21st century is currently underway in Japan [Nonogaki and Ueda, 1991]. Fourteen companies are involved in this endeavor, called the FRIEND21 project, which is designed to bring about a technical breakthrough in human-computer interfaces to enable people to obtain, use, and send various kind of information necessary for daily activities, using personal computers or other computing machinery in the 21st century.

1.3 Thesis Overview

This dissertation presents a speech interface which was implemented for the "hands-free" and "eyes-free" entry of fluid balance data for an intensive care unit patient data management system.

The following chapter describes the current version of the patient data management system first starting with an overview of the system, followed by a presentation of the the hardware configuration and by a discussion of the individual software modules.

Chapter 3 gives a high-level description of the speech interface with a discussion of the hardware sub-systems followed by the functional description of the interface. Following this will be a description of the hardware and software architecture of the interface and a description of the data link communication protocol used to transmit messages between the patient data management system and the computer containing the speech interface systems.

Chapter 4 concentrates on the implementation issues of the speech interface focusing on the various software components which make up the speech interface. Sample sessions and experimental results obtained from a test set of eight users with different computer experience are included. A discussion on future extensions and improvements for the speech interface is presented before concluding in chapter 5.

Chapter 2 The Patient Data Management System

This chapter describes the Patient Data Management System (PDMS), an ongoing project developed at the McGill Research Centre for Intelligent Machines (McRCIM) in conjunction with the Pediatric Intensive Care Unit of the Montreal Children's Hospital. The first section of this chapter will give an overview of the scope and of the functionality of the current PDMS which is installed and running at the Montreal Children's Hospital. The subsequent section will outline the hardware configuration of the PDMS, focusing on the network which furnishes the PDMS with physiological patient data. Following the description of the hardware configuration will be the software configuration section, which will present a more comprehensive treatment of the individual software modules, and of the interaction between these different modules.

2.1 PDMS Overview

The development of the PDMS was inspired by the need for increased automation in the acquisition and management of patient physiological data, as the monitoring systems at the patient bedside were themselves being automated. These bedside monitoring systems generate a large volume of data electronically. While such amounts of data are useful to the nurses in the Intensive Care Unit, the manual charting of the electronically generated data would be unwieldy to say the least. Much of this data is lost as the nurses take readings of the physiological and fluid balance data, at the patient bedside, at half-hour or hourly intervals. These readings are then accumulated and plotted for the interpretation of a patient's state over the past few hours or days. One of the limitations of the manual charting techniques is that patient readings taken at these intervals may not accurately reflect the state of the patient between the readings. Another problem with the manual data acquisition techniques is the high probability of transcription, addition, and entry errors. In addition, if a patient requires resuscitation, then all data acquisition is usually suspended until the patient's condition stabilizes. Moreover, the manual charting process places an unnecessary burden on the nurses, who would prefer to spend more time on patient care than on the manual accumulation and plotting of patient data.

As an attempt to tackle the problems inherent in the manual acquisition process, the PDMS was conceived to handle patient data acquired automatically from the bedside physiological monitors, as well as data which cannot be acquired automatically and must be input manually. Data which cannot be acquired automatically by the PDMS includes the intravenous fluid intake and fluid output of a given patient in the intensive care unit, better know as fluid balance data. The function of the PDMS is to automatically monitor and record the patient data, to facilitate the review and interpretation of the data by presenting colour trends, plots, and charts on a screen display, and to assist in the hardcopy documentation by producing color printouts of the screens as well as producing the required end-of-shift reports. Since the majority of the data acquired by the PDMS originates from electronic monitoring equipment, the PDMS interfaces directly to the electronic monitoring system; continuously acquiring, logging, and archiving the physiological patient data. The main motivation of the system is to contribute to a higher standard of patient care by presenting data to the health care professionals which would otherwise be lost in the manual charting method between the charted intervals, in addition to removing the nurses from performing clerical tasks which can easily be done by computer, thus allowing them to spend more time on patient care.

Certain factors must to be taken into account when developing an computer application for a critical care environment such as an intensive care unit. The PDMS must not only be computationally and functionally powerful, but it must also be easy to use for people who are not computer literate. Most importantly, it must meet the needs and expectations of the intended end users, otherwise, it will not gain user acceptance or integration with existing hospital computing facilities.

The following objectives were laid out in the PDMS design and incorporated into the current version of the PDMS system:

- automatically collecting physiological data from the bedside monitors,
- facilitating the review of patient data,
- meeting the user's needs for a system which is both powerful and easy to use,
- provide for manual data input in formats which resemble the current paper forms which are familiar to the hospital staff,
- integrate with the existing hospital computer network,
- provide a backup mechanism and hardware redundancy in case of computer failure.

2.2 Hardware Configuration

The hardware configuration of the PDMS, shown in figure 2.1, is based on the Hewlett-Packard CareNet system. The CareNet system provides a local area network which links up HP78532A Physiological Monitor/Terminals located at the patient bedside, to a HP78581A Network Systems Communications Controller in a *star* network architecture. At one of the branches of the star sits the HP78580A Careport. This unit provides a programmable interface between the network controller and a host computer system. The function of Careport is to translate the proprietary network messages and signal formats to standard RS-232C messages which can be understood by the host computer. It also forwards requests for physiological data and other information, supplied to the bedside monitors, from the

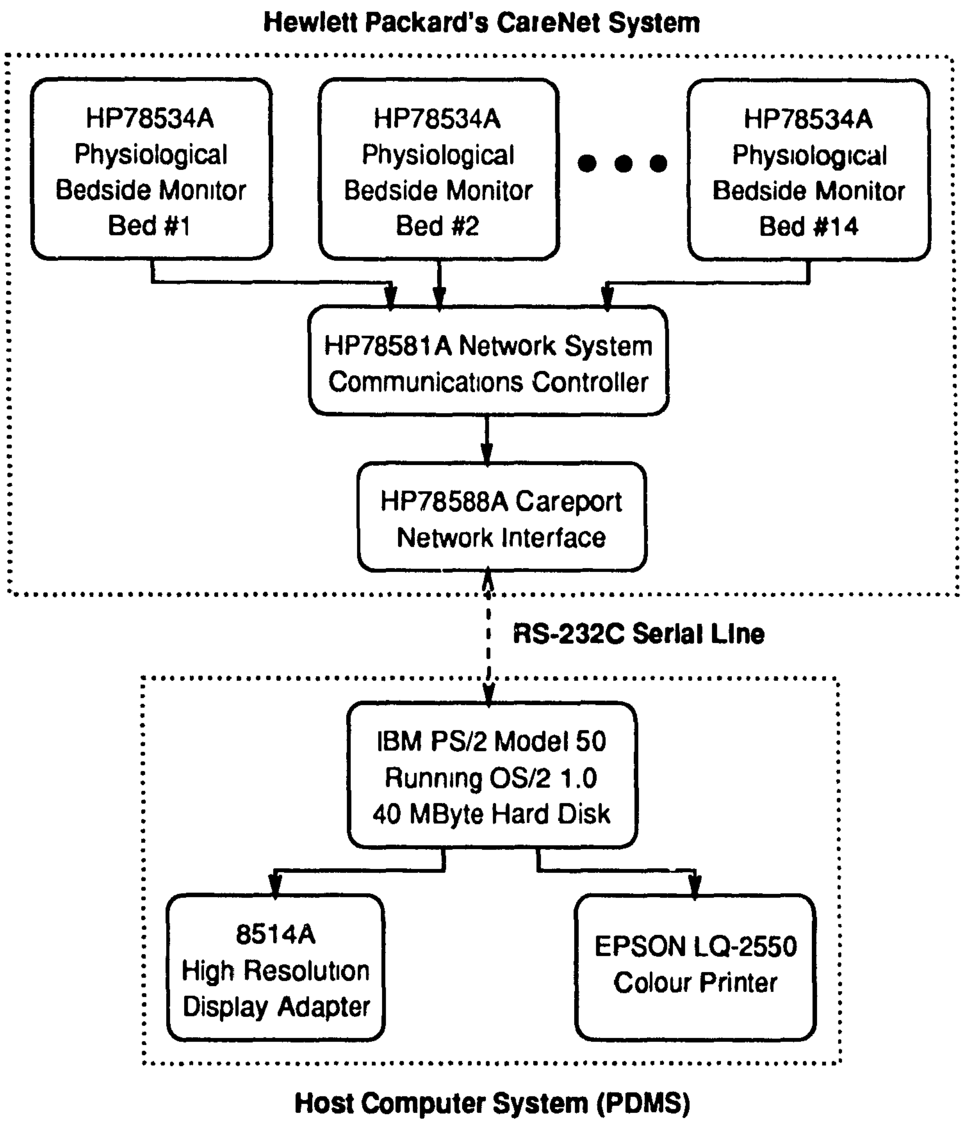

Hewlett Packard's CareNet System
HP78534A Physiological Bedside Monitor Bed #1
HP78534A Physiological Bedside Monitor Bed #2
HP78534A Physiological Bedside Monitor Bed #14
HP78581A Network System Communications Controller
HP78588A Careport Network Interface
RS-232C Serial Line
IBM PS/2 Model 50 Running OS/2 1.0 40 MByte Hard Disk
8514A High Resolution Display Adapter
EPSON LQ-2550 Colour Printer
Host Computer System (PDMS)

Figure 2.1: PDMS Hardware Configuration

host computer to the network. In this configuration, the host computer system is the PDMS.

Currently in the Pediatric Intensive Care Unit of the Montreal Children's Hospital, there are fourteen physiological bedside monitors. These monitors provide stand-alone data management capabilities which include the real-time display of measured parameters, automatic smoothing of the parameters, and the generation of alarms when either pre-set thresholds are exceeded or when certain critical patient conditions are detected.

The host computer system at the hospital presently consists of an IBM PS/2 Model 50 with 5 Megabytes of RAM memory, and a 40 Megabyte hard disk. The display consists of a high resolution 8514/A adapter which has a resolution of 1024 × 768 pixels. A colour printer, the EPSON LQ-2550, provides the colour printout of the PDMS screens in addition to printing out the required forms and reports for the nurses. This configuration will soon be upgraded with the addition of an IBM PS/2 Model 80 with 8 Megabytes of RAM memory, 300 Megabytes of hard disk space, a magneto-optical drive for data archiving and storage, and a Token-Ring network as a link between the two machines.

2.3 Software Configuration

This section describes the software architecture of the PDMS. After a brief outline of some operating system considerations, it then proceeds to describe the different PDMS software modules.

2.3.1 Operating System Considerations

Although the design of the PDMS has been in progress for quite some time, the previous design efforts were hampered by the lack of a real-time multi-tasking operating system for IBM personal computers. The release of OS/2 1.0 changed this. The benefits of a multi-tasking operating system enabled the PDMS to be developed as a series of independent, interacting modules to perform the required tasks. This was a clear improvement to designing the PDMS as a single-thread program to perform all of the necessary tasks. Interaction between the modules, which are run as separate processes, is defined in terms of operating system structures such as pipes, semaphores, and shared memory segments. With such a modular design, it then becomes straight forward to add more modules as the design and functionality of the PDMS increases and evolves.

2.3.2 PDMS Modules

The modules which are presently implemented and are part of the current version of the PDMS are:

- The **Manager Module:** which initiates all other modules, and acts as a main menu allowing the user to select other modules for interaction,
- The **Data Link Controller (DLC) Module:** which gathers information from the bedside monitor network through a RS-232C link to the Careport network interface. This module has exclusive command over the data acquisition, data storage, and the transmission of commands from other modules to the network. A typical command would be a request for parameters to be transmitted by Careport every *n* seconds when a patient is admitted into the ward, and added onto the PDMS. As this module does not require direct user interaction, it is run the background.
- The **Patient Registration Module:** which handles the admission and discharge of patients to and from the Pediatric Intensive Care Unit. This module also handles administrative patient information, such as the patient's name, date of birth, and hospital id number.
- The **Fluid Balance Module:** manages the data concerning the fluid intake and output of a given patient. The format of the fluid balance sheet is in a spreadsheet format, effectively emulating the "look and feel" of the paper forms used in the intensive care unit. All fluid balance data must be entered into the PDMS manually.
- The **Trend Display Module:** displays the data acquired from the bedside physiological monitors using graphical trends, facilitating the review of the

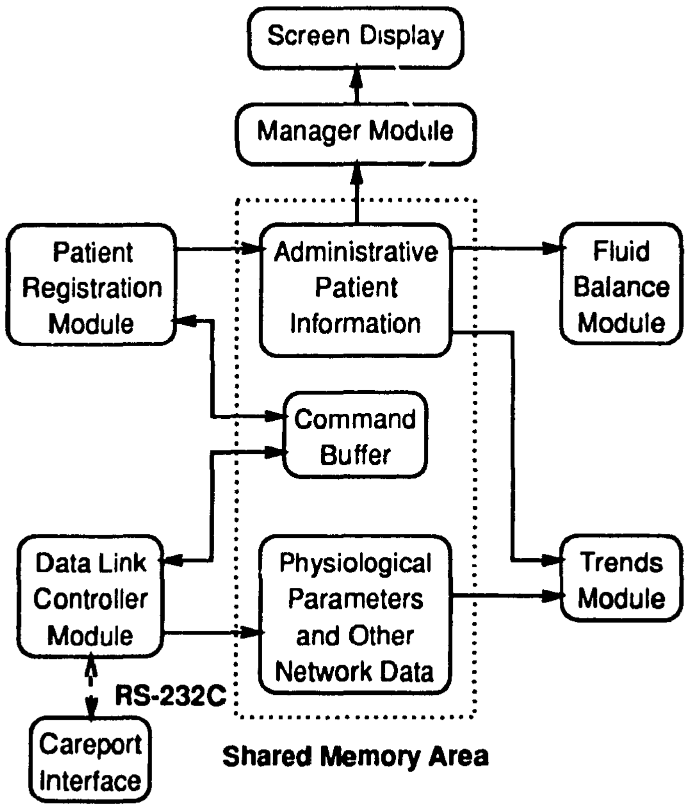

Screen Display
Manager Module
Patient Registration Module
Administrative Patient Information
Fluid Balance Module
Command Buffer
Data Link Controller Module
Physiological Parameters and Other Network Data
Trends Module
RS-232C
Careport Interface
Shared Memory Area

Figure 2.2: PDMS Software Configuration

patient's state by the health care professionals. The graphical trend scale can be selected to be one-half hour intervals, minute intervals, or second intervals.

The following subsections describe the PDMS software modules in more depth.

The overall organization and interaction between the PDMS software modules listed above is shown in figure 2.2. This organization was partly shaped by the limitation of version 1.0 of the OS/2 operating system. This first version of the operating system supported only text modes of operation, requiring the programmer to implement the graphics routines required in the application program. This changed with the release of OS/2 1.1, which introduced a graphical user interface environment in the form of the Presentation Manager.

OS/2 version 1.0 solved the problem of having multiple processes writing to the screen by separating processes into screen groups. A process can be switched in and out of the foreground. When a process is in the foreground, it has exclusive control over the keyboard, mouse, and the screen. The processes in the background place their screen output into *logical buffers* whose contents are displayed on the screen when the process is once again brought into the foreground.

The Manager Module

The Manager module is first PDMS module started, and it, in turn, starts up the other modules as different "child" screen groups. This module then displays the main menu of the PDMS, from which any one of the other modules, except the Data Link Controller module, can be selected. When a module is selected by the user, the Manager module proceeds to put the selected module in the foreground, allowing user interaction. By selecting the proper option in the other modules, one can return to the main menu screen. Having a modular Manager module allows new modules to be added simply by placing the new module in the list of processes that Manager needs to start, and by adding a new entry in the menu list for it.

2.3.3 The Data Link Controller

The Data Link Controller (DLC) module is responsible for interfacing the PDMS with the bedside monitor network through Careport. DLC stores the information acquired automatically by Careport for easy access by the other modules, and transmits commands received from the other PDMS modules into the proper format to Careport. To do this, DLC exploits OS/2 operating system features such as shared memory segments, pipes, and multiple threads.

The principal functions of DLC is to acquire the physiological parameter values from the bedside monitors every two seconds and to place them into circular queues. The *seconds* data is averaged every minute, and these values are placed in minute queues. These minute values are in turn averaged every one-half hour and placed in the one-half hour queues. All these queues are located in shared memory, so that these values can later be accessed by the Trends module for graphic display. Periodically, DLC archives data from the minute queues and half-hour queues to the hard drive for future use.

Information about the bed states on the network must also be shared with the Trends and Fluid Balance modules. This information is available to the aforementioned modules through shared memory segments. Access to these segments by the respective processes is controlled by semaphores.

Most of the DLC's activity deals with communicating with the Careport interface. This involves three levels of activity: a subset of the ANSI x3.28-1976 communication protocol which must be respected for physical message transfer, the encoding and decoding of messages from the logical message format from Careport, and the proper definition and manipulation of logical sources in order to obtain the desired network information. Admission, suspension, and discharge of patients on the network is done through the use of logical source definitions. These definitions are virtual connections between the different data sources which are designed to increase the efficiency of the serial communications line, in addition to determining the manner in which the information is passed to DLC from the network.

DLC automatically receives data sampled every two seconds from the Careport interface. In addition, alarm messages may be transmitted by Careport to DLC asynchronously, as they occur. The different parameter averaging functions as well as the reception and transmission of data are implemented as different threads in DLC. Communication of relevant information between threads is achieved through pipes.

The Patient Registration Module

This module manages the administrative patient information as well as the admission, suspension, and discharge of patients from the PDMS's representation of the intensive care unit ward. A general menu of commands is presented to the user corresponding to the chosen function. With this menu, the user can enter, modify, or review both patient and ward information. This module creates a shared memory structure for storing the administrative patient information, which is indexed by bed number, and is accessible by both the Trends and the Fluid Balance modules so that they can display the patient name with the respective patient data. This module flags events such as the addition, suspension, or discharge of a patient to DLC through the use of shared memory and semaphores.

The Trends and Fluid Balance Modules

The Trends and Fluid Balance modules constitute the most important parts of the PDMS, implementing the main functions for which the PDMS was created. The Trends module allows the review of the physiological parameter values of a particular patient with different time scale resolutions of seconds, minutes, or half hours.

The Fluid Balance module enables the entry, calculation, and correction of the volumes of all the fluid intake and output of a given patient. The spreadsheet format of the form, shown in figure 2.3, emulates as closely as possible the actual paper forms used in the intensive care unit by the nurses. As previously mentioned, the fluid balance data is still input manually in this module, hence despite computerization, the possibility of data entry errors still exists.

Keyboard bedside data entry for the Fluid Balance module data presents a bottleneck, since there is currently only one PDMS console and fourteen beds in the pediatric ICU. Ideally, the fluid balance data should be read by the nurses,

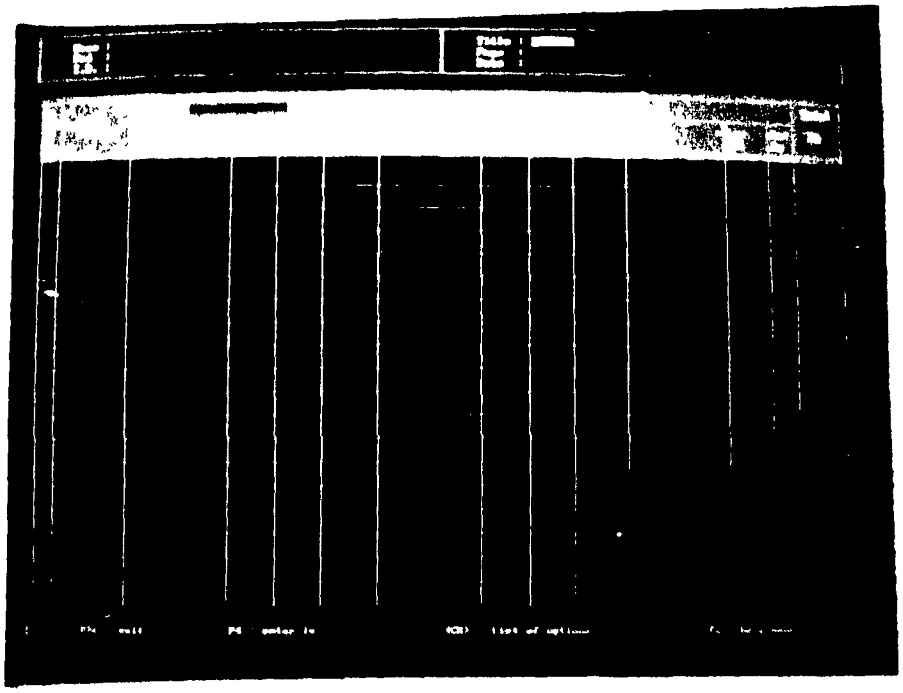


Figure 2.3: Fluid Balance Module Screen under OS/2 1.0

and entered directly into the PDMS. Having one console in the middle of the intensive care unit ward would require nurses to transcribe values read from the infusion pumps onto paper for entry into the PDMS at a later time. This process is unacceptable for two reasons. First, the possibility of transcription errors would still exist, as nurses would still be required to write down data on paper, and later type it into the PDMS. With this system, the nurses' workload would actually increase, and not decrease as desired. The ideal solution would be to have infusion pumps linked electronically to the PDMS, much in the same manner as the bedside monitors are linked through Careport. Presently, the ICU has a variety of infusion pumps, each of which provides a different interface to the pump, making this option unwieldy.

In order to effectively integrate the manual entry of data into the PDMS, while attempting to minimize the occurrence of entry and transcription errors, and facilitating human-computer interaction, data entry using a speech interface was proposed. The interface which would use both speech recognition, for data entry and module operation, and speech generation, for feedback, verification of spoken commands, and audio prompting, providing an "eyes-free" and "hands-free" means of directly entering fluid balance data at the bedside and at a distance from the PDMS console. Even if computers were introduced at the patient bedside, the speech interface would still be an integral part of the bedside data entry interface, since the intravenous solution infusion pumps can be situated at different locations around a patient's bed. A "hand-free" and "eyes-free" system would still permit the nurse to have extra mobility.

