Proceedings of