Another consideration in the Fluid Balance module of the current PDMS version is the quality of the user interface. As previously mentioned, the Fluid Balance module's interface effectively emulates the spreadsheet format of the manual Fluid Balance sheet. The current Fluid Balance module's Ingesta sheet, for recording fluid intake, and Excreta sheet, for recording fluid output, cannot be displayed completely on one 1024 × 768 pixel display. Consequently, cursor keys and other specially assigned key combinations, such as "ALT-scroll keys" and "CTRL-scroll keys" are used to navigate through the spreadsheet. The key assignments for this module differ from those used in other PDMS modules, presenting a very inconsistent user interface which could be quite confusing for a new user.'

In addition, the movement between the rows and columns on these sheets is quite slow. It typically requires one second to scroll in a new row or a new column into the screen display. Where fast, easy to operate data entry is required, the current version of this module may not be adequate.

Thus is was decided to implement the next version of the Fluid Balance module using the graphical interface of OS/2's Presentation Manager. All the drawing routines are handled by the operating system, and the graphical user interface is consistent among all applications which use Presentation Manager. The speech interface was then added to the module, adding another input and output modality. The modularity and high-level device independence of the OS/2 system function calls, allow alternate input modalities to added on in a relatively seamless manner. The description of the speech interface, as well as a discussion about the new Presentation Manager version of the Fluid Balance module are discussed in the following chapter.

Chapter 3 Speech Interface Description

This chapter focuses the implementation of the speech interface. The first two sections of the chapter are devoted respectively to the description of the OS/2 Presentation Manager environment and the speech recognition and generation sub-systems used in the speech interface, namely the Verbex Voice Conversational I/O sub-system and the Covox Voice Master Key II respectively. Following this, the hardware and software architecture of the interface, as well as its implementation, are presented in greater detail.

3.1 OS/2 and the Presentation Manager Environment

Developed by Microsoft and IBM as a successor to MS-DOS, OS/2 is an operating system for small computers based on Intel's 80286 and 80386 microprocessors. OS/2 uses the protected mode of these microprocessors which allows a large address space, virtual memory, and multitasking capabilities. In addition, OS/2 also offers operating system features such as dynamic linking, a device independent program interface, and a graphics interface. Interprocess communication, by means of pipes, semaphores, and dynamic data exchange (DDE), is one of the important features of OS/2 which allow for a greater level of program modularity. Under OS/2, the computing environment can consist of multiple programs which are loaded separately, but which can still work intimately together through interprocess communication. Semaphores also allow synchronization of events either between different, separate processes, or between different threads in the same program.

One of the more interesting and structured means of interprocess communica-

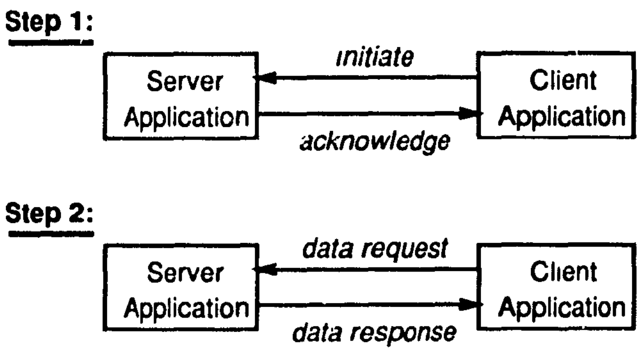


Figure 3.1: Dynamic Data Exchange Client Server Model (from [Southerton, 1989])

tion, which merits further discussion, is dynamic data exchange. Dynamic data exchange (DDE) allows two or more applications to share data in an automatic fashion [Southerton, 1989]. As far as the user is concerned, DDE happens invisibly between the applications, administering its own needs with no user interaction. From the programmer's point of view, dynamic data exchange means becoming familiar with a pre-defined protocol. Under DDE, protocol initiates, maintains, and terminates conversations between applications. The *client-server* model, shown in figure 3.1 is the basis for all dynamic data exchange transactions. In DDE, the client application always begins a conversation, often referred to as an *initiate*. After the server application *acknowledges* the initiate, the client application can make a data request. In turn, the server application sends the data to the client. It should be noted that an application can be both a client and a server, can have multiple servers, or can serve to multiple clients. The client can request the server to transmit data automatically as soon as it becomes available, and not have the client continuously query the server for data, thus decreasing overhead. The format used for the exchange of data can either be text, which is supported by Presentation Manager, or can be a programmer-defined format.

OS/2 also has a rich application programming interface (API). These API functions generally operate at a high level of performance, allowing the development of application program which avoid dependence on specific hardware features of a particular machine.

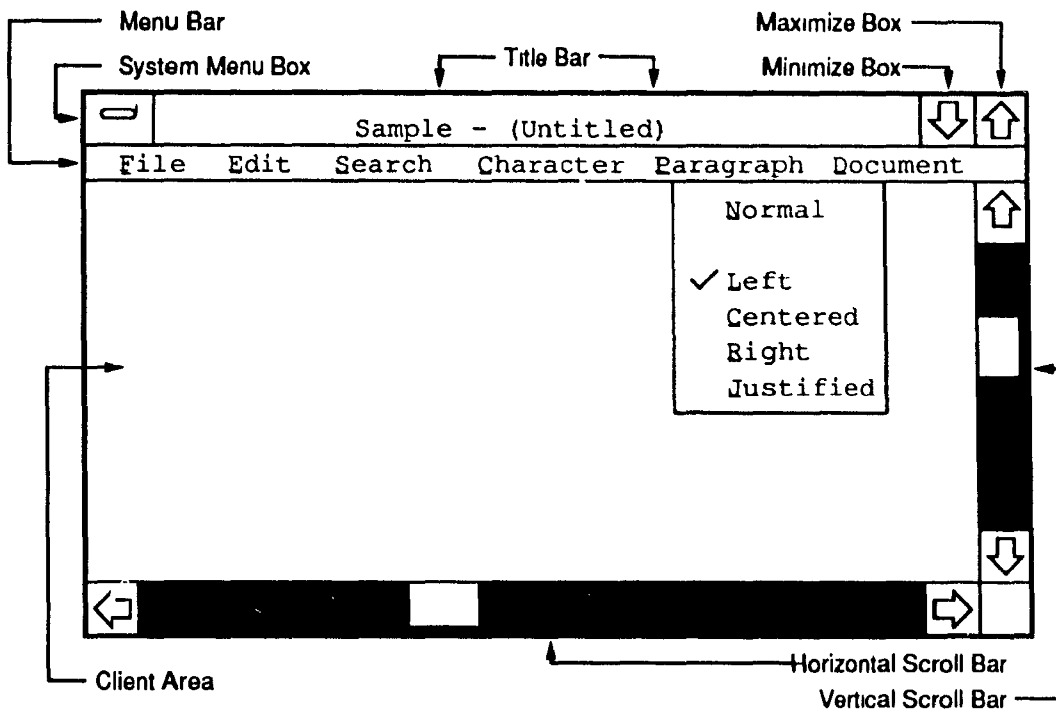


Figure 3.2: Presentation Manager Window Components

Another important element of the OS/2 operating system environment, is the graphical user interface (GUI) which displays individual applications in windows, and offers a rich set of device-independent graphics functions to application programs [Quedens and Beacons, 1990]. Figure 3.2 shows the components of a typical Presentation Manager window. A Presentation Manager window shows the list of available menu items for a particular application program in the *menu bar*. Some of these items in the menu bar display a list of menu items that drop down from the menu bar item. These items are called pull-down menus. The client area of a window is the part of the window in which the application has complete control over what is displayed there. The application can write text messages, or display graphics in this area.

Due to the modularity of the OS/2 operating system, such a graphical interface can be added seamlessly. Logic within the Presentation Manager assists the application in obtaining user input from various controls on the screen, such as menus, push buttons, scroll bars and dialog boxes. Since the menu and dialog box interface is built into the Presentation Manager, rather than into each individual application, the interface is consistent across all applications. This means that a user with experience with one Presentation Manager program can easily learn a new Presentation Manager program. With the other operating system functions, the Graphics Programming Interface (GPI) of the Presentation Manager are device independent. An application need not identify the input or output devices in order to use them effectively. It is this facet of the OS/2 Presentation Manager environment that makes the addition of a speech interface possible.

Aside from the important role that the Presentation Manager and the API plays in OS/2, they are also a part of IBM's System Application Architecture (SAA). SAA attempts to set user interface and API standards. If the goals of SAA come to pass, then the Presentation Manager user interface will become a common sight on the IBM microcomputer and mainframe terminals. More importantly for the program developer, it may one day be possible to write a PM program in a high-level language, and compile it to run on a variety of computers ranging from the IBM AT, to the IBM 370.

Most traditional operating systems provide a set of functions that a program can call for various system services. This is still the case with PM, but a PM program also obtains information from the operating system in a very different way: through messages. When either keys are pressed, the mouse is moved, or a mouse button is pressed, a program will receive either keyboard or mouse mouse messages that are sent to it by the Presentation Manager; a way that is similar to object oriented programming (OOP). Messages are also sent to inform a program when either a user has selected an item from a menu, when a program's window is resized, or when a program should repaint part of it's window. Therefore, PM programs can be considered to be "message-driven", as they remain dormant until a message is received [Petzold, 1989]. The messages *sent* or *posted* to a particular application by the operating system or another application program are placed in the application's message queue. Every application program which uses the graphical Presentation Manager user interface must create a message queue for the storage of messages to be processed.

Finally, OS/2 also provides network support, database support, and communication support. These parts of the operating system allows access to remote resources, such as databases or other programs, over a local area network (LAN). This is achieved by using Remote Data Services, Advanced Program-to-Program Communications (APPC) APIs, named pipes, semaphores, or DDE over the network. These APIs allow PM to communicate a variety of other operating systems over the network, such as OS400, VM, VMS, MUS, and AIX, which also subscribe to the SAA platform.

3.2 Speech Interface Sub-Systems

This section presents an overview of the Verbex Voice I/O and Covox Voice Master Key II systems which are used in the speech interface to provide speech recognition and generation, respectively.

3.2.1 Verbex Conversational Voice I/O System

The Verbex Voice Conversational I/O System is a computer peripheral which translates words spoken into a headset microphone into data that the attached host computer can understand and then sends this data to the computer. The host computer can then respond by sending data to the recognizer which is to be converted into speech by the recognizer's built-in speech synthesizer, and generated either through the recognizer's headset earphones, or through the system's audio output jack.

In order to perform recognition of spoken words or phrases, the recognizer requires a set of valid words to listen for and the order and patterns in which these words are spoken, also referred to as a *grammar*. Since the system recognizes continuous speech, the grammar definition can contain either individual words, or a collection of words concatenated together. Any statement which is not included in the grammar rules is considered to be improper or invalid and is ignored by the recognizer. As the system is speaker dependent, it also requires a given speaker's voice patterns to perform recognition for that particular user. With the a set of voice patterns, the recognizer links the sound of a human voice speaking each word in the grammar rules with the word itself. If a word is spoken as a part of a continuous, multi-word, phrase, the recognizer must also know how a word sounds in combination with other grammar-rule words.

Setting up an application for the Verbex recognizer is a five-step process [Verbex, 1990]. First, a text file containing the grammar definition specified in *Verbex Standard Notation* must be created. Appendix A contains the grammar file used for the main menu commands and the ingesta sheet commands of the fluid balance module in Verbex Standard Notation. A grammar file can also contain word *translations*, which are the character strings to be sent to the host computer when a valid statement, according to the grammar rules, is recognized. This file is then compiled into a "machine-readable" file which can be used by the recognizer for recognition and translation. The compiled recognizer file is then transferred from the host computer system to the recognizer's memory. The words and phrases defined in the grammar rules can then be trained by the user. After training, the user's voice templates for the grammar definition are then stored in a voice file. The grammar should be trained by all the users that wish to operate the system.

The text grammar file, in Verbex Standard Notation, can also contain multiple grammar definitions. The recognizer uses one grammar definition at a given time time for recognition purposes, and thus will only recognize statements contained in the grammar definition that it is currently using. When certain words or phrases are recognized, the recognizer can switch to other grammar definitions included in the file. The recognizer will then recognize statements specified in the new grammar definition only. The current grammar definition will continue to be used until a word or phrase in the grammar definition will cause it to switch to another grammar definition. This allows a large, complex vocabulary, which can be logically divided according to context dependence, to be entered as a sequence of grammar definitions. For example, a data entry process with many words in its grammar can be divided into a collection of smaller grammar definitions; one for each step in the data entry process. Such logical divisions can not only reduce search time and decrease individual grammar definition complexity, but can also increase recognition accuracy, by excluding words which are not relevant in a given context.

3.2.2 Covox Voice Master Key II

The Covox Voice Master Key II is a speech and music input and output computer I/O peripheral which connects to a host computer's parallel port [Covox, 1990a]. Speech can be sampled by the system at a maximum rate of 20 Khz with a representation of 8 bits per sample and saved on the host computer's memory. The digitized speech can then be edited, saved on the host computer's storage media, or played back either on headset earphones plugged into the system, or on the system's external speaker. For the digitized speech files, compression routines are supplied in order to save storage space, at the expense of some degradation in quality, using either the 2, 3, or 4 bit adaptive differential pulse code modulation (ADPCM) speech coding scheme.

The Voice Master Key II system is also capable of performing some simple speech recognition. It is limited to a speaker dependent 64-word vocabulary which does not support continuous speech. These are serious limitations for the speech interface application, which is one of the reasons why the Verbex system was chosen for the speech recognition component of the interface. The Voice Master

Key II also provides some routines which generate synthetic speech, according to English phonetical rules, from an ASCII text file.

In addition to supplying various software utilities for speech capture, compression, and playback, viewing and editing of the captured digitized speech signal is also possible. This feature is useful for deleting the leading or trailing silence samples in digitized words. This allows the concatenation of words to sound more natural without the extra intervals of silence between successive words.

Lastly, but more importantly for the speech interface application, a library of "C" language functions is provided, enabling customized application development through the use of these special routines. This permits the playback of digitized words or phrases to be played back on the system when certain events occur in the application program. The playback "vocabulary", which is collection of words or phrase files to be played back by an application program, can be defined in a *word list* [Covox, 1990b].

A word list is a set of digitized speech files which are linked together in one large file; created using a speech utility editing function. The utility will also automatically create a sound file index for the word list. In the application program, instead of loading the digitized speech files one at a time from disk, the word list is loaded into memory at the beginning of the program. From then on, to "say" an individual word or phrase, one only needs to refer to the proper index number in the word list. One drawback of the word list is that the digitized speech files defined in the word list are loaded directly in the host computer's RAM memory for fast playback. If the number of words or phrases in a given application is considerable, then the use of a word list might prove to be impractical; requiring too much host memory to store the files. Alternatively, the speech files can also be called and played back on the system referenced according to their respective file names.

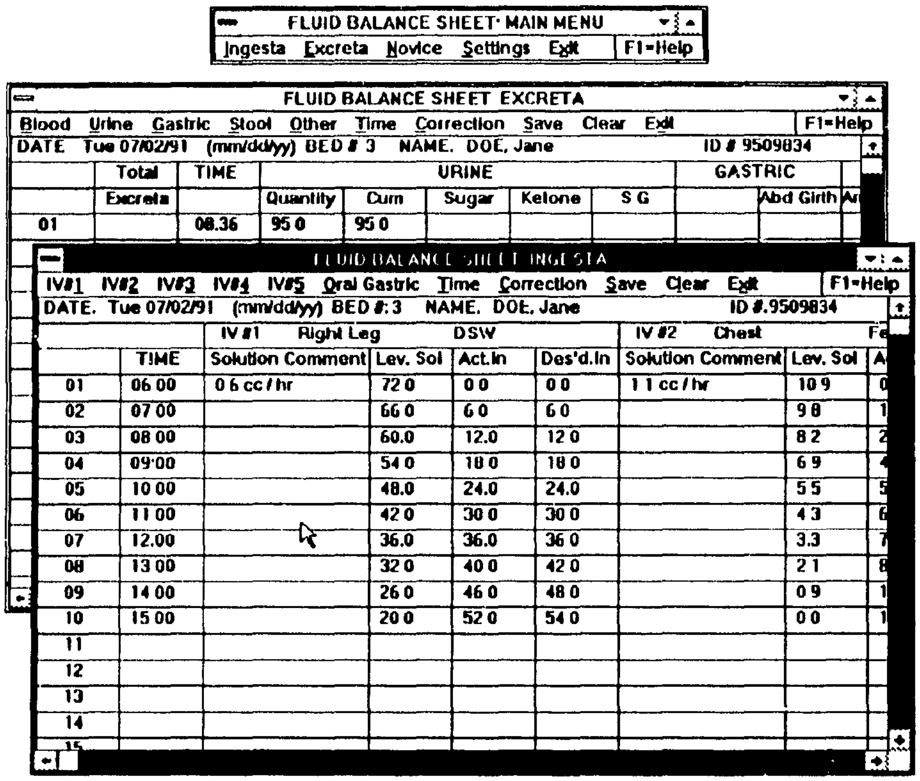
3.3 A Speech Interface

The following sections present the global organization of the speech interface, including as both the hardware and software architecture. This section first begins with a functional description of the new version of the Fluid Balance module which was implemented under the graphical environment of OS/2's Presentation Manager. Then the speech interface to this module will be described.The following section will then present the hardware architecture of the speech interface, with a description of the systems used in the interface outlining features specific to the respective systems which are relevant to the implementation of the interface. The last section will describe the software architecture of the speech interface highlighting the various software components and the communication methods used between the components.

3.3.1 Functional Description

Figure 3.3 shows a typical display of the Fluid Balance module implemented under the Presentation Manager environment of OS/2. As previously mentioned, the fluid balance module records all of a patient's fluid intake and output in the *ingesta* and *excreta* sheets, respectively. Readings of the various fluids, ranging from the intake of intravenous solutions to the output of blood and urine, are taken at either half hour or hourly intervals. These readings are accumulated over a given period, typically a day, in order to determine the total fluid intake and output of the individual solutions. The speech interface is meant to allow "hands-free" and "eyes-free" data entry of the fluid balance level values, in addition to providing a more natural and familiar means of entering data by the use of voice.

The fluid balance module operates in a fully graphical, keyboard and mouse-driven environment which is now augmented by another input modality, namely, the speech interface. The options available to the user are displayed at the top of a


FLUID BALANCE SHEET· MAIN MENU
Ingesta Excreta Novice Settings Exit F1=Help
FLUID BALANCE SHEET EXCRETA
Blood Urine Gastric Stool Other Time Correction Save Clear Exit F1=Help
FLUID BALANCE SHEET INGESTA
IV#1 IV#2 IV#3 IV#4 IV#5 Oral Gastric Time Correction Save Clear Exit F1=Help
DOS

DATE Tue 07/02/91 (mm/dd/yy) BED # 3 NAME. DOE, Jane ID # 9509834

	Total	TIME	URINE					GASTRIC		
	Excreta		Quantity	Cum	Sugar	Ketone	S G		Abd Girth	Ar
01		08.36	95 0	95 0						

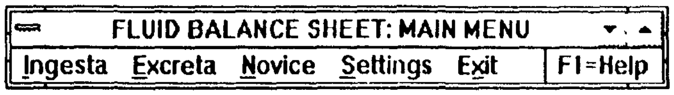
DATE. Tue 07/02/91 (mm/dd/yy) BED #:3 NAME. DOE, Jane ID #.9509834

		IV #1 Right Leg		DSW		IV #2 Chest		Fe
	TIME	Solution Comment	Lev. Sol	Act.In	Des'd.In	Solution Comment	Lev. Sol	A
01	06 00	0 6 cc / hr	72 0	0 0	0 0	1 1 cc / hr	10 9	0
02	07 00		66 0	6 0	6 0		9 8	1
03	08 00		60.0	12.0	12 0		8 2	2
04	09·00		54 0	18 0	18 0		6 9	4
05	10 00		48.0	24.0	24.0		5 5	5
06	11 00		42 0	30 0	30 0		4 3	6
07	12.00		36.0	36.0	36 0		3.3	7
08	13 00		32 0	40 0	42 0		2 1	8
09	14 00		26 0	46 0	48 0		0 9	1
10	15 00		20 0	52 0	54 0		0 0	1
11								
12								
13								
14								
15								

Figure 3.3: Fluid Balance Module Screen under the Presentation Manager Environment of OS/2

given window in the menu bar. These commands can either, display sub-menus, create other windows, or create dialog boxes for obtaining user input. Figure 3.4 shows the main window of the fluid balance module, which is first displayed when the module is first run. The menu bar of this window offers the following options:

- **Ingesta:** which creates an instance of the Ingesta sheet window for recording the fluid intake of a patient.
- **Excreta:** which creates an instance of the Excreta sheet window for recording the fluid output of a patient.
- **Novice:** which allows the user to indicate to the module whether or not s/he is a novice user. If the user is a novice, the messages echoed back on the user's headset are more detailed and descriptive than for a non-novice user which



FLUID BALANCE SHEET: MAIN MENU

Ingesta Excreta Novice Settings Exit | F1=Help

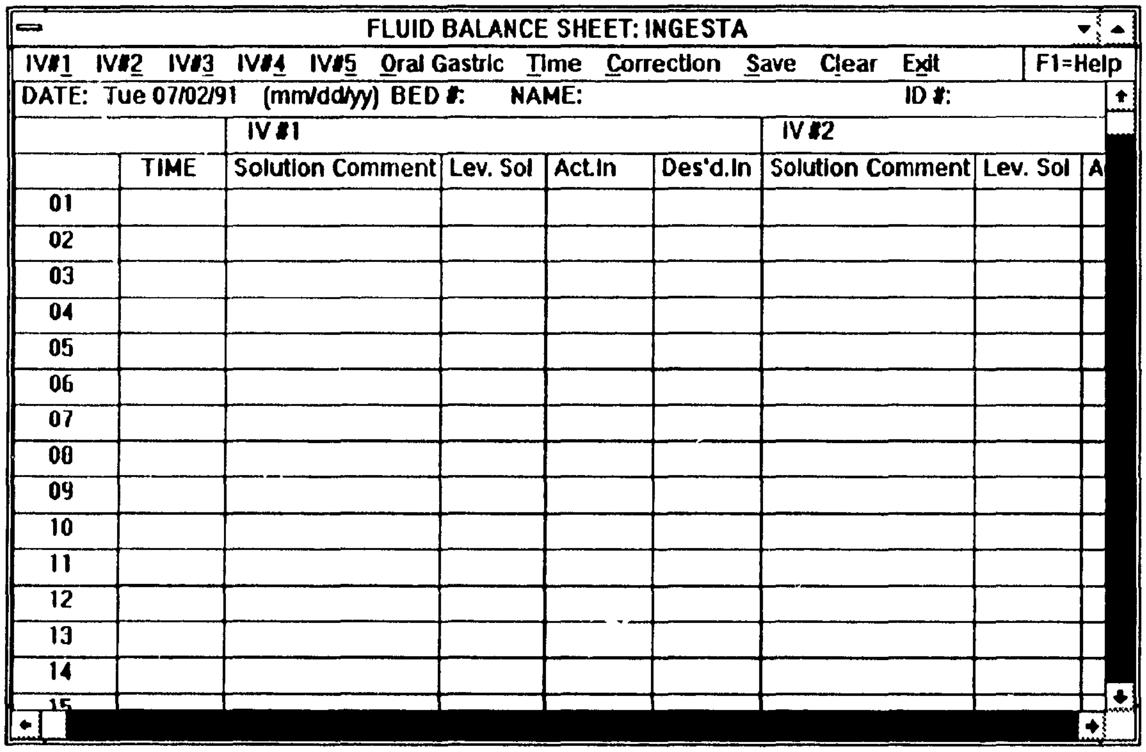
Figure 3.4: Main Window of the Fluid Balance Module Screen

is more familiar with the system.

- **Settings:** which allows the user to select the patient's name, hospital ID number, or bed number whose data is to be recalled or entered.
- **Help:** which displays a context sensitive help menu.
- **Quit** which terminates the fluid balance module destroying all windows created by the module during its execution.

A typical ingesta sheet window is shown in figure 3.5. The options available to the user for this window are:

- **IV1** to **IV5** items which allow the recording of five different intravenous solution levels, and the times at which the readings were taken.
- **Oral Gastric:** which allows the recording of individual oral gastric fluids according to their type.
- **Time** item allows the time in the "time" column of the ingesta sheet to be set or corrected.
- **Correction:** which allows the user to correct any of the data which has been input in the sheet according to the solution number and line number.
- **Save:** which saves the ingesta sheet data into a file.
- **Clear:** which clears the ingesta sheet of all input data. This option is presently used for debugging purposes only and should be eliminated from versions which will be installed at the hospital.
- **Exit:** which exits the ingesta sheet destroying the window.



FLUID BALANCE SHEET: INGESTA

IV#1 IV#2 IV#3 IV#4 IV#5 Oral Gastric Time Correction Save Clear Exit F1=Help

DATE: Tue 07/02/91 (mm/dd/yy) BED #: NAME: ID #:

		IV #1				IV #2		
	TIME	Solution Comment	Lev. Sol	Act.In	Des'd.In	Solution Comment	Lev. Sol	A
01								
02								
03								
04								
05								
06								
07								
08								
09								
10								
11								
12								
13								
14								

Figure 3.5: Ingesta Sheet Window of the Fluid Balance Module

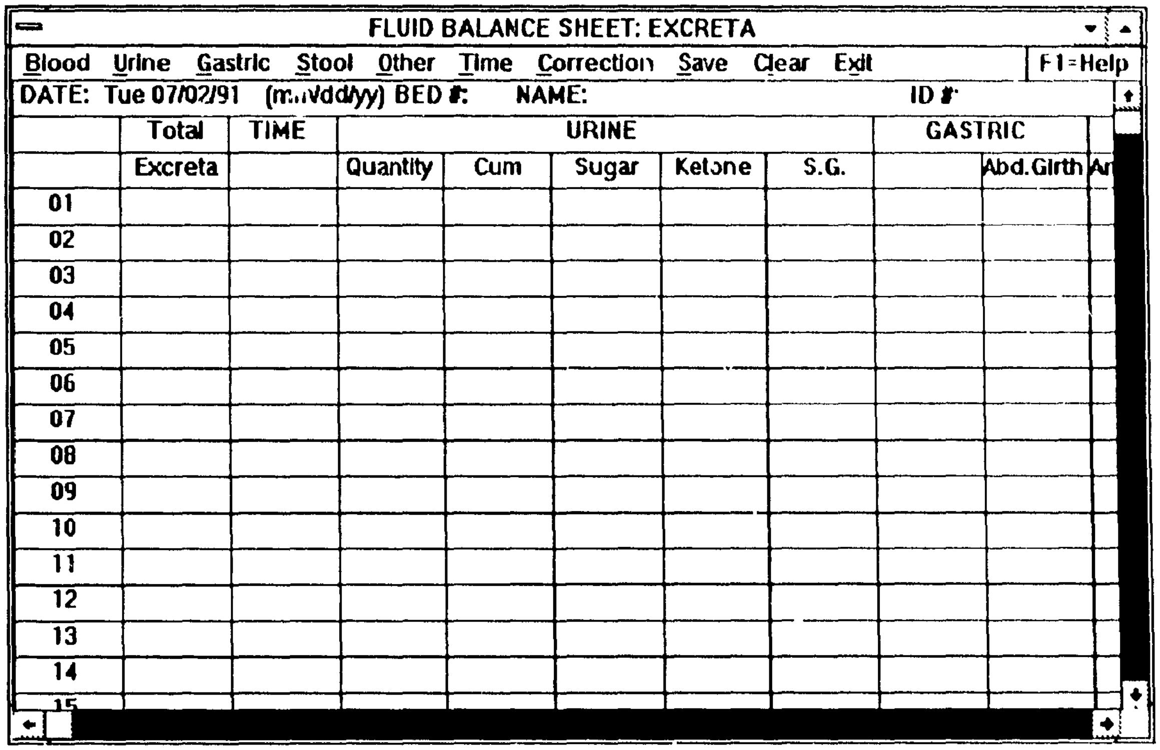
- **Help:** which creates an interactive help screen.

The other window created by the Fluid Balance module is the excreta sheet window, which is shown in figure 3.6. The menu bar options for this window are:

- **Blood, Urine, Gastric, Stool,** and **Other:** which allow the recording of output for these items, and also indicate the times at which these readings were entered.

- **Time, Correction, Save, Clear, Exit,** and **Help** options which perform tasks identical to those in the ingesta sheet described above.

Currently, there is no speech recognition or speech generation hardware or software which runs under the OS/2 operating system environment. All available systems on the market today run under the DOS operating system. This constraint influences to a large degree the hardware and software architecture of the speech interface which are described in the two following sections.


FLUID BALANCE SHEET: EXCRETA
Blood Urine Gastric Stool Other Time Correction Save Clear Exit
F1=Help

DATE: Tue 07/02/91 (mm/dd/yy) BED #: NAME: ID #:

	Total	TIME	URINE					GASTRIC		
	Excreta		Quantity	Cum	Sugar	Ketone	S.G.		Abd.Girth	Ar
01										
02										
03										
04										
05										
06										
07										
08										
09										
10										
11										
12										
13										
14										

Figure 3.6: Excreta Sheet Window of the Fluid Balance Module

3.3.2 Hardware Architecture

Figure 3.7 shows the hardware architecture of the speech interface used in our laboratory. Two personal computers are used: an IBM PS/2 Model 80, running OS/2, which is the operating system platform required for the PDMS software, and an Intel System 120, running DOS, which is the operating system platform required by both the speech recognition and generation hardware peripherals and software utilities. The two computers are linked by a serial RS-232C cable for the transmission of data between the two systems. The PS/2 model 80 also has a high resolution 1024 x 768 pixel 8514/A adapter card for the display of the Presentation Manager graphics.

On the Intel System 120, the speech recognition hardware, the Verbex Series 6000 Conversational Voice I/O system interfaces to the host by means of a 16-bit expansion slot. The Verbex system can output the translation of the recognized word or phrase either on its serial port or through the 16-bit slot at address 200 hex.

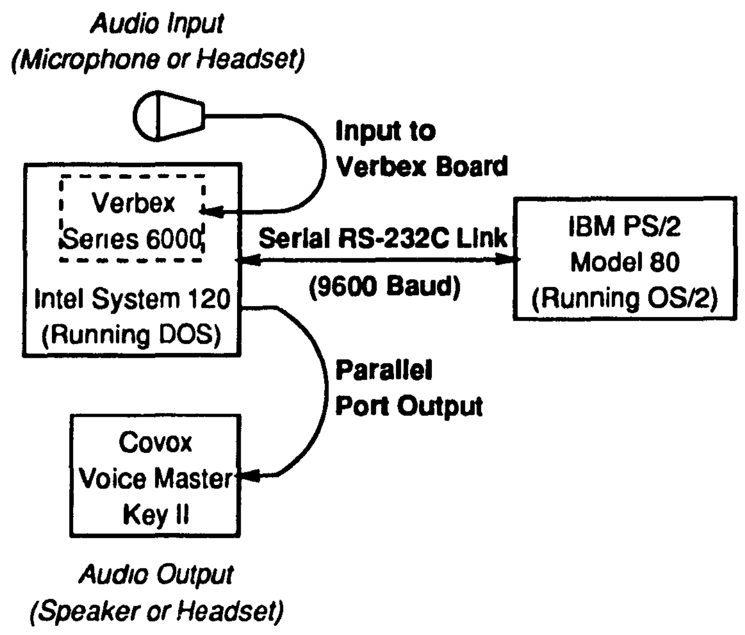


Figure 3.7: Hardware Architecture of the Speech Interface

Verbex also provides a terminate-and-stay resident (TSR) program which redirects the output of the board at 200 hex to the host computer's keyboard buffer, allowing the Verbex board to effectively replace the computer's keyboard. In addition, the Verbex board performs all of it's recognition on the board, using none of the host computer's resources. For the recognition process, the board uses an Intel 80286 microprocessor with 2 megabytes of RAM and 256 Kilobytes of FPROM as a control processor to coordinate information transfer between the host and the board. The recognition engine of the Verbex consists of a Texas Instruments TMS320C30 floating-point digital signal processing (DSP) chip with 512K of 32-bit words of high-speed static RAM. The board uses a 15-pin VGA connector for the audio input and output connection.

Also attached to the Intel System 120 is the Covox Voice Master Key II. This system interfaces to the host computer system sending and receiving messages through the parallel printer port. The Voice Master Key II has an audio input jack for the capture and digitization of sound, and has both an audio jack and an external speaker for audio output.

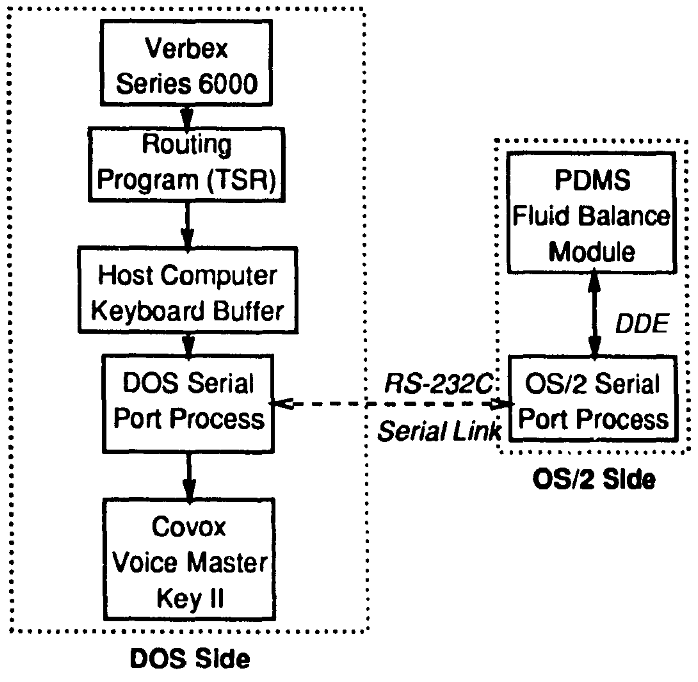


Figure 3.8: Software Architecture of the Speech Interface

3.3.3 Software Architecture

Due to the hardware architecture outlined in the previous sub-section, several interacting modules are required to ensure proper and reliable transfer of messages between the DOS and OS/2 systems. Figure 3.8 shows the general software architecture and the means of communication between the respective programs running on the both systems. In order to allow reliable transmission of voice commands from the DOS side to the OS/2 side, and of voice responses from the OS/2 side to the DOS side, a modified version of the Intel Hex protocol was used. This packet structure, as well as the data link control protocol will be treated in greater depth in section 3.3.4. This requires programs on both systems to be able to package the data to be transmitted to the remote system in this format, and to be able to receive, decode, and route the information received to the appropriate process.

In the DOS operating system environment, the absence of multitasking capabilities requires that the packaging and transmission of data, as well as the reception, decoding, and routing functions all be performed by the same application program.

However, on the OS/2 side different processes can handle separate tasks. In this operating system environment, a dedicated process is run to handle the serial port, and to communicate the decoded data to the appropriate process using dynamic data exchange (DDE). In order to establish the DDE links between the appropriate interacting modules, the programs must be executed in a specific order. Figure 3.9 shows the overall flow diagram between the various program modules on both systems and between both systems.

First, before any of the programs can be executed on either system, the user must have trained the grammar on the DOS side. Alternatively, the user can transfer his voice templates with the compiled grammar file from the host computer to the Verbex board. In order to emulate the keyboard pressed on the host computer, a terminate-and-stay-resident (TSR) program is loaded which redirects the output translations of recognized words or phrases from the recognizer to the host computer's keyboard buffer. The translation data can subsequently be read from the buffer by the application program through the use of any one of the standard "C" language function calls.

When the DOS communication program is first run, it installs an interrupt service routine for the serial port, enabling it to operate at a baud rate of 9600 without loss of characters. A high baud rate is desirable for fast data transmission between the two systems. Since the serial port is not an interrupt-driven device under the DOS operating system, a user-defined interrupt service routine was implemented. This routine stores the received characters in a circular buffer asynchronously from the normal flow of execution of the main program. Once the interrupt service routine is installed, the DOS communication program waits for a NULL initialization packet from the remote OS/2 system. The DOS communication program proceeds in its main processing loop once the packet has been received and acknowledged, indicating the presence on the OS/2 side of a program to receive the transmitted information.

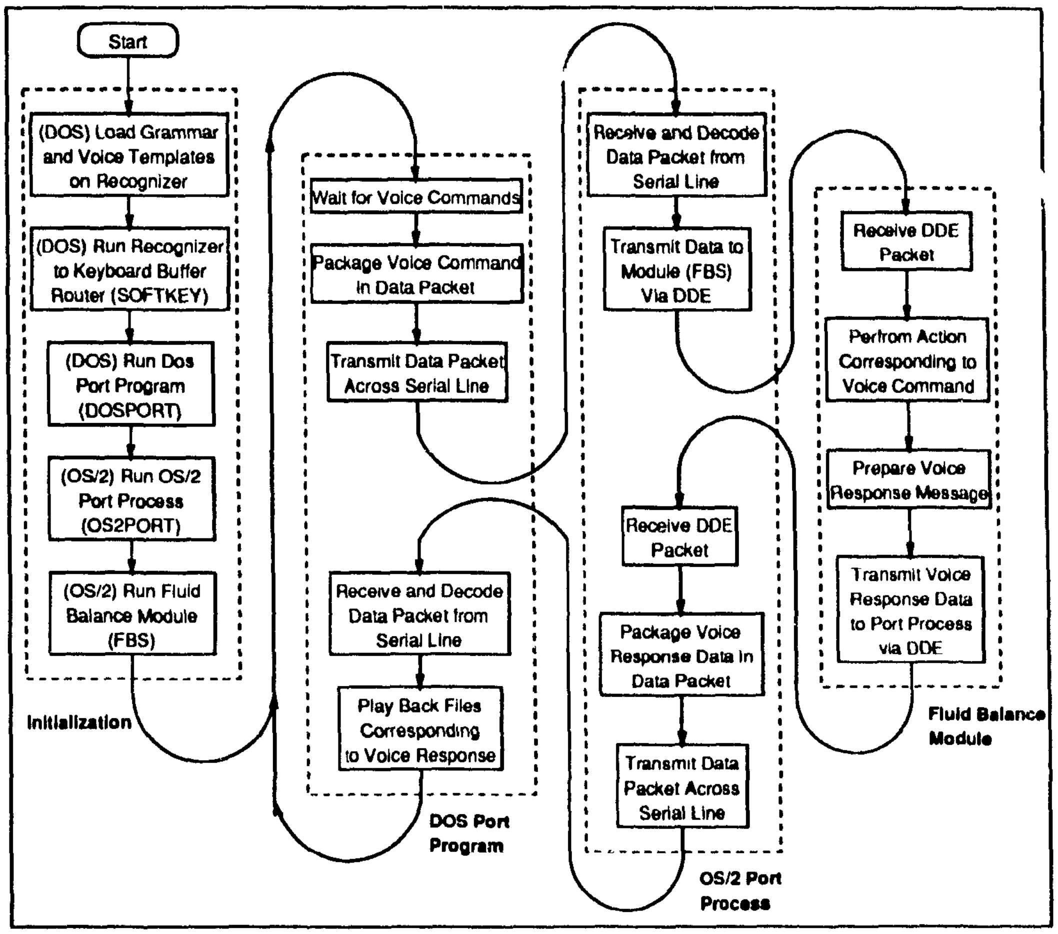

Start
(DOS) Load Grammar and Voice Templates on Recognizer
(DOS) Run Recognizer to Keyboard Buffer Router (SOFTKEY)
(DOS) Run Dos Port Program (DOSPORT)
(OS/2) Run OS/2 Port Process (OS2PORT)
(OS/2) Run Fluid Balance Module (FBS)
Initialization
Wait for Voice Commands
Package Voice Command in Data Packet
Transmit Data Packet Across Serial Line
Receive and Decode Data Packet from Serial Line
Play Back Files Corresponding to Voice Response
DOS Port Program
Receive and Decode Data Packet from Serial Line
Transmit Data to Module (FBS) Via DDE
Receive DDE Packet
Package Voice Response Data in Data Packet
Transmit Data Packet Across Serial Line
OS/2 Port Process
Receive DDE Packet
Perfrom Action Corresponding to Voice Command
Prepare Voice Response Message
Transmit Voice Response Data to Port Process via DDE
Fluid Balance Module

Figure 3.9: Program Flow Diagram of the Speech Interface

The main processing loop on the DOS side performs the following sequentially:

- obtains, from the host computer's keyboard buffer, the translation of the recognized spoken word or phrase from the recognizer,
- package the data with the appropriate header and tailer information,
- transmit the data across the serial RS-232C link to the remote OS/2 system,
- wait for the voice response packet from the remote OS/2 system,
- receive and decode the data packet from the remote OS/2 system,
- acknowledge proper receipt of packet,

- play back the word, or sequence of words as indicated in the data packet and as specified by the Fluid Balance module.

This process continues on the DOS side until the user terminates the program's operation.

On the OS/2 side, certain features of the operating system allow a more modular program architecture for this component of the speech interface. Consequently, the process of reading from and writing to the serial port is an entirely different and separate program from the Fluid Balance module, the program which sends and receives data from the serial port.

The advantages of separating the serial port handling process from the Fluid Balance module are three-fold. First, the Fluid Balance module need not be burdened with the additional task of processing the serial port directly. Secondly, if there exists more than one process which expects data from the serial port, then there is no way of ensuring that a given process will get the data that was sent to it. With many processes reading the serial port, it is possible that a process mistakingly reads the data intended for another process. Lastly, as more PDMS modules have speech added as another input modality, a dedicated port process will be a necessity; facilitating the routing of information to and from the serial port.

Consequently, in a multi-tasking environment, when potentially more than one process expects data from a given serial port, a dedicated process should handle the serial port input and output, and communicating to the other processes via interprocess communication.

In the speech interface, the OS/2 serial port process begins by first initializing the serial port with the proper baud rate, word size, parity, and stop bits. Next, it send out a NULL initialization packet over the RS-232C serial line to the remote DOS system and waits for an acknowledgement from the DOS serial port program. If the OS/2 serial port process does not receive an acknowledgement from the remote system within 15 seconds of having transmitted the packet, the user is notified of the condition, and can either re-try the connection between the two programs, or quit the program and fix the problem.

After the initialization packet is acknowledged from the remote DOS port program, the OS/2 serial port handling process enters its main processing loop, which performs the following operations sequentially:

- wait for a packet containing voice commands from the remote DOS system,
- receive and decode the data packet,
- acknowledge receipt of packet,
- transmit the data decoded from the packet to the appropriate PDMS module,
- wait for voice response data from the PDMS module which received the voice commands,
- package the voice response data with proper header and tailer information,
- transmit packet across the serial link to the remote DOS system.

This process continues until the user exits the OS/2 serial port handling process.

Interprocess communication between the dedicated OS/2 serial port process and the Fluid Balance module is achieved by using the OS/2 component called dynamic data exchange (DDE). These DDE messages that are sent to the Fluid Balance Module from the OS/2 serial port process, are posted in the message queue of the Fluid Balance module by the serial port process. These messages are then handled by the message handling procedure of the Fluid Balance module. Voice response data messages are then sent from the Fluid Balance module to the OS/2 serial port process also using DDE. In DDE terminology, the OS/2 port process is the *server* to the Fluid Balance module *client* for the voice command data, and the Fluid Balance module is the *server* to the OS/2 port process *client* for the voice

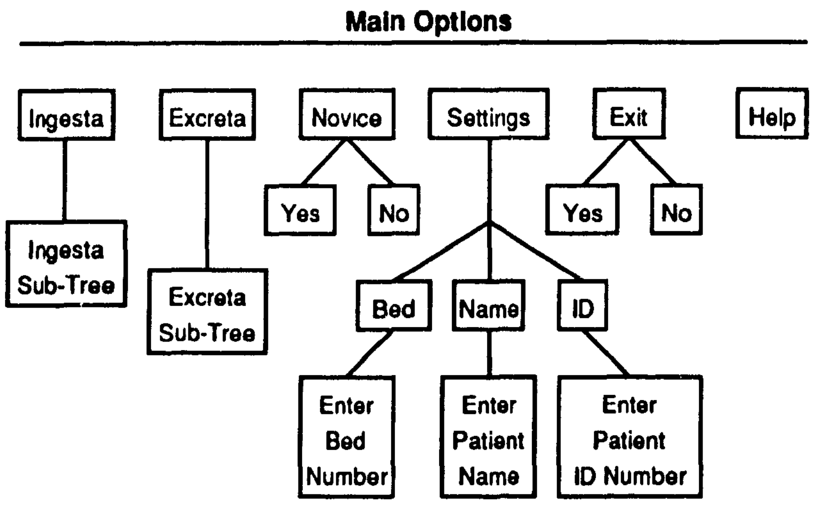


Figure 3.10: Fluid Balance Module Menu Tree (Main Menu)

response data. Hence, both the Fluid Balance module and the OS/2 port process behave as *both* client and server.

Once the Fluid Balance module processes the DDE data message, it proceeds to call the appropriate sequence of Presentation Manager functions which emulate either menu pull-downs, menu clearing, dialog box creating, entry field filling, or push button pressing, depending on the context of the current transaction. As far as the system is concerned, these events are identical to mouse clicks and to the pressing a sequence of keys on the keyboard. Such high-level operating system function allow non-mouse and non-keyboard actions to trigger events which are identical to mouse clicks or keyboard presses.

In the Fluid Balance module, state variables are used by the speech interface to keep track of where user is in the menu tree. Figures 3.10, 3.11, and 3.12 shows the menu tree for the main window, the Ingesta window, and the Excreta window respectively. As the user voices options in the menus, the menu tree is traversed. State variables ensure that the user will traverse the tree in syntactically legal, procedural fashion, consistent with the menu hierarchy inherent in the menu structure of the respective Fluid Balance module windows. In addition, the organization of the grammar definitions in the Fluid Balance grammar loaded on the recognizer,

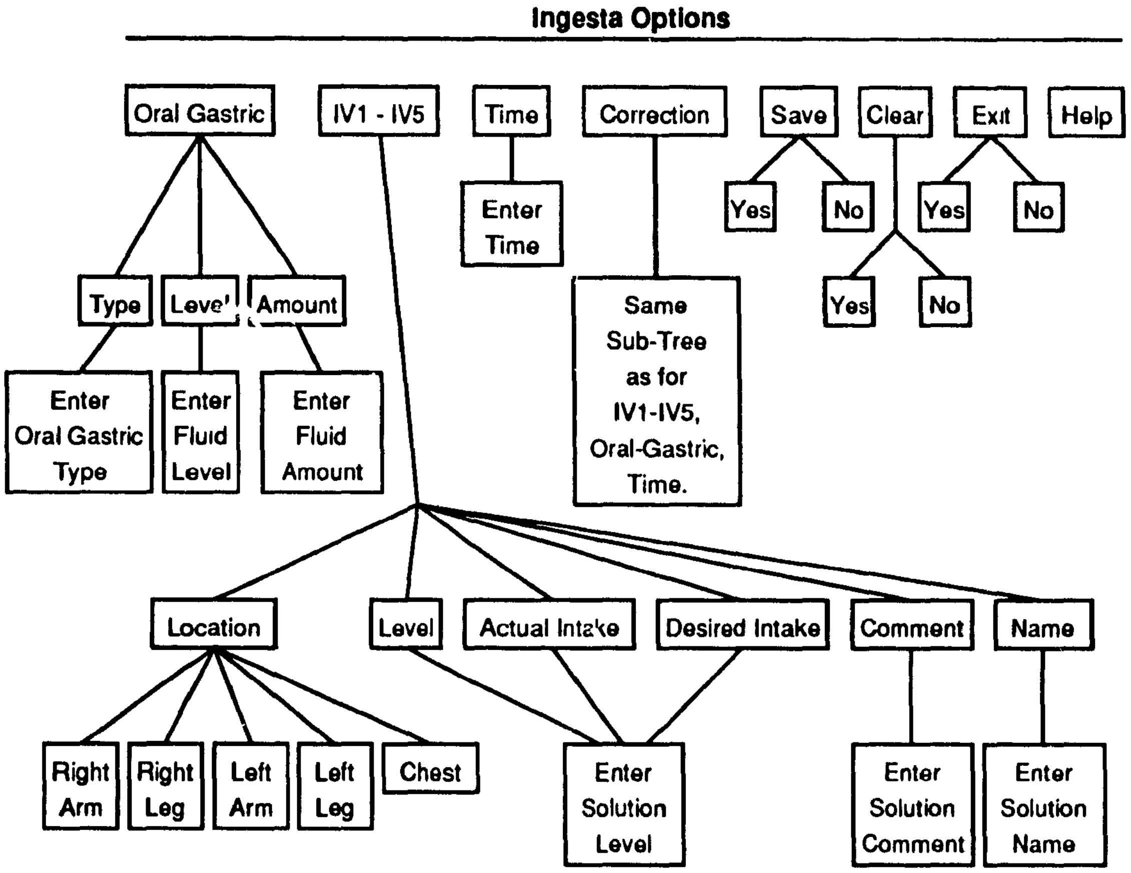


Figure 3.11: Fluid Balance Module Menu Tree (Ingesta Menu)

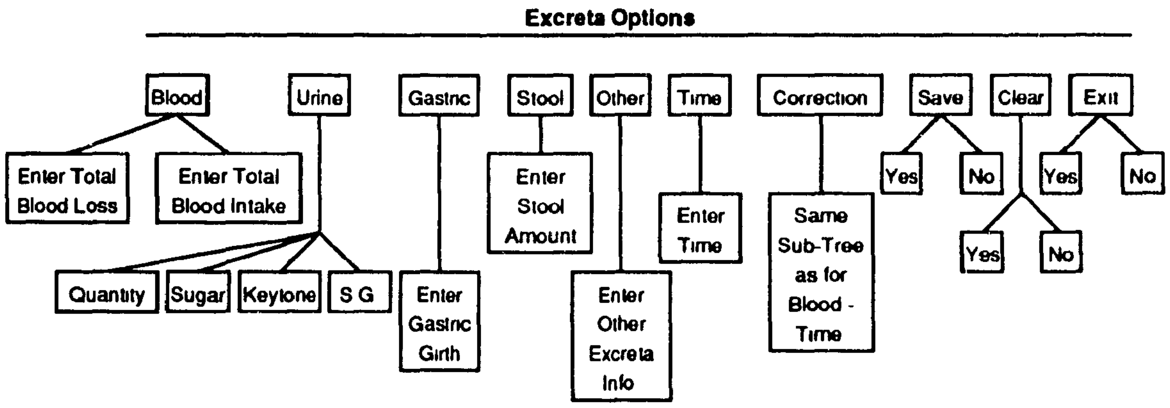


Figure 3.12: Fluid Balance Module Menu Tree (Excreta Menu)

also reflect the menu hierarchy of the Fluid Balance menus. This logical organization of menu items also facilitates the program's determination of the words or phrases to be echoed back to the user. All that needs to be known is where the user is in the menu tree, and the voice feedback messages can be generated accordingly. Another advantage of this structured division in the grammar and in the menu tree traversal is that illegal, invalid voice commands, as well as selections which are not valid in the current context will be ignored by the speech interface.

For menu tree traversal via mouse or keyboard selections, the menu tree structure is inherent in the options available to the user at a given level in the tree, and in the sequence of actions that the user must perform in order to achieve a certain action. These menu structures are handled internally by the Presentation Manager and require no tracking by the programmer. For voice operation, however, the structure of the tree must be tracked in the Fluid Balance module's code, in order to ensure the same syntactic consistency that the Presentation Manager ensures for mouse and keyboard selections.

3.3.4 Data Link Control

As mentioned in section 3.3.3, the data link packet is implemented in a modified Intel Hex format as shown in figure 3.13. In the data packet, one byte of information is coded as the ASCII representation of the two hex numbers. The information contained in the packet is as follows:

Bytes :	0	1 - 2	3 - 6	7 - 8	9 - (n-2)	(n-1) - n
	:	Module Number	Menu Number	Data Length	Data	Checksum

Figure 3.13: Data Link Packet Structure

- ":" - denotes the start of the packet.

- **"module number"** - two hex numbers representing the destination module number. A total of 256 modules can be serviced by this design. In the current implementation, since there is only only one module in the PDMS which supports the speech interface, this number is always the same.

- **"menu number"** - four hex numbers representing the menu id number in which the data, corresponding to either the voice command or voice response, is taken from. These menu numbers correspond directly to the id numbers of the menus in the Fluid Balance screens. This information is included to ensure that the user is "speaking" to the proper menu in the menu tree, as well as to ensure that the syntactic structure of the tree is being followed.

- **"data length"** - two hex numbers representing the length of the data of the data in the data field in bytes, or in hex characters divided by 2.

- **"2 × data length"** - of data. This corresponds to either the voice command information, or the voice response information. If the data field contains voice commands, the data is either the id number of the menu item selected, or the ASCII representation of text or non-menu selection commands. If the data field contains voice response messages, the data contains a code or a sequence of codes corresponding to the id number of the word or words to be echoed back by the system.

- **"checksum"** - two hex numbers representing the calculated checksum for a given packet. This is calculated by adding up modulo 256, all the character values in the data packet, except the leading colon (":"), and subtracting this value from 256 to obtain the checksum.

The protocol used for packet transmission is the ACK-NACK (acknowledge-no acknowledge) protocol. If the process receiving the packet receives it uncorrupted, that is, there is no discrepancy between the checksum calculated by the receiving process internally and the two numbers representing the checksum received at the end of the packet, an ACK, or acknowledge, is sent, and both processes on both sides of the RS-232C serial link proceed normally. Otherwise, a NACK, or not acknowledge, is sent to the sender, and that process retransmits the packet until an ACK is finally received, indicating the reception of an uncorrupted packet.

The format used in the speech interface is an adaptation of the true Intel Hex packet, where the header has been modified. What is retained, is the representation of one byte of data as two hex characters, and the use of the checksum for detecting packet corruption.

Chapter 4 Implementation, Results, and Future Extensions

This chapter describes the implementation issues of the speech interface and reproduces a sample session of a typical data entry process in the Fluid Balance module, showing some sample screens. Results of the speech interface will then be reviewed and discussed with particular attention being paid to the recognition accuracy in the recognition portion of the interface. Latency times of the system will also be discussed. Lastly, future extensions and improvements to the speech interface will be presented and discussed.

4.1 Speech Interface Implementation

4.1.1 The Fluid Balance Module

The new version of the Fluid Balance module was implemented using the Microsoft "C" compiler version 6.00 under the Presentation Manager environment of OS/2. The migration to the Presentation Manager version from the original full-screen mode version implemented under version 1.0 of 0S/2, began with the arrival of version 1.1 of OS/2. This version of the Fluid Balance module was originally compiled using version 5.1 of the Microsoft "C" compiler and was later recompiled with version 6.0 of the "C" compiler under the most recent version of OS/2 available at this time, version 1.3. The source code for the Fluid Balance module is divided among 10 program files totaling about 6400 lines of code. Out of this total, 2500 lines are dedicated to the speech interface. The size of the executable file is approximately 170 Kilobytes. All function definitions comply with the ANSI C standard [Kernighan and Ritchie, 1988]. The text and graphical drawing operations in the client area of the respective Fluid Balance module windows, rely on the standard API Presentation Manager functions available from the Presentation Manager Programmer's Toolkit lioraries. The graphical user interface of the module is based on the Presentation Manager's graphical user interface of the OS/2 operating system, and adheres to the Systems Application Architecture (SAA) definition for user interfaces.

As described in section 3.3.1, the Fluid Balance module consists of three windows; the main window, the Ingesta sheet window, and the Excreta sheet window. Although there are three separate windows, there is only one message queue for the entire program. All the data structures which contain the administrative patient information, such as the *name, hospital id number,* and *bed number* are shared among all the windows the program. Also, variables which store window handles, required for message posting between windows, are also available to all three window handling procedures. Both the Ingesta and the Excreta sheets are the *children* of the main window. The 10 program files which make up the Fluid Balance module are divided according to the functions that the procedures they contain perform, and the windows that these routines are used in. For example, there are files which contain the procedures which handle the drawing functions in the client areas of the Ingesta and Excreta windows, and procedures which handle the input from the various dialog boxes created by both the Ingesta and Excreta windows. There is also a file which contains the procedures that handle the processing of DDE messages, as well as a file which contains the procedures for responding to voice command messages and sending the appropriate voice response messages for playback on the DOS system.

The voice operation procedure has three state variables which track the user's progress through the menu tree of the main window, the Ingesta window, and the Excreta window. As the user traverses the menu tree using voice commands, the value of theses variables are changed to reflect the user's selection, and consequently, the new position in the menu tree.

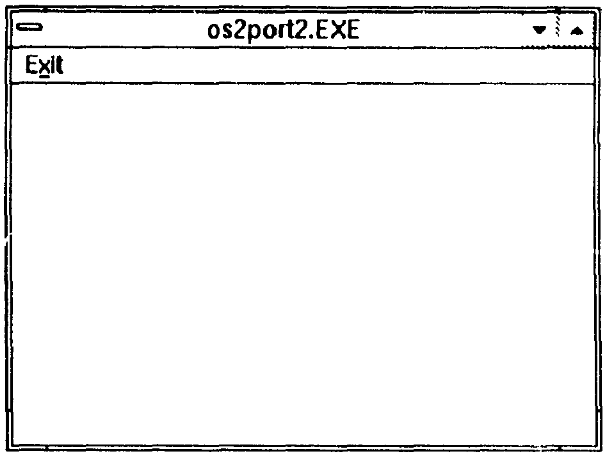


Figure 4.1: Window of OS/2 Serial Port Handling Process

The Fluid Balance module is implemented as a single-thread process; none of the functions in the program are so computationally intensive as to warrant the creation of another thread to handle the task.

4.1.2 The OS/2 Serial Port Handler

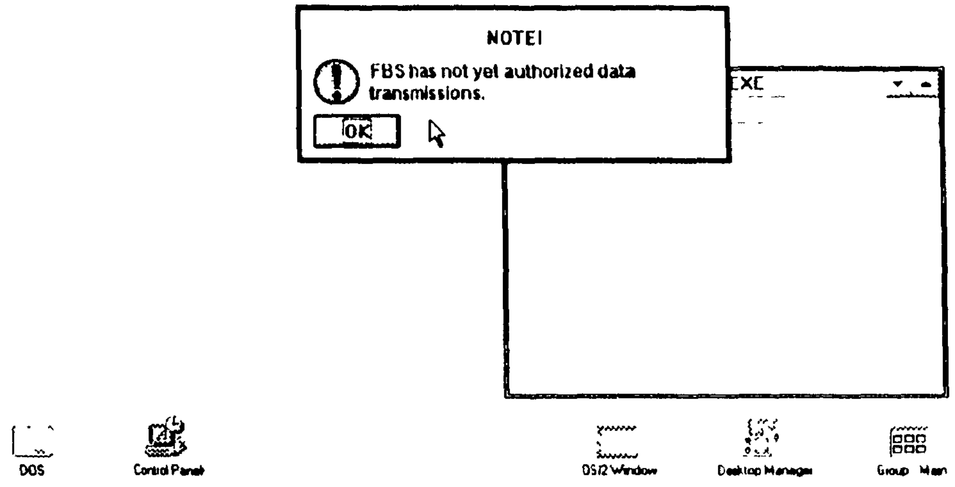
As was the case with the Fluid Balance module, the dedicated OS/2 serial port handling process was also implemented and compiled using version 6.0 of the Microsoft "C" compiler under the Presentation Manager environment of OS/2 version 1.3. A sample window of the serial port process is shown in figure 4 1. For this process, no real user interaction is required, but the graphical user interface is included in order to provide a simple means of terminating the program from the menu bar. In addition, running the OS/2 serial port process in the Presentation Manager environment enables the creation and use of a message queue, which is crucial for the processing of DDE messages sent from the Fluid Balance module. The source code for this program is contained in one file totalling about 950 lines of code. The size of the executable code for this process is about 32 Kilobytes

Two threads make up this process: a dedicated child thread which handles the serial port read and writes, and a main thread which processes the messages posted to its message queue. The motivation behind the creation of a dedicated thread to handle the serial port is to not to degrade the main thread's message processing tasks. Communication between the dedicated serial port thread and the main thread is done via message posting. The main thread communicates to the dedicated serial port thread via a semaphore.

As the dedicated serial port thread reads and decodes a data packet from the OS/2 serial port buffer, it dynamically allocates memory to store the incoming data based upon the data length field in the packet's header. This memory is subsequently freed when it is no longer needed by either the main thread or the dedicated serial port thread. When the serial port has received and decoded a valid data packet from the remote DOS system, it posts a message in the main thread's message queue. This indicates to the main thread that a valid voice command has arrived. The main thread then proceeds to send the the menu number, the data length, as well as the data in the packet's data field, to the module specified by the module number field in the packet header. Currently, the only module which supports a speech interface is the Fluid Balance module, but, as mentioned previously, this was included for future extensions.

If the Fluid Balance module has not established a DDE connection with the serial port process, an error is flagged to the user as shown in figure 4.2. If the DDE connection has been established, the main thread of the serial port procedure places the received menu number, data length, and data is placed in the *DDE-DATA* structure, whose elements consist of separate character arrays for storing the aforementioned information.

Once the dedicated serial port thread has sent the message to the main thread, it then sets a semaphore which will be cleared by the main thread when it received a DDE data message containing the voice response data to be transmitted to the remote DOS system. The structure for the DDE data transmission of voice response messages from the Fluid Balance module to the serial port process is identical to



NOTE!

FBS has not yet authorized data transmissions.

Figure 4.2: Dynamic Data Enchange Link Error Message

that described in the previous paragraph. Once the voice response data has been received by the main thread, the information stored in the DDE data structure is placed in global storage, accessible by both threads in the process. The main thread clears a semaphore, signalling to the port thread that there is data to be transmitted to the remote system which is contained in the global variables Storage for the data packet to be transmitted is also allocated dynamically, based on the length of the data to transmit. Once the data has been transmitted, the port thread waits for another data packet containing voice commands from the remote DOS system.

4.1.3 The DOS Port Program

On the DOS side, the single-task nature of the operating system does not allow the creation of threads to separate the serial port handling from the other tasks which are handled by this program. As was previously, mentioned the serial link operates at 9600 baud which necessitates an interrupt service routine for data reception. The incoming characters are stored in a 16 Kilobyte circular queue. Two pointers are used to mark the head and the tail of the queue. As characters arrive from the serial port, the end pointer is incremented, and as characters are read from the array, the front pointer is incremented. Since the serial port is not an interrupt-driven device under DOS, the host computer's programmable interrupt controller's (PIC) serial port interrupt flag must be set for the interrupt service routine to be called when a character is received from the serial port.

As with the OS/2 serial port process, the program also dynamically allocates memory for the transmission and receipt of data packets depending on the length of the data to transmit, or depending on the length field in the packet header received from the remote OS/2 system. The dynamically allocated memory is freed when it is no longer needed.

For voice response information, the data received corresponds to a sequence of four hex number codes which represent the word or sequence of words to be spoken back by the system. The translation from code numbers to file names is done by doing a binary search through the tree of code numbers, using a function supplied by "C" [Microsoft, 1990]. Since there are about 250 words in the playback vocabulary, it is unfeasible to store the content of the voice play back files in the host computer's 640 Kilobytes of conventional memory accessible by DOS. This implies that the *word list* feature for file play back on the Covox Voice Master Key II, described in section 3.2.2 system could not be used. Since there are 9 Megabytes of expanded memory on the host computer, 3 Megabytes have been allocated as a RAM disk to store these speech files for fast retrieval and the Covox file playback feature has been exploited instead of the word list feature. With the world list feature, the playback functions ir the "C" code could have been called with the code numbers received from the remote system directly. In the design implemented in the speech interface, the codes are translated into files names before the files are played back from the RAM disk directly.

The "C" binary tree search function was used for the code-to-speech-file-name translation as searches can be done in $O(\log(n))$ time, where n is the number of elements in the tree. This is better than doing a linear search which requires $O(n)$ time, and needs to be coded explicitly in the program.

This program was also compiled using version 6.0 of the Microsoft "C" compiler. The source code is contained in one source file of approximately 900 lines, and the executable version of this file is about 25 Kilobytes in size.

4.1.4 The Grammar File and Playback Files

The grammar file for the Fluid Balance module contains approximately 200 words. This large grammar is broken up into several context dependent grammars having no more than 20 words per grammar definition. The Fluid Balance grammar is made up of menu bar commands, menu item names, dialog box commands, some intravenous solution names, as well as the digits from 0 to 9. Two and more digit numbers are specified in the grammar by concatenating a sequence of two or more digits. The compiled recognizer file requires about 25 Kilobytes of disk space, and a typical user file of voice templates trained on the grammar requires about 20 Kilobytes per speaker. Therefore, for a given user, a total of 45 Kilobytes of space is required in the recognizer's memory, which is fits quite comfortably into the board's 512 Kilobytes of RAM.

The training time required to train the Fluid Balance module grammar was typically 23 minutes. The training process requires uttering each of the individual words which make up the vocabulary, and then training *scripts*, which are the concatenation of the words into phrases, as specified in the grammar file, for continuous speech recognition. The number of scripts which required training in the second part of the training process was 546 scripts. Note that all possible scripts in the grammar are not trained. If one wanted to train the concatenation of all possible four-digit numbers, one would have to train 10 000 scripts for this case alone! The Verbex training process only requires a few scripts which give the recognition algorithm enough information to extrapolate the remaining scripts in the grammar.

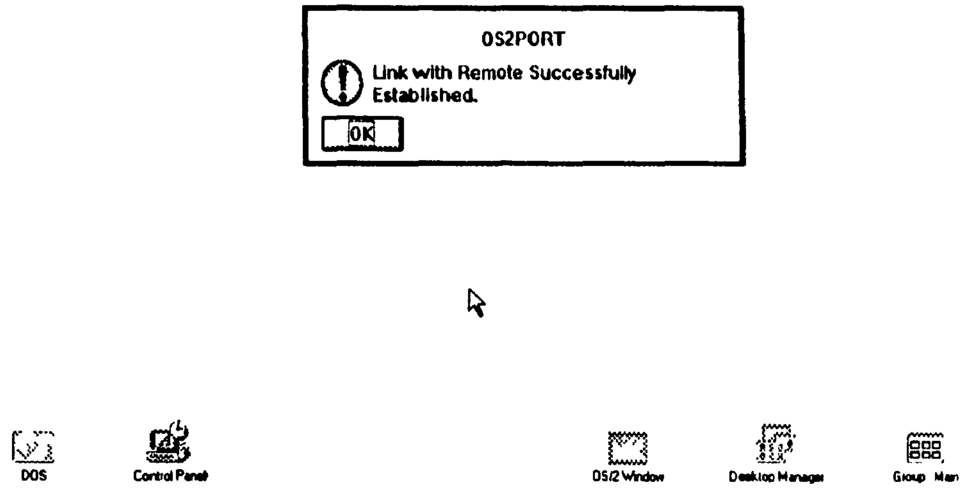
There are also about 250 possible words and phrases to be echoed back to the user. The speech for these words was digitized using a sampling rate of 14 Khz, using and 8-bit representation per sample. No compression was used to reduce the amount of space required to store these files on magnetic media. Consequently, 14 Kilobytes of disk space are required to store 1 second of digitized speech. The average length of a word is about 0.7 seconds. Hence, the disk space required to store the 250 words is about 2.5 Megabytes.

4.2 Sample Session Results

This section presents a sample session of the Fluid Balance module operation performing a typical data input process on the ingesta screen. All the functions performed in the data entry process were done using voice operation, and voice echoing was also provided.

The speech interface to the Fluid Balance module uses three programs: the OS/2 serial port handling process, the DOS serial port handling process, and the Verbex-supplied router of messages from the Verbex recognizer's hardware address into the keyboard buffer of the host computer.

A program cycle begins with the loading of the compiled grammar file and

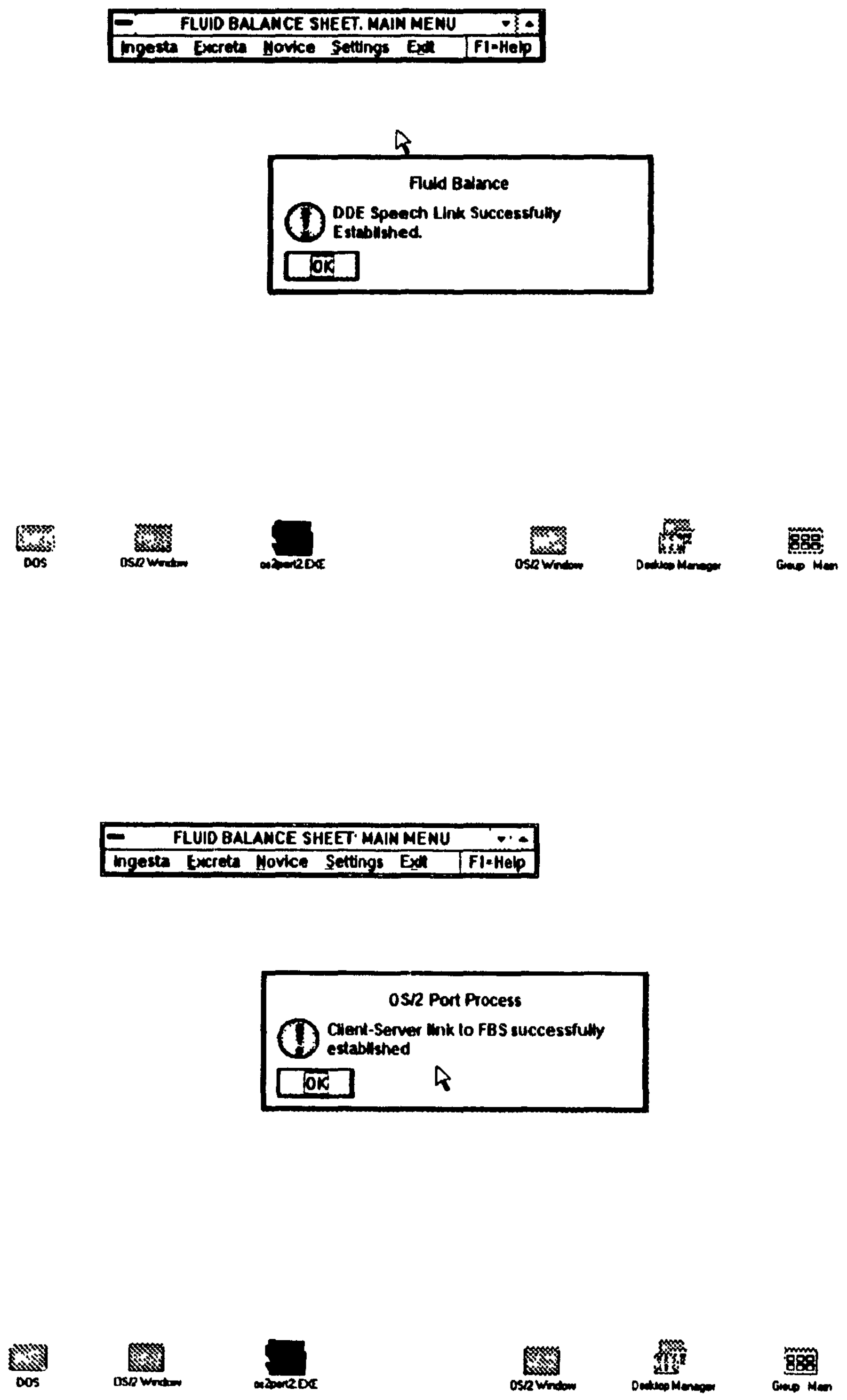

OS2PORT

Link with Remote Successfully Established.

Figure 4.3: Message Flagging Successful Linkup of DOS and OS/2 Systems

the user's voice templates for the grammar into the recognizer. Next, the Verbex message router is run, which installs itself as a terminate-and-stay-resident program, followed by the DOS port program, which waits for an initialization packet from the remote system, signalling that the OS/2 port process is active. On the OS/2 side, the serial port handling process is started first. It begins be sending an initialization packet through the serial line to the DOS system. If the DOS port program is active, then it will send an acknowledgement to the OS/2 serial port process. The successful connection of the two programs on both sides of the serial line is flagged to the user by the OS/2 port process as shown in figure 4 3.

Next, the Fluid Balance module is started. It first begins by establishing the DDE connections to and from the OS/2 port process. Messages flag the establishment of



Fluid Balance

DDE Speech Link Successfully Established.

OK

OS/2 Port Process

Client-Server link to FBS successfully established

OK

Figure 4.4: Successful Dynamic Data Exchange Link Messages

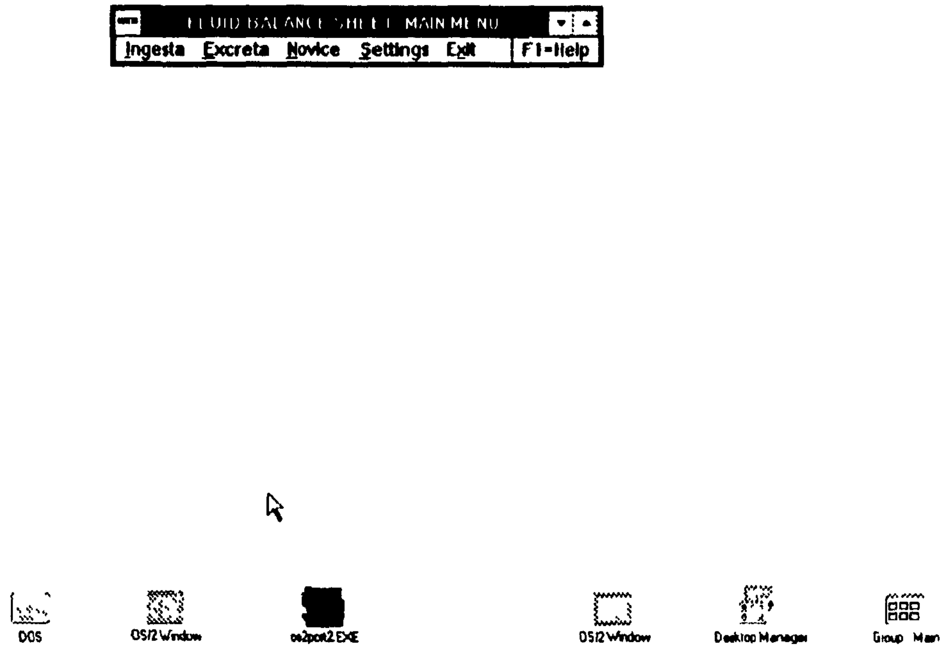


Figure 4.5: Main Fluid Balance Module Window at Start Up

the client-server links between the two processes as shown in figure 4.4. Following the establishment of the DDE links, the main window of the Fluid Balance module is created and displayed on the screen, as shown in figure 4.5.

The user can then create a blank Ingesta sheet by either clicking on the Ingesta item in the menu bar of the main window, by pressing the *ALT-I* key combination on the keyboard, or by saying "ingesta" into the microphone. Once the ingesta sheet is displayed, the user can proceed to select any of the ingesta menu bar items or switch back to the main menu window to enter the patient's administrative information.

Figures 4.6, 4.7, 4.8, and 4.8, included at the end of the chapter, reproduce the display of the Ingesta sheet of the Fluid Balance module at different stages during some sample data-input phases of operation. Note that a running total is kept of the Fluid Balance intravenous (IV) solution levels. Therefore, in the data entry process, the user only needs to enter the current IV solution levels in the appropriate columns; calculations of the total solution intake are done automatically, and displayed in the *Act. In* (actual intake) column. Finally, if for any reason, the DDE link should fail, error messages are displayed on the screen, informing the user of the problem.

4.3 Test and Evaluation Process

For the testing process of the speech interface, it was desired to have a mix of both novice and knowledgeable computer users. This would gave a better idea of how users with different backgrounds would find data entry using a non-traditional means of computer input and output.

Eight users were tested in the development laboratory at McGill. For the test process, users were asked to perform some typical fluid balance data entry tasks. Prior to testing, those users not already familiar with the PDMS or the Presentation Manager environment of OS/2 were given a tutorial outlining the functionality and the motivation behind the use of the PDMS in general and of the Fluid Balance module in particular. The users were then given a paper copy of a sample Fluid Balance sheet obtained from the Montreal Children's Hospital. After a review of how the nurses proceed to enter the fluid balance data on the form and training the Fluid Balance module vocabulary, the users were asked to enter the data into the Fluid Balance module first using the speech interface only, and then performing the same tasks using the conventional mouse and keyboard input modalities.

Of the eight users tested, two were experienced computer programmers with good experience with the Presentation Manager environment of OS/2 and the Fluid Balance module, two were experienced Presentation Manager users, but had no prior experience with the Fluid Balance module, two were experienced computer users but had no previous Presentation Manager experience, and the last two users tested had little computer experien^e and no experience using either the Presentation Manager environment or the Fluid Balance module. The following tables show some information obtained during the test process, for each of the four groups of two users with similar computer backgrounds.

The tables display the following information of interest for data entry using the speech interface only, and the keyboard and mouse only:

- **Average task completion time:** indicates the average time required to complete the entry of one fluid balance value into the Fluid Balance module.
- **Number of errors in performing task:** indicates the total number of errors the users did when performing the sample entry of data in the Fluid Balance module.
- **Number of tasks performed in 10 minutes:** indicates the total number of fluid levels entered into the Fluid Balance module during a 10 minute interval.

Such information useful for obtaining some insight into which input modality is more effective for which category of user, as well as giving an indication as to whether a speech interface is an effective input modality for this application.

In addition, the tables also present information on the performance of the speech recognition portion of the speech interface by displaying the following information:

- **Training Time:** the time required by the user to train the Fluid Balance module vocabulary.
- **False acceptance rate:** the rate at which the speech recognition system responds to extraneous noises such a coughs.
- **False rejection rate:** the rate at which valid utterances are ignored or rejected by the system (no response).

- **Substitution rate:** the rate at which a word uttered by the speaker is incorrectly recognized to be another word in the vocabulary.

- **Recognition rate:** is the rate at which the system responds correctly to user utterances $(1-(false\, acceptance\, rate + false\, rejection\, rate + substitution\, rate))$.

The breakdown of the recognition rate into its various components generally gives a better indication of system performance than quoting the recognition rate alone [O'Shaughnessy, 1987].

The typical response time for the speech interface was 1.5 seconds, which is made up of an approximate 0.56 seconds for the recognition time, 0.07 seconds between the time the word is recognized and the time in which the command is executed on the Fluid Balance module, 0.07 seconds for the response string to be passed on to the system, and a typical 0.8 seconds for speech file playback.

From the test results the following observations can be made. For experienced Fluid Balance module and Presentation Manager users, whose test results are shown in table 4.1., there was no significant advantage to using voice over mouse and keyboard for input. This can be attributed to the fact that experienced users of a given application program can easily manoeuvre the mouse and keyboard to perform the required operations. As shown above the speech interface operation from utterance to audio feedback completion typically requires 1.5 seconds. Even if no audio feedback were supplied by the interface, the interface would still require an average of 0.63 seconds to execute a user command [Petroni *et al.*, 1991].

Once users have mastered the graphical user interface of the system, the speech interface offers no speed advantage, when compared to the quasi-instantaneous time that the system requires to respond to a mouse click or a keyboard press. Hence expert users of a system seem to be bound by the the speed of the input modality and the relative task complexity in the system.

During the testing process, both expert users found the voice feedback annoy-

User A

	Speech	Keyboard/Mouse
Average Task Completion Time	12 seconds	10 seconds
Number of Errors in Performing Task	0	0
Number of Tasks Completed in 10 minutes	33	35

Training Time	20 minutes 15 seconds		
	Occurrences	Utterances	Rate (%)
False Acceptance	0	600	0%
False Rejection	13	600	2.17%
Substitutions	0	600	0%
Recognition Rate	97.8%		

User B

	Speech	Keyboard/Mouse
Average Task Completion Time	20 seconds	15 seconds
Number of Errors in Performing Task	0	0
Number of Tasks Completed in 10 minutes	27	32

Training Time	22 minutes 38 seconds		
	Occurrences	Utterances	Rate (%)
False Acceptance	0	600	0%
False Rejection	26	600	4.33%
Substitutions	2	600	0.33%
Recognition Rate	95.3%		

Table 4.1: Test Set 1: Expert Presentation Manager and Fluid Balance Module Users

ing, since they already knew from memory what options were available to them at a given level in the menu tree. It was also mentioned that for "eyes-free" operation, some speech generation is essential. Overall, both users felt that a speech interface would be quite useful in the intensive care unit, where users are not expected to be seasoned computer users, and the mobility for "hands-free" and "eyes-free" operation is desired.

The next set of users tested were had previous Presentation Manager experience but had no prior experience with the Fluid Balance module. The results of their sessions are shown in table 4.2. Comparing table 4.2 and table 4.1, one can notice that the task completion times are generally higher for both voice and keyboard and mouse input modalities. Also, the task completion times for the two input

User C

	Speech	Keyboard/Mouse
Average Task Completion Time	25 seconds	25 seconds
Number of Errors in Performing Task	0	1
Number of Tasks Completed in 10 minutes	24	24

Training Time	20 minutes 57 seconds		
	Occurrences	Utterances	Rate (%)
False Acceptance	0	600	0%
False Rejection	20	600	3.33%
Substitutions	0	600	0%
Recognition Rate	96.7%		

User D

	Speech	Keyboard/Mouse
Average Task Completion Time	22 seconds	23 seconds
Number of Errors in Performing Task	0	2
Number of Tasks Completed in 10 minutes	27	26

Training Time	21 minutes 20 seconds		
	Occurrences	Utterances	Rate (%)
False Acceptance	0	600	0%
False Rejection	21	600	3.50%
Substitutions	3	600	0.50%
Recognition Rate	96.0%		

Table 4.2: Test Set 2: Results for Experienced Presentation Manager Users

modalities are almost identical for a given user. In the case where users are familiar with the graphical user interface of a system, but not the menus and menu items of a particular application, voice operation is equivalent to keyboard and mouse. At first, the more detailed audio feedback messages were used by both users. As the input process progressed, and the users became more familiar with the system, the detailed messages were substituted by the shorter, less descriptive ones, even when "eyes-free" data entry was performed.

Table 4.3 shows the session results for the experience computer users with no Presentation Manager experience. As expected, the task completion time for both voice and keyboard/mouse input is greater than the previous two sets of users. This increase is primarily due to the unfamiliarity of the users to the Presentation

User E

	Speech	Keyboard/Mouse
Average Task Completion Time	30 seconds	45 seconds
Number of Errors in Performing Task	2	5
Number of Tasks Completed in 10 minutes	20	13

Training Time	23 minutes 15 seconds

	Occurrences	Utterances	Rate (%)
False Acceptance	0	600	0%
False Rejection	28	600	4 67%
Substitutions	1	600	0.17%
Recognition Rate	95.2%		

User F

	Speech	Keyboard/Mouse
Average Task Completion Time	32 seconds	40 seconds
Number of Errors in Performing Task	4	5
Number of Tasks Completed in 10 minutes	19	15

Training Time	21 minutes 47 seconds

	Occurrences	Utterances	Rate (%)
False Acceptance	0	600	0%
False Rejection	19	600	3.17%
Substitutions	0	600	0%
Recognition Rate	96.8%		

Table 4.3: Test Set 3. Test Results for Experienced Computer Users

Manager environment. In this test case, the use of the speech interface lead to a shorter task completion time than performing the same tasks using the keyboard and mouse. The users here felt that the use of voice feedback was especially helpful in the data entry process. Also, they felt that the use of voice feedback coupled with the visual feedback on the screen accelerated the command learning process, decreased task completion time, and helped with command retention. These users were especially appreciative of the detailed voice messages which told them of the available options at every step in the entry process They also mentioned that operating a module in an unfamiliar operating system environment was less intimidating when the speech interface was used than when keyboard and mouse was used.

User G

	Speech	Keyboard/Mouse
Average Task Completion Time	35 seconds	45 seconds
Number of Errors in Performing Task	6	12
Number of Tasks Completed in 10 minutes	17	13

Training Time	20 minutes 45 seconds

	Occurrences	Utterances	Rate (%)
False Acceptance	0	600	0%
False Rejection	18	600	3.00%
Substitutions	0	600	0%
Recognition Rate	97.0%		

User H

	Speech	Keyboard/Mouse
Average Task Completion Time	32 seconds	48 seconds
Number of Errors in Performing Task	0	0
Number of Tasks Completed in 10 minutes	5	10

Training Time	24 minutes 05 seconds

	Occurrences	Utterances	Rate (%)
False Acceptance	0	600	0%
False Rejection	22	600	3.67%
Substitutions	0	600	0%
Recognition Rate	96.3%		

Table 4.4: Test Set 4: Results for Novice Computer Users

Table 4.4 shows the results of the test session with the two users with little or no computer experience, and, consequently, no Presentation Manager or Fluid Balance module exposure. As expected the average time required to perform a typical data entry task in the Fluid Balance module was the longest of the four test sets. As with the previous case, this can be attributed to the unfamiliarity of the users to the both the graphical user interface of the Presentation Manager environment, and to the Fluid Balance module's commands. In addition, this case also showed that operation of the Fluid Balance module and data entry with the speech interface yield a shorter average task completion time than performing the equivalent tasks by keyboard or mouse. These users echoed the sentiments of the previous set of users by saying that the use of voice feedback coupled with the visual feedback from the screen was especially helpful in getting familiar with the system and its commands. They also felt less intimidated and more secure with the speech interface than with the keyboard and mouse modalities, adding that a new environment with an unfamiliar program was very overwhelming for users which are not computer literate.

Some observations which were noted on all the users was that in the initial stages of the training process, training would go slow, as users were not sure what would happen next. As the users got more comfortable with the system and the training process, the grammar training would then proceed at a more rapid pace. Also, the novice users felt more comfortable entering data by voice than by either keyboard or mouse. Overall, the speech interface was not rejected by any of the users tested, but it was better received by the novice users who found that it was a less intimidating, easier to use, and more familiar means of communication, especially when learning a new system.

The results obtained for the speech recognition portion of the speech interface during the tests also gives some useful information. Looking at the values for the rejection rate for all the users, one can observe that this part of the recognizer errors is what contributes the most to the lowering of the overall recognition accuracy. Most of the users were not bothered by this, but two users mentioned that if the system didn't recognize a valid word after two repetitions, they would get annoyed and frustrated with the system. They also added that having to repeat a word more than twice could be very frustrating, especially for a novice user. From the values obtained from the tables, the speech recognition had an overall recognition rate of 96.4%. There were no false acceptances recorded. Substitution errors amounted to 2.89% of the total errors recorded, and false rejection errors amounted to 97.1% of the total errors.

One of the users got into what is known as a "degradation spiral" in the recognition process. When a command was not recognized, the user would repeat the name with slow or exaggerated pronunciation as a familiar way of "being clearer".

These habits hurt recognition. When original templates were spoken in a normal voice, exaggerated inputs were recognized less frequently than normal utterances. While this occurred with only one of the users in the test set, it is suspected that such attempts at "being clearer" when dealing with speech recognition systems, are hard habits to break for users when a system does not recognize a valid word [Brennan *et al.*, 1991].

Overall, it can be said that the speech interface is especially effective and useful for the novice user specifically, and for all users generally when "eyes-free" and "hands-free" operation is required. The recognizer accuracy of the speech recognition system should be as high and as robust as possible in order to minimize the potential user frustration when trying to input voice commands. For novice users, the more detailed voice response messages were especially well received, giving them a sense of being "walked through" the module's operation step-by-step. For the more experienced users, some form of voice feedback is required to ensure that the system correctly "understood" what was said by the user.

It must be kept in mind that while speech recognition is a useful technology, it is still not a mature one. Much work in the research and development of more sophisticated systems still needs to be done. While most systems on the market today offer speaker dependent, discrete word recognition with relatively small allowable vocabularies, some systems today do allow either continuous speech or speaker independence with larger vocabulary sizes. Systems of tomorrow will be capable of understanding natural language statements in the form of sentences.

4.4 Discussion and Future Extensions

One disadvantage of having a separate system housing the speech interface hardware sub-systems is that two computer systems are required: one for the speech interface sub-systems which run under DOS, and the other for the PDMS software which runs under OS/2. It would be more desirable to have both the PDMS and the speech interface hardware operating in one self-contained system. This would also eliminate the need for the communication programs, and would allow for a more seamless interface to the Fluid Balance module. The advantage of the "distributive" nature of the system is that the interface can be added onto any system that has a serial port and a serial port handling process, with the same functionality as the OS/2 serial port process, to handle the protocol used in the speech interface to communicate between separate systems over the serial line. This would allow the speech interface system to operate a UNIX system, for example.

To avoid having to train the system, a speaker independent system would be desirable. This would alleviate users from having to spend one-half hour training the system in order to use it. The advantage of a speaker dependent system is that it can be trained in the language that the user is most comfortable with Also, recognition accuracy is usually better for speaker dependent systems than for speaker independent systems. A speaker dependent system will also allow a certain degree of security in the data entry process. In an unauthorized person decided to tamper with the system, the recognition system will not respond to voice commands, since it will not recognize a person whose voice features differ from the person who trained the system.

The Covox Voice Master Key II speech output system provides reasonable speech quality for the digitized speech. When sampled at 14 Khz, the digitized speech is of broadcast quality with no noise in the played back waveform. The use of digitized speech output allows the messages to be played back either in English or in French. The "Speech Thing" synthesis system was also investigated for this application, but it was found that the initial intelligibility and quality of the speech were not acceptable without a visual prompting of what was being uttered. Perhaps with sufficient training, a user could eventually be satisfied with the speech synthesis output quality.

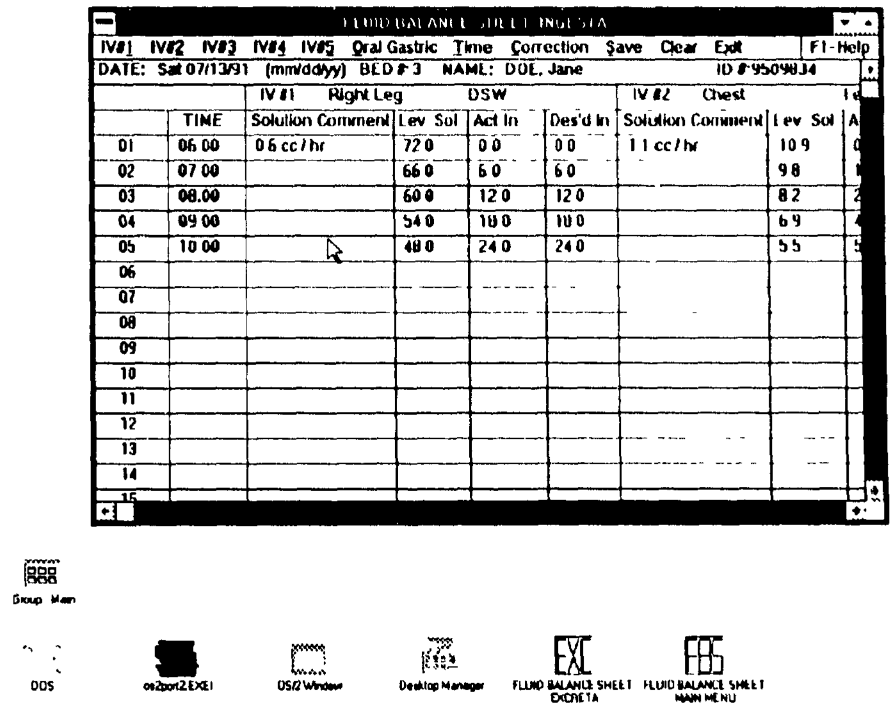
As the speech interface described here currently stands, it would be straight forward to extend it to other PDMS modules. All that would need to be tailored for a particular module would be the grammar, code the proper procedures to respond to the menu bar items, menu items, and dialog box items specific to the application, as well as tailoring the voice response messages for a specific module.

While the speech interface provides a "hand-free" and "eyes-free" means of manually entering data, another interesting technology is emerging to specifically deal with the handling of data which can only be entered manually. This technology is based on hand-written character recognition. While still in the prototype stages, systems are being announced which will present emulated forms to the user on a portable flat "screen", the size of a clipboard, which will allow the user to write information in the fields of the simulated form. The printed characters are then converted to their ASCII equivalents and stored in the system's local storage. The only interface to this pen-based system is through a stylus. A nurse could thus visit the various infusion pumps and write the data down directly on the simulated from. Since all of the intensive care unit staff are familiar with form fill-in, no complex familiarity with the system is required. The pen-based system also contains an RS-232C port where it can download data to a host computer. A training process is required in order for the system to recognize a particular user's handwriting. While these pen-based systems are not yet available, it will be interesting to see how this technology will challenge the speech interface for the manual data entry process.

Lastly, some consideration should be given to the design, implementation, and generation process of the graphical user interface. With the adoption of IBM's unified user interface specification, better known as the Systems Application Architecture (SAA), the graphical user interface on IBM platforms will have the same consistent interface. Consistency of the user interface across applications is important for the user when dealing with potentially many different programs. For a novice user, this is especially important, since dealing with different interfaces, each with its own unique mode of operation, may be overwhelming. Differing from the version of the PDMS implemented in the full-screen mode of OS/2 1.0, the next generation of the PDMS modules will have interfaces which will be similar to the new Fluid Balance module to which the speech interface was added. This will free the user of having to learn different interfaces for the different PDMS modules.

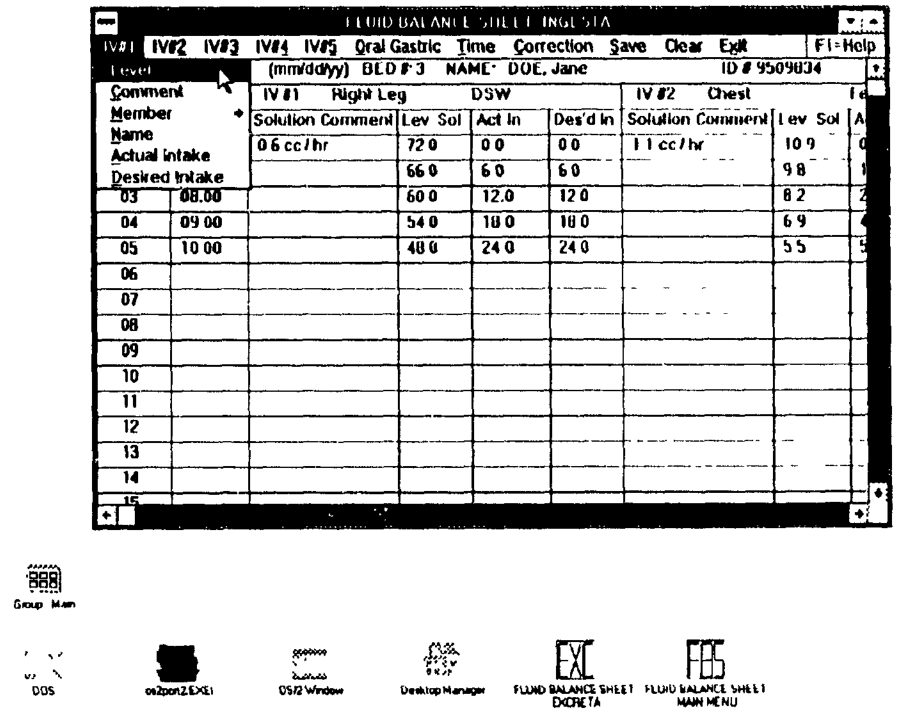
In applications where the user interface is the main consideration of a software's acceptability to the end users, as is the case with the PDMS, it is imperative the system developers get feedback from the end users as the software goes through various stages in its development and evolution. In a programming environment such as OS/2's Presentation Manager, the implementation of the graphical user interface in high-level language code, requires a good knowledge of the various OS/2 Presentation Manager function calls and their uses. In addition, the implementation of the graphical user interface often requires many lines of code, which would be discarded if the interface proved to be inadequate in the eyes of the end users. Such a situation would waste many hours of programming in the implementation of an interface which needs to be improved in the future. Small, seemingly minor changes could require many hours and lines of code. A means of quickly generating a graphic interface prototype would be desirable. This would allow prototype systems to be developed in a very short time, without the need of an in depth knowledge of the operating system functions from the developers. Moreover, any changes required in the "look and feel" of the interface could be implemented in a relatively short time, effectively shortening the development life cycle of the software. In the end, it would hoped that this would produce an interface which also better meets the expectations and specifications of the end users.

Computer Aided Software Engineering (CASE) tools attempt to address this problem by allowing fast prototyping of user interfaces by allowing the user to cut, copy, and paste graphical objects onto the screen. These tools also produce source code for the prototype interfaces, allowing the programmer to add the code for the computational "meat" to the code for the interface "skeleton". Many CASE tools also produce source code for from flow diagrams of the program. While these new fourth generation programming "languages" are far from perfect, their emergence has served to help to reduce the software development life cycle. As these tools mature, and their use becomes more widespread, it is expected that interface and program generation will be quite rapid, and the quality of the resulting software will be better.


FLUID BALANCE SHEET INGESTA
IV#1 IV#2 IV#3 IV#4 IV#5 Oral Gastric Time Correction Save Clear Exit
F1-Help
Group Main
DOS
os2port2.EXE
OS/2 Window
Desktop Manager
FLUID BALANCE SHEET EXCRETA
FLUID BALANCE SHEET MAIN MENU

DATE: Sat 07/13/91 (mm/dd/yy) BED # 3 NAME: DOE, Jane ID # 9509834

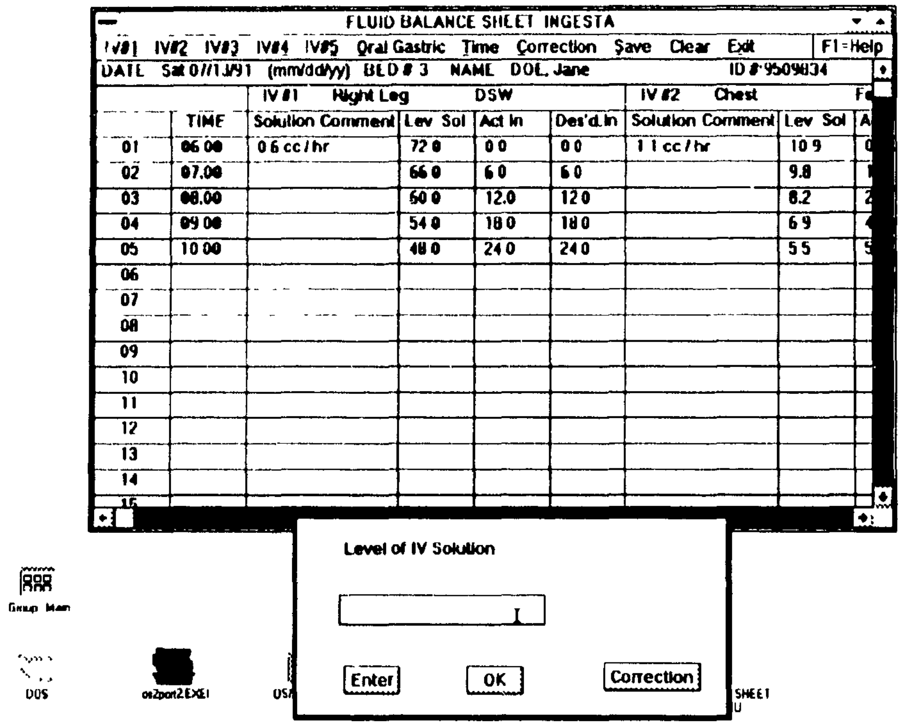
		IV #1 Right Leg		DSW		IV #2 Chest		
	TIME	Solution Comment	Lev Sol	Act In	Des'd In	Solution Comment	Lev Sol	A
01	06 00	0 6 cc / hr	72 0	0 0	0 0	1 1 cc / hr	10 9	0
02	07 00		66 0	6 0	6 0		9 8	1
03	08.00		60 0	12 0	12 0		8 2	2
04	09 00		54 0	18 0	18 0		6 9	4
05	10 00		48 0	24 0	24 0		5 5	5
06								
07								
08								
09								
10								
11								
12								
13								
14								


FLUID BALANCE SHEET INGESTA
IV#1 IV#2 IV#3 IV#4 IV#5 Oral Gastric Time Correction Save Clear Exit
F1=Help
Level
Comment
Member
Name
Actual Intake
Desired Intake
Group Main
DOS
os2port2.EXE
OS/2 Window
Desktop Manager
FLUID BALANCE SHEET EXCRETA
FLUID BALANCE SHEET MAIN MENU

(mm/dd/yy) BED # 3 NAME: DOE, Jane ID # 9509834

		IV #1 Right Leg		DSW		IV #2 Chest		
		Solution Comment	Lev Sol	Act In	Des'd In	Solution Comment	Lev Sol	A
		0 6 cc / hr	72 0	0 0	0 0	1 1 cc / hr	10 9	0
			66 0	6 0	6 0		9 8	1
03	08.00		60 0	12.0	12 0		8 2	2
04	09 00		54 0	18 0	18 0		6 9	4
05	10 00		48 0	24 0	24 0		5 5	5
06								
07								
08								
09								
10								
11								
12								
13								
14								

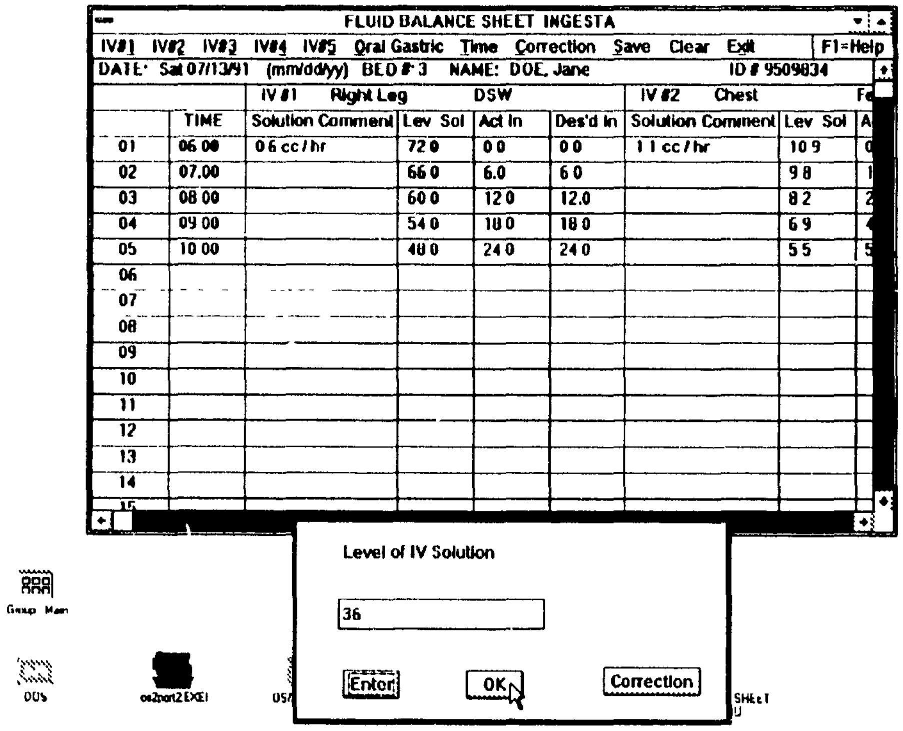
Figure 4.6: Sample Fluid Balance Module Data Entry Session (Part A)


FLUID BALANCE SHEET INGESTA
IV#1 IV#2 IV#3 IV#4 IV#5 Oral Gastric Time Correction Save Clear Exit F1=Help
Group Man
DOS
os2port2.EXE
Enter
OK
Correction
SHEET

DATE Sat 07/13/91 (mm/dd/yy) BED # 3 NAME DOE, Jane ID # 9509834

		IV #1 Right Leg	DSW			IV #2 Chest		F
	TIME	Solution Comment	Lev Sol	Act In	Des'd In	Solution Comment	Lev Sol	A
01	06 00	0 6 cc / hr	72 0	0 0	0 0	1 1 cc / hr	10 9	0
02	07.00		66 0	6 0	6 0		9.8	1
03	08.00		60 0	12.0	12 0		8.2	2
04	09 00		54 0	18 0	18 0		6 9	4
05	10 00		48 0	24 0	24 0		5 5	5
06								
07								
08								
09								
10								
11								
12								
13								
14								

Level of IV Solution


FLUID BALANCE SHEET INGESTA
IV#1 IV#2 IV#3 IV#4 IV#5 Oral Gastric Time Correction Save Clear Exit F1=Help
Group Man
DOS
os2port2.EXE
Enter
OK
Correction
SHEET

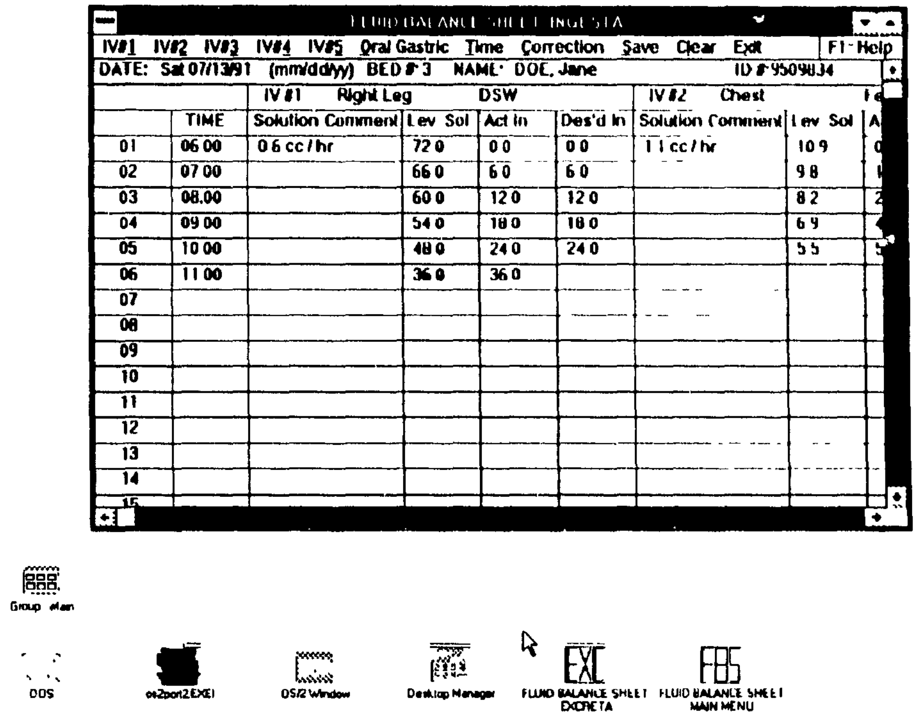
DATE: Sat 07/13/91 (mm/dd/yy) BED # 3 NAME: DOE, Jane ID # 9509834

		IV #1 Right Leg	DSW			IV #2 Chest		F
	TIME	Solution Comment	Lev Sol	Act In	Des'd In	Solution Comment	Lev Sol	A
01	06 00	0 6 cc / hr	72 0	0 0	0 0	1 1 cc / hr	10 9	0
02	07.00		66 0	6.0	6 0		9 8	1
03	08 00		60 0	12 0	12.0		8 2	2
04	09 00		54 0	18 0	18 0		6 9	4
05	10 00		48 0	24 0	24 0		5 5	5
06								
07								
08								
09								
10								
11								
12								
13								
14								

Level of IV Solution

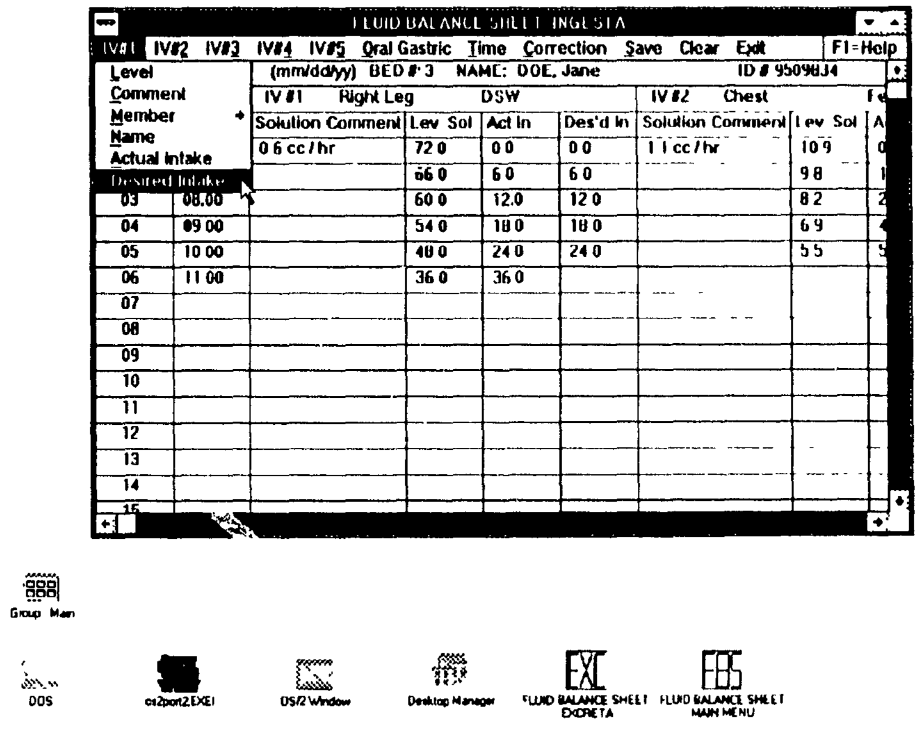
36

Figure 4.7: Sample Fluid Balance Module Data Entry Session (Part B)


FLUID BALANCE SHEET INGESTA
IV#1 IV#2 IV#3 IV#4 IV#5 Oral Gastric Time Correction Save Clear Exit F1=Help
Group Main
DOS
os2port2.EXE
OS/2 Window
Desktop Manager
FLUID BALANCE SHEET EXCRETA
FLUID BALANCE SHEET MAIN MENU

DATE: Sat 07/13/91 (mm/dd/yy) BED # 3 NAME: DOE, Jane ID # 9509834

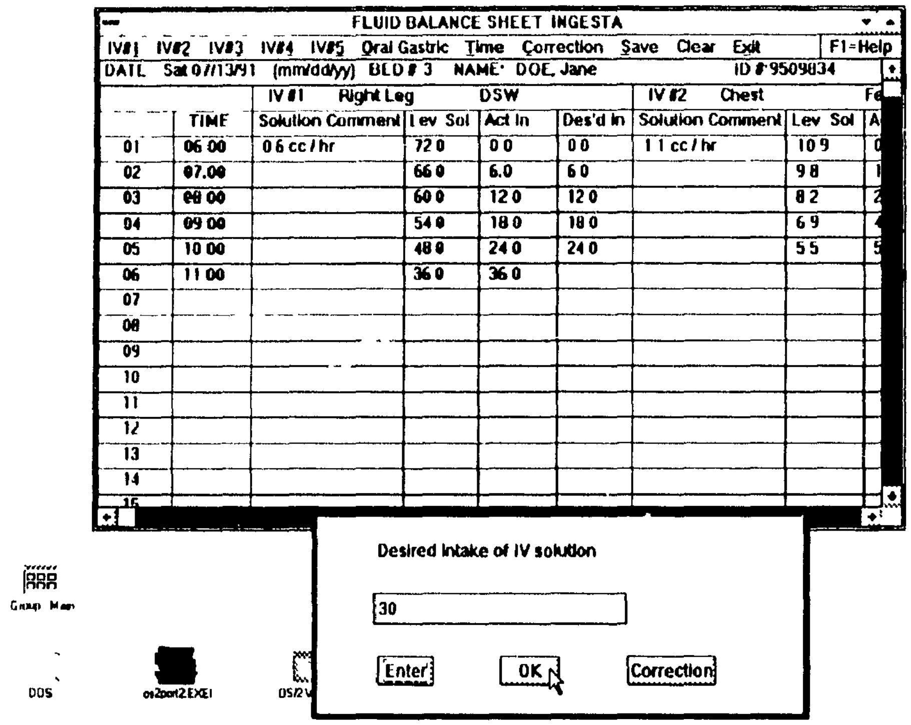
		IV #1 Right Leg	DSW			IV #2 Chest		F
	TIME	Solution Comment	Lev Sol	Act In	Des'd In	Solution Comment	Lev Sol	A
01	06.00	0.6 cc/hr	72.0	0.0	0.0	1.1 cc/hr	10.9	0
02	07.00		66.0	6.0	6.0		9.8	1
03	08.00		60.0	12.0	12.0		8.2	2
04	09.00		54.0	18.0	18.0		6.9	4
05	10.00		48.0	24.0	24.0		5.5	5
06	11.00		36.0	36.0				
07								
08								
09								
10								
11								
12								
13								
14								


FLUID BALANCE SHEET INGESTA
IV#1 IV#2 IV#3 IV#4 IV#5 Oral Gastric Time Correction Save Clear Exit F1=Help
Level
Comment
Member
Name
Actual Intake
Desired Intake
Group Main
DOS
os2port2.EXE
OS/2 Window
Desktop Manager
FLUID BALANCE SHEET EXCRETA
FLUID BALANCE SHEET MAIN MENU

(mm/dd/yy) BED # 3 NAME: DOE, Jane ID # 9509834

		IV #1 Right Leg	DSW			IV #2 Chest		F
		Solution Comment	Lev Sol	Act In	Des'd In	Solution Comment	Lev Sol	A
		0.6 cc/hr	72.0	0.0	0.0	1.1 cc/hr	10.9	0
			66.0	6.0	6.0		9.8	1
03	08.00		60.0	12.0	12.0		8.2	2
04	09.00		54.0	18.0	18.0		6.9	4
05	10.00		48.0	24.0	24.0		5.5	5
06	11.00		36.0	36.0				
07								
08								
09								
10								
11								
12								
13								
14								

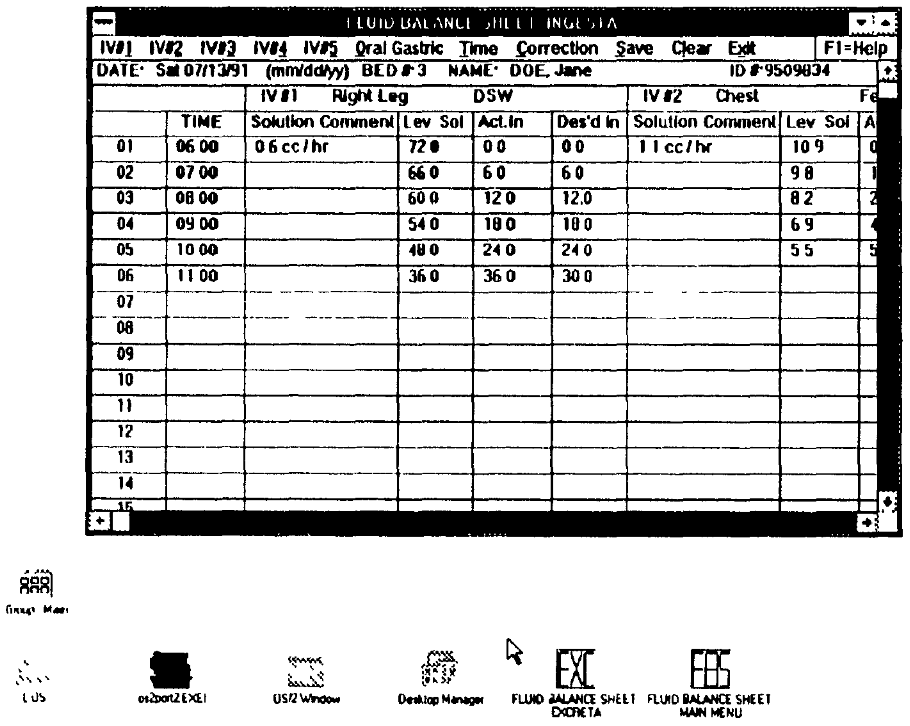
Figure 4.8: Sample Fluid Balance Module Data Entry Session (Part C)


FLUID BALANCE SHEET INGESTA
IV#1 IV#2 IV#3 IV#4 IV#5 Oral Gastric Time Correction Save Clear Exit F1=Help
Desired Intake of IV solution
Enter
OK
Correction
Group Main
DOS
os2port2.EXE
OS/2

DATE Sat 07/13/91 (mm/dd/yy) BED # 3 NAME: DOE, Jane ID # 9509834

		IV #1 Right Leg	DSW			IV #2 Chest		Fe
	TIME	Solution Comment	Lev Sol	Act In	Des'd In	Solution Comment	Lev Sol	A
01	06 00	0 6 cc / hr	72 0	0 0	0 0	1 1 cc / hr	10 9	0
02	07.00		66 0	6.0	6 0		9 8	1
03	08 00		60 0	12 0	12 0		8 2	2
04	09 00		54 0	18 0	18 0		6 9	4
05	10 00		48 0	24 0	24 0		5 5	5
06	11 00		36 0	36 0				
07								
08								
09								
10								
11								
12								
13								
14								

30


FLUID BALANCE SHEET INGESTA
IV#1 IV#2 IV#3 IV#4 IV#5 Oral Gastric Time Correction Save Clear Exit F1=Help
Group Main
DOS
os2port2.EXE
OS/2 Window
Desktop Manager
FLUID BALANCE SHEET EXCRETA
FLUID BALANCE SHEET MAIN MENU

DATE: Sat 07/13/91 (mm/dd/yy) BED # 3 NAME: DOE, Jane ID # 9509834

		IV #1 Right Leg	DSW			IV #2 Chest		Fe
	TIME	Solution Comment	Lev Sol	Act.In	Des'd In	Solution Comment	Lev Sol	A
01	06 00	0 6 cc / hr	72 0	0 0	0 0	1 1 cc / hr	10 9	0
02	07 00		66 0	6 0	6 0		9 8	1
03	08 00		60 0	12 0	12.0		8 2	2
04	09 00		54 0	18 0	18 0		6 9	4
05	10 00		48 0	24 0	24 0		5 5	5
06	11 00		36 0	36 0	30 0			
07								
08								
09								
10								
11								
12								
13								
14								

Figure 4.9: Sample Fluid Balance Module Data Entry Session (Part D)

Chapter 5 Conclusion

In this thesis manuscript, the development of a speech interface for bedside data entry in an intensive care unit was described. A literature survey of medical systems and speech systems was presented. Following an overview of the patient data management system (PDMS), for which the speech interface was implemented, its implementation was elaborated. Test results were presented to evaluate the interface and some suggested improvements were also discussed. An evaluation of the implementation described here by nursing and medical personnel in the pediatric intensive care unit of the Montreal Children's Hospital will be undertaken as soon as the system can be installed at the hospital. These tests will reveal the robustness of the system in the presence of noise such as alarms and crying.

References

[Abrami and Johnson, 1990] P. F. Abrami and J. E. Johnson, eds., *Bringing Computers to the Hospital Bedside: an Emerging Technology*. New York, NY: Springer Publishing Company, 1990.

[Aguglia *et al.*, 1990] F. Aguglia, G. Bertazzoni, and A. Scillone, "Computers in emergency medicine," *Medicina (Firenze)*, vol. 10, no. 1, pp. 53–8, 1990.

[Alesch *et al.*, 1991] F. Alesch, E. Donauer, and G. Heinen, "The use of personal computers for the collection, storage and analysis of data from neurosurgical intensive care monitoring," *Anaesthesist*, vol. 40, no. 1, pp. 52–6, 1991.

[Bergeron and Locke, 1990] B. Bergeron and S. Locke, "Speech recognition as a user interface.," *MD Comput*, vol. 7, no. 5, pp. 329–34, 1990.

[Bitzer and Domer, 1989] B. Bitzer and R. Domer, "Speech-interface for automation and power control systems," in *Proceedings of the 24th University Power Engineers Conference*, (Belfast, UK), pp. 373–376, September 1989.

[Brennan *et al.*, 1991] P. Brennan, G. Deffner, D. Lawrence, M. Marics, E. Schwab, and M. Franzke, "Should we or shouldn't we use spoken commands in voice interfaces?," in *Proceedings of Human Factors in Computing Systems (CHI'91)*, (New Orleans, LA), pp. 369–372, ACM, April-May 1991.

[Brunner *et al.*, 1990] H. Brunner, K. Wittenburg, M. Williams, Y. Sekine, S. Dahlgren, and P. Washco, "A snapshot of natural language interfaces (panel)," in *Proceedings of Human Factors in Computing Systems (CHI'90)*, (Seattle, WA), pp. 53–55, ACM, April 1990.

[Carnevale, 1986] F. A. Carnevale, "Computer applications in nursing," *Axon*, vol. 8, no. 1, pp. 269–275, 1986.

[Clark, 1989] M. Clark, "A voice-controlled vocational robotic workstation for people with physical disabilites," *Speech Technology*, vol. 5, pp. 78–80, October-November 1989.

[Cohen, 1990a] B. Cohen, "Emerging technologies in the field of healthcare: enhancing the interface to the medical professional.," *Ann Acad Med Singapore*, vol. 19, no. 5, pp. 627–39, 1990.

[Cohen, 1990b] P. Cohen, "Voice entry in the lab [cytology]," *Comput. Healthc.*, vol. 11, pp. 33–36, March 1990.

[Collet *et al.*, 1989] C. Collet, N. Fumai, M. Petroni, S. Malowany, J. Panisset, A. Malowany, F. Carnevale, R. Gottesman, and A. Rousseau, "A patient data management system for an intensive care unit," in *Proceedings of the IEEE Pacific Rim Conference on Communications, Computers, and Signal Processing,* (Victoria, B.C., Canada), pp. 594–597, June 1989.

[Covox, 1990a] Covox Inc., Eugene, OR, *Covox Voice Master Key II,* 2.04 ed., April 1990.

[Covox, 1990b] Covox Inc., Eugene, OR, *Speech File Building Utility,* 1.1b ed., May 1990.

[Cupples and Beek, 1990] E. J. Cupples and B. Beek, "Application of audio/speech recognition for military applications," in *Speech Analysis and Synthesis and Man-Machine Speech Communications for Air Operations,* (Trondheim, Norway), pp. 8/1–8/10, AGARD, May 1990.

[Dalsgaard and Backgaard, 1990] P. Dalsgaard and A. Backgaard, "Recognition of continuous speech using neural nets and expert systems," *Speech Communication,* vol. 9, pp. 509–520, December 1990.

[Dasta, 1990] J. Dasta, "Computers in critical care: opportunities and challenges.," *DICP,* vol. 24, no. 11, pp. 1084–92, 1990.

[Diethrich, 1988] E. B. Diethrich, "Voice recognition and voice synthesis in the cardiac intensive care unit," *Speech Technology,* vol. 4, pp. 46–50, September-October 1988.

[Edmonds, 1990] E. Edmonds, "Human-computer interface evaluation: not user-friendliness but design for operation.," *Med Inf (Lond),* vol. 15, no. 3, pp. 253–60, 1990.

[EMT, 1988] EMTEX, Tempe, Arizona, *The EMTEX System 2000: Cost Savings and Benefit Realization,* 1988.

[Feldman and Stevens, 1990] C. Feldman and D. Stevens, "Pilot study on the feasibility of a computerized speech recognition charting system.," *Community Dent Oral Epidemiol,* vol. 18, no. 4, pp. 213–5, 1990.

[Fellbaum, 1987] K. Fellbaum, "A blind communication system based on speech recognition and speech synthesis," in *Proceedings of the International Conference on Digital Signal Processing,* (Florence, Italy), pp. 679–686, North-Holland, September 1987.

[Ferguson *et al.*, 1989] J. C. Ferguson, C. J. Martin, C. Rayner, and J. R. Mallard, "A computer based system for monitoring heat and fluid balance in severely burned patients," in *Proceedings of the Annual International Conference of the IEEE Engineering in Medicine and Biology Society,* (Seattle, WA), pp. 1982–1983, November 1989.

[Foster and Bryson, 1989] J. Foster and S. Bryson, "Voice recognition for the IBM 7535 robot," in *Proceedings of Energy and Information Technologies in the Southeast (SOUTHEASTCON)*, (Columbia, SC), pp. 759–764, IEEE, April 1989.

[Gibbs, 1989] R. Gibbs, "The present and future medicolegal importance of record keeping in anesthesia and intensive care: the case for automation [see comments]," *J Clin Monit*, vol. 5, no. 4, pp. 251–5, 1989.

[Goldberg *et al.*, 1989] M. Goldberg, J. Robertson, G. Belanger, N. Georganas, J. Mastronardi, Cohn-SfetcuS, R. Dillon, and J. Tombaugh, "A multimedia medical communication link between a radiology department and an emergency department.," *J Digit Imaging*, vol. 2, no. 2, pp. 92–8, 1989.

[Guerrouad, 1989] A. Guerrouad, "Voice control in the surgery room," in *Proceedings of the Annual International Conference of the IEEE Engineering in Medicine and Biology Society*, (Seattle, WA), pp. 904–905 November 1989.

[Hammond and Stead, 1988] W. E. Hammond and W. W. Stead, "Bedside terminals: An overview," *M.D. Computing*, vol. 5, no. 1, pp. 5–6, 1988.

[Hammond *et al.*, 1991a] J. Hammond, H. Johnson, R. Varas, and C. Ward, "A qualitative comparison of paper flowsheets vs a computer-based clinical information system.," *Chest*, vol. 99, no. 1, pp. 155–7, 1991.

[Hammond *et al.*, 1991b] J. Hammond, H. Johnson, C. Ward, R. Varas, R. Dembicki, and E. Marcial, "Clinical evaluation of a computer-based patient monitoring and data management system.," *Heart Lung*, vol. 20, no. 2, pp. 119–24, 1991

[Harrison *et al.*, 1986] J. A. Harrison, G. R. Hobbs, J. R. Howes, and G. R. Wright, "Machine supported dialogue, used in training air traffic controllers," in *Proceedings of the International Conference on Speech Input/Output: Techniques and Applications*, (London, England), pp. 110–113, IEE, 1986.

[Helm *et al.*, 1991] R. Helm, K. Marriott, and M. Odersky, "Building visual language parsers," in *Proceedings of Human Factors in Computing Systems (CHI'91)*, (New Orleans, LA), pp. 105–112, ACM, April-May 1991.

[Hewlett-Packard, 1987] Hewlett-Packard, "When seconds count . . physicians count on data management for critical care decisions," *Advances in Medicine*, vol. 8, no. 4, 1987.

[Hickey *et al.*, 1990] N. Hickey, J. Robertson, and M. Coristine, "Integrated radiologic information system: a radiology multimedia communication system.," *J Thorac Imaging*, vol. 5, no. 1, pp. 77–84, 1990.

[Holtermann *et al.*, 1990] W. Holtermann, M. Knoch, H. Pfeiffer, E. Muller, and H. Lennartz, "Marburger concept for computer-aided acquisition, processing and documentation of patient data in the intensive care unit.," *Int J Clin Monit Comput*, vol. 7, no. 1, pp. 7–13, 1990.

[Ikehira *et al.*, 1990] H. Ikehira, T. Matsumoto, T. Iinuma, T. Yamasaki, K. Fukuhisa, H. Tsunemoto, F. Shishido, Y. Kubo, K. Inamura, and Y. Tateno, "Analysis of bone scintigram data using speech recognition reporting system–data analysis with speech recognition system.," *Radiat Med*, vol. 8, no. 1, pp. 8–12, 1990.

[James *et al.*, 1990] d. James, JC, N. Gantenberg, and G. Hageman, "A sample computer system for physiological data acquisition and analysis.," *Comput Biol Med*, vol. 20, no. 6, pp. 407–13, 1990.

[Joseph, 1989] R. Joseph, "Large vocabulary voice-to-text systems for medical reporting," *Speech Technology*, vol. 4, pp. 49–51, April-May 1989.

[Kari *et al.*, 1990] A. Kari, E. Ruokonen, and J. Takala, "Comparison of acceptance and performance of automated and manual data management systems in intensive care.," *Int J Clin Monit Comput*, vol. 7, no. 3, pp. 157–62, 1990.

[Kernighan and Ritchie, 1988] B. W. Kernighan and D. M. Ritchie, *The C Programming Language*. Englewood Cliffs, NJ: Prentice Hall, 1988.

[King and Smith, 1990] P. King and B. Smith, "An IBM AT based monitoring system with touchscreen input.," *Int J Clin Monit Comput*, vol. 7, no. 2, pp. 107–11, 1990.

[Kitano, 1991] H. Kitano, "ϕdm-dialog: An experimental speech-to-speech dialog translation system," *IEEE Computer*, vol. 24, pp. 36–50, June 1991.

[Kriewall and Long, 1991] T. J. Kriewall and J. M. Long, "Computer-based medical syst ıs," *IEEE Computer Magazine*, vol. 24, pp. 9–12, March 1991.

[Kuhn *et al.*, 1990] K. Kuhn, W. Doster, D. Roesner, P. Kottmann, W. Swobodnik, and H. Ditschuneit, "An integrated medical workstation with a multimodal user interface, knowledge-based user support, and multimedia documents," in *Proceedings of the Thrid Annual IEEE Symposium on Computer-Based Medical Systems*, (Chapel Hill, NC), pp. 469–476, June 1990.

[Kurogi, 1991] S. Kurogi, "Speech recognition by an artificial neural network using findings on the afferent auditory system.," *Biol Cybern*, vol. 64, no. 3, pp. 243–9, 1991.

[Landau *et al.*, 1989] J. A. Landau, K. H. Norwich, and S. J. Evans, "Automatic speech recognition - can it improve the man-machine interface in medical expert systems?," *Int. J. Bio-Med. Comput.*, vol. 24, pp. 111–117, July 1989.

[Laporte, 1989] C. Y. Laporte, "A voice-interactive evaluation system for command functions in military aircraft," *Microcomputer Applications*, vol. 8, no. 1, pp. 20–26, 1989.

[Large, 1990] W. Large, "Calculating haemodynamic parameters and interpreting arterial blood gas samples using a pocket computer.," *Intensive Care Nurse*, vol. 6, no. 4, pp. 196–9, 1990.

[Laurel *et al.*, 1990] B. Laurel, T. Oren, and A. Don, "Issues in multimedia interface design: Media integration and interface agents," in *Proceedings of Human Factors in Computing Systems (CHI'90)*, (Seattle, WA), pp. 133–139, ACM, April 1990.

[Liang and Narayanan, 1988] M. D. Liang and K Narayanan, "Voice-controlled microscope facilitates intricate microsurgical procedures," *Speech Technology*, vol. 4, pp. 52–54, September-October 1988.

[Martin, 1989] G. L. Martin, "The utility of speech input in user-computer interfaces," *International Journal of Man-Machine Studies*, vol. 30, pp. 355–375, 1989.

[Matrouf *et al.*, 1988] K. Matrouf, F. Neel, and J. Mariani, "Task-oriented dialogue system: an application to aeronautics," *J. Acoust.*, vol. 2, pp. 85–93, March 1988.

[Matrouf *et al.*, 1990] K. Matrouf, J. L. Gauvain, F. Neel, and J. Mariani, "An oral task-oriented dialogue for air-traffic controller training," in *Proceedings of SPIE*, (Orlando, FL), pp. 826–837, International Society of Optical Engineers, April 1990.

[Matumoto *et al.*, 1987] T. Matumoto, T. A. Iinuma, Y. Tateno, H. Ikehira, T. Yamasaki, K. Fukuhisa, H. Tsunemoto, F. Shishido, Y. Kubo, and K. Inamura, "Automatic radiologic reporting system using speech recognition," *Medical Progress Through Technology*, vol. 12, pp. 243–257, 1987.

[McIntyre and Nelson, 1989] J. W. R. McIntyre and T. M. Nelson, "Application of automated human voice delivery to warning devices in an intensive care unit: a laboratory study," *International Journal of Clinical Monitoring and Computing*, vol. 6, pp. 255–262, December 1989.

[Microsoft, 1990] Microsoft Corporation, Redmond, WA, *Microsoft C Reference*, 6.0 ed., 1990.

[Moore and Moore, 1986] C. A. Moore and R. D. Moore, "Pilot-aircraft interface: the voice channel," in *Proceedings of the International SPEECH TECH '87: Voice Input/Output Applications Show and Conference*, (London, England), pp. 174–178, May 1986.

[Moser *et al.*, 1990] L. Moser, W. Schramm, and G. Pauser, "PC-based computer monitoring in an intensive care unit," *Anaesthesist*, vol. 39, no. 10, pp. 561–4, 1990.

[Murchie and Kenny, 1988] C. J. Murchie and G. N. C. Kenny, "A comparison of keyboard, light pen and voice recognition as methods of data input," *International Journal of Clinical Monitoring and Computing*, vol. 5, no. 4, pp. 243–246, 1988.

[Nolan-Avila and Shabot, 1987] L. Nolan-Avila and M. Shabot, "Life without computers in the ICU," *Critical Care Nurse*, vol. 7, no. 3, pp. 80–83, 1987.

[Nonogaki and Ueda, 1991] H. Nonogaki and H. Ueda, "Friend21 project: A contruction of 21st century human interface," in *Proceedings of Human Factors in Computing Systems (CHI'91)*, (New Orleans, LA), pp. 407–414, ACM, April-May 1991.

[O'Malley, 1990] M J. O'Malley, "Text-to-speech conversion technology," *IEEE Computer*, vol. 23, pp. 17–23, April 1990.

[O'Shaughnessy, 1987] D. O'Shaughnessy, *Speech Communication: Human and Machine*. Reading, Massachusetts: Addison-Wesley Publishing Company, 1987.

[Perry and Adam, 1991] T. S. Perry and J. A. Adam, "Improving the world's most advanced system," *IEEE Spectrum*, vol. 28, pp. 22–36, February 1991.

[Petroni *et al.*, 1991] M. Petroni, C. Collet, N. Fumai, K. Roger, F. Groleau, C. Yien, A. Malowany, F. Carnevale, and R. Gottesman, "An automatic speech recognition system for bedside data entry in an intensive care unit," in *Proceedings of the Fourth Annual IEEE Symposium on Computer-Based Medical Systems*, (Baltimore, MD), pp. 358–365, May 1991.

[Petzold, 1989] C. Petzold, *Programming the OS/2 Presentation Manager*. Redmond, WA: Microsoft Press, 1989.

[Pittman, 1991] J. A. Pittman, "Recognizing handwritten text," in *Proceedings of Human Factors in Computing Systems (CHI'91)*, (New Orleans, LA), pp. 271–275, ACM, April-May 1991.

[Quedens and Beacons, 1990] G. Quedens and P. S. Beacons, *Developing Presentation Manager Applications: an Introduction*. Glenview, Illinois: Scott, Foresman, and Company, 1990.

[Rabiner, 1989] L. R. Rabiner, "A tutorial on hidden markov models and selected applications," *Proceedings of the IEEE*, vol. 77, pp. 257–286, February 1989.

[Reddy, 1976] D. R. Reddy, "Speech recognition by machine: a review," *Proceedings of the IEEE*, vol. 56, pp. 501–531, 1976.

[Robbins *et al.*, 1987] A. H. Robbins, D. M. Horowitz, M. K. Srinivasan, M. E. Vincent, K. Shaffer, and N. L. Sadowsky, "Speech-controlled generation of radiology reports," *Radiology*, vol. 164, pp. 569–573, August 1987.

[Roukos, 1989] S. Roukos, "Integrating speech and natural language," in *Proceedings of a Workshop on Speech and Natural Language*, (Philadelphia, Pa), pp. 21–23, Morgan Kaufmann, February 1989.

[Salisbury and Chilcote, 1989] M. Salisbury and J. Chilcote, "Investigating voice i/o for the airborne warning and control system (awacs)," *Speech Technology*, vol. 5, pp. 50–55, October-November 1989.

[Salisbury *et al.*, 1990] M. W. Salisbury, J. H. Hendrikson, T. L. Lammers, C. Fu, and S. A. Moody, "Talk and draw: Bundling speech and graphics," *IEEE Computer*, vol. 23, pp. 59–65, August 1990.

[Schneiderman, 1987] B. Schneiderman, *Designing the User Interface: Strategies for Effective Human–Computer Interaction*. Reading, Massachusetts: Addison-Wesley Publishing Company, 1987.

[Schwid *et al.*, 1990] H. Schwid, C. Olson, P. Wright, and P. Freund, "Microcomputer-based data acquisition system for clinical research.," *J Clin Monit*, vol. 6, no. 2, pp. 141–6, 1990.

[Shozo, 1990] A. Shozo, "Speech input for meat inspection and pathological coding used thereby," *Joho Kari*, vol. 33, pp. 619–626, October 1990.

[Smith *et al.*, 1990] M. F. Smith, R. J. Jaszczak, C. E. Floyd Jr., K. L. Greer, and R E. Coleman, "Interactive visualization of three-dimensional spect cardiac images," in *Proceedings of the Thrid Annual IEEE Symposium on Computer-Based Medical Systems*, (Chapel Hill, NC), pp. 213–219, June 1990.

[Soligen and Shabot, 1988] S. Soligen and M. M. Shabot, "A 32 key keyboard for the hp pdms," *Int J Clin Monit Comput*, vol. 5, no. 1, 1988.

[Southerton, 1989] A. Southerton, *Advanced OS/2 Presentation Manager Programming: The Graphics Programming Interface*. Reading, Massachusetts: Addison-Wesley Publishing Company, 1989.

[Stickland Jr., 1991] T. J. Stickland Jr., "Development of an information system to assist management of critically ill patients," in *Proceedings of the Fourth Annual IEEE Symposium on Computer-Based Medical Systems*, (Baltimore, MD), pp. 70–77, May 1991.

[Tachakra *et al.*, 1990] S. Tachakra, D. Potts, and A. Idowu, "Evaluation of a computerized system for medical records in an accident and emergency department.," *Int J Clin Monit Comput*, vol. 7, no. 3, pp. 187–91, 1990

[Verbex, 1990] VERBEX Voice Systems, Edison, NJ, *VERBEX Conversational Voice Input/Output System Grammar Development Manual*, 2.00 ed., May 1990.

[Zhang *et al.*, 1990] J. Zhang, M. Levesque, C. Wilson, R. Harper, J. Engel, J, R. Lufkin, and E. Behnke, "Multimodality imaging of brain structures for stereotactic surgery.," *Radiology*, vol. 175, no. 2, pp 435–41, 1990.

Appendix A Grammar Definition File for the Fluid Balance Module

This appendix contains the grammar file for the Fluid Balance module's Main and Ingesta windows in *Verbex Standard Notation (VSN)* [Verbex, 1990]. The grammar file contains many grammar definitions, or sub-vocabularies, denoted by the `!grammar-definition-name=`. A grammar definition begins with a `#REC` (RECognition) section which consists of two subsections, a `#G` (Grammar) and a `#TR` (TRanslation) section. The `#Grammar` section contain the words or phrases that the recognizer should listen for. These are the words and phrases that will be trained by the user. The grammar section can also contain a *structure*, which begins with a period ".", which is a way of representing a list of words with a name in order to keep the grammar compact.

The `#TR` (translation) section contains the information that will be sent to the host computer for each word recognized. If a word appearing in the grammar definition does not appear in the translation section, the recognizer will send the words to the host computer, exactly as they appear in the `#G` (grammar) section.

The last section in the grammar definition is the `#N` (next) section. If more than one grammar definition is to be used in a given application, the `#N` (next) section allows the recognizer to switch to other grammar definitions when certain words or phrases are recognized. Multiple grammar definitions are useful when the grammar can be logically devided and when common words require multiple output translations. For the Fluid Balance module grammar, both are true. Since the module has a "menu tree", as previously mnetioned, it is advantageous to divide the grammar definitions to follow the syntactic structure of the menu tree. When certain words are recognized, the recognizer switches to other grammar definitions. This can be considered as being the "recognizer equivalent" of displaying submenus.

Since some Fluid Balance module menus have common items in their submenus but which must have unique identifiers, using multiple grammars allows the same words to have different output translations depending from which submenu in the tree it was called from.

Note that the grammar file only includes the grammar for the Main window and for the Ingesta window.

Jul 18 1991 10 16 39 **fbgram2.grm** Page 1

```
;==================================================================
; Fluid Balance Module Grammar (Main and Ingesta Only).
;==================================================================

;Created by Marco Petroni: May  9, 1991.
;Last Modified :           May 21, 1991.

; Global definitions

; NONE

#vocab=FBS_GRM
'mainmenu=
   #REC
        #G
                ingesta
                excreta
                echo
                settings
                help

        #TR
               ingesta          "1"
               excreta          "2"
               echo             "4"
               settings         "7"
               help             "6"


        #N
                ingesta  >       ingestamenu
                excreta  >       mainmenu
                echo     >       echomenu
                settings >       settmenu

;Grammar definition for the Ingesta Sheet (ingestamenu):
'ingestamenu=
   #REC
        #G
                i_v .DIGIV
                oral_gastric
                time
                correction
                save
                clear
                help
                exit
                quit
                main

               .DIGIV =
                        1
                        2
                        3
                        4
                        5

            #TR
                i_v             ||
                oral_gastric    "160"
                time            "180"
                correction      "170"
                save            "220"
                clear           "210"
                1               "110"
                2               "120"
```

Jul 18 1991 10:16 39 **fbgram2.grm** Page 2

```
                        3               "130"
                        4               "140"
                        5               "150"
                        exit            "190"
                        quit            "190"
                        main            "999"

                #N
                        i_v 1           >       ivmenu1
                        i_v 2           >       ivmenu2
                        i_v 3           >       ivmenu3
                        i_v 4           >       ivmenu4
                        i_v 5           >       ivmenu5
                        oral_gastric    >       ogmenu
                        correction      >       cormenu
                        save            >       savemenu
                        clear           >       clearmenu
                        time            >       timemenu
                        main            >       mainmenu

;Grammar Definition for the IV menus...

'ivmenu1=
    #REC
        #G
                level
                comment
                location
                name
                actual_intake
                desired_intake
                escape

        #TR
                level           "111"
                comment         "112"
                location        "113"
                name            "114"
                actual_intake   "115"
                desired_intake  "116"
                escape          "esc"

        #N
                level           >       levelmenu
                comment         >       commentmenu
                location        >       locationmenu1
                name            >       namemenu
                actual_intake   >       levelmenu
                desired_intake  >       levelmenu
                escape          >       ingestamenu

'ivmenu2=
    #REC
        #G
                level
                comment
                location
                name
                actual_intake
                desired_intake
                escape

        #TR
                level           "121"
                comment         "122"
                location        "123"
                name                "124"
                actual_intake       "125"
                desired_intake      "126"
                escape              "esc"

        #N
                level               >       levelmenu
                comment             >       commentmenu
                location            >       locationmenu2
                name                >       namemenu
                actual_intake       >       levelmenu
                desired_intake      >       levelmenu
                escape              >       ingestamenu

'ivmenu3=
    #REC
        #G
                level
                comment
                location
                name
                actual_intake
                desired_intake
                escape

        #TR
                level               "131"
                comment             "132"
                location            "133"
                name                "134"
                actual_intake       "135"
                desired_intake      "136"
                escape              "esc"

        #N
                level               >       levelmenu
                comment             >       commentmenu
                location            >       locationmenu3
                name                >       namemenu
                actual_intake       >       levelmenu
                desired_intake      >       levelmenu
                escape              >       ingestamenu

 ivmenu4=
    #REC
        #G
                level
                comment
                location
                name
                actual_intake
                desired_intake
                escape

        #TR
                level               "141"
                comment             "142"
                location            "143"
                name                "144"
                actual_intake       "145"
                desired_intake      "146"
                escape              "esc"
        #N
                level               >       levelmenu
                comment             >       commentmenu
                location            >       locationmenu4
```

```
                name                >       namemenu
                actual_intake       >       levelmenu
                desired_intake      >       levelmenu
                escape              >       ingestamenu

'ivmenu5=
    #REC
        #G
                level
                comment
                location
                name
                actual_intake
                desired_intake
                escape

        #TR
                level               "151"
                comment             "152"
                location            "153"
                name                "154"
                actual_intake       "155"
                desired_intake      "156"
                escape              "esc"

        #N
                level               >       levelmenu
                comment             >       commentmenu
                location            >       locationmenu5
                name                >       namemenu
                actual_intake       >       levelmenu
                desired_intake      >       levelmenu
                escape              >       ingestamenu

;Grammar Definition for Level Dialog Boxes (levelmenu)

 levelmenu=
    #REC
        #G
                level_is  DIGIT @1,4 enter
                escape
                delete
                ok
                correction

        #TR

                level_is
                enter
                escape              "esc"
                delete              "del"

        #N
                escape              >       ingestamenu
                ok                  >       ingestamenu

;Grammar Definition for Comment Dialog Box (commentmenu)

 commentmenu=
    #REC
        #G
                [illegible]
                [illegible]
                delete
                ok
                correction
                escape

        #TR
                cut                 "Cut for Blood"
                cfp                 "Cut for CFP"
                delete              "del"
                escape              "esc"

        #N
                ok                  >       ingestamenu
                escape              >       ingestamenu

;Grammar Definition for Location Submenus in IV1 - IV5 (locationmenu)

'locationmenu1=
    #REC
        #G
                rightarm
                leftarm
                rightleg
                leftleg
                chest
                escape

        #TR
                rightarm            "1131"
                leftarm             "1133"
                rightleg            "1132"
                leftleg             "1134"
                chest               "1135"
                escape              "esc"


        #N
                >       ingestamenu

'locationmenu2=
    #REC
        #G
                rightarm
                leftarm
                rightleg
                leftleg
                chest
                escape

        #TR
                rightarm            "1231"
                leftarm             "1233"
                rightleg            "1232"
                leftleg             "1234"
                chest               "1235"
                escape              "esc"

        #N
                >       ingestamenu

'locationmenu3=
    #REC
        #G
                rightarm
                leftarm
                rightleg
                leftleg
```

```
                chest
                escape

        #TR
                rightarm            "1331"
                leftarm             "1333"
                rightleg            "1332"
                leftleg             "1334"
                chest               "1335"
                escape              "esc"

        #N
                >       ingestamenu

'locationmenu4=
    #REC
        #G
                rightarm
                leftarm
                rightleg
                leftleg
                chest
                escape

        #TR
                rightarm            "1431"
                leftarm             "1433"
                rightleg            "1432"
                leftleg             "1434"
                chest               "1435"
                escape              "esc"

        #N
                >       ingestamenu

'locationmenu5=
    #REC
        #G
                rightarm
                leftarm
                rightleg
                leftleg
                chest
                escape

        #TR
                rightarm            "1531"
                leftarm             "1533"
                rightleg            "1532"
                leftleg             "1534"
                chest               "1535"
                escape              "esc"

        #N
                >       ingestamenu


;Grammar Definition for IV Name Dialog Box (namemenu)

 namemenu=
    #REC
        #G
                Actiprophen
                ASA
                ART
                LA
                DSW
                ok
                delete
                escape

        #TR
                Actiprophet     "ACT"
                delete          "del"
                escape          "esc"

        #N
                ok              >       ingestamenu
                escape          >       ingestamenu

;Grammar Definition for Ogal Gastric on Ingesta Menu Bar (ogmenu)

'ogmenu=
    #REC
        #G
                type
                level
                amount
                escape

        #TR
                type            "161"
                level           "162"
                amount          "163"
                escape          "esc"

        #N
                type            >       typemenu
                level           >       levelmenu
                amount          >       levelmenu
                escape          >       ingestamenu

;Grammar Definition for Type Sub-Item of Oral Gastric (typemenu)

'typemenu=
    #REC
        #G
                plasma
                PRGC
                FFP
                PRBG
                ok
                escape
                delete

        #TR
                plasma          "Plas"
                escape          "esc"
                delete          "del"

        #N
                escape          >       ingestamenu
                ok              >       ingestamenu

;Grammar Definition for Correction Menu Item of Ingesta Menu Bar (cormenu)

 cormenu=
    #REC
        #G
```

```
                time
                i_v DIGIV
                escape

        #TR
                i_v             ||
                time            "180"
                1               "110"
                2               "120"
                3               "130"
                4               "140"
                5               "150"
                escape          "esc"

        #N
                i_v 1           >       ivmenu1
                i_v 2           >       ivmenu2
                i_v 3           >       ivmenu3
                i_v 4           >       ivmenu4
                i_v 5           >       ivmenu5
                time            >       timemenu
                escape          >       ingestamenu

;Grammar definition for Save Dialog Box (savemenu)

'savemenu=
    #REC
        #G
                yes
                no

        #N
                >       ingestamenu

;Grammar definition for Clear Dialog Box (clearmenu)

'clearmenu=
    #REC
        #G
                yes
                no

        #N
                >       clearsavemenu

 clearsavemenu=
    #REC
        #C
                ok
                cancel

        #N
                >       ingestamenu

;Grammar definition for Time Item in Ingesta Menu Bar (timemenu)

 timemenu=
    #REC
        #C
                time_is DIGIT32 hours DIGIT32 minutes enter
                correction
                escape
                delete
                ok

                .DIGIT=
                0
                1
                2
                3
                4
                5
                6
                7
                8
                9


        #TR
                |||
                time_is         ||
                hours           ":"
                minutes         ||
                enter           ||
                escape          "esc"
                delete          "del"

        #N
                escape          >       ingestamenu
                ok              >       ingestamenu

;Grammar Definition for Echo Sub-Menu (echomenu)

'echomenu=
    #REC
        #G
                on
                off
                escape

        #TR
                on      "40"
                off     "41"
                escape  "esc"

        #N
                >       mainmenu

;Grammar Definition for Settings Menu (settings)

'settmenu=
    #REC
        #G
                bed
                id
                name
                escape

        #TR
                bed     "71"
                id      "72"
                name    "73"
                escape  "esc"

        #N
                escape          >       mainmenu
                bed             >       bedmenu
                name            >       pnamemenu
```

```
                id              >       idmenu

 bedmenu=
    #REC
        #G
                number .DIGIT@1,2 enter
                ok
                escape
                delete

        #TR
                |||
                number  ||
                enter   ||
                escape  "esc"
                delete  "del"

       #N
                ok      >       mainmenu
                escape  >       mainmenu

'idmenu=
    #REC
       #G
                id .DIGIT@1,3 enter
                ok
                escape
                delete


       #TR
                |||
                id      ||
                enter   ||
                delete  "del"
                escape  "esc"

       #N
                ok      >       mainmenu
                escape  >       mainmenu


'pnamemenu=
    #REC
       #G
                jones
                smith
                doe
                ok
                escape
                delete

       #TR
                jones   "JONES, Barbara"
                smith   "SMITH, John"
                doe     "DOE, Jane"
                escape  "esc"
                delete  "del"

       #N
                ok      >       mainmenu
                escape  >       mainmenu
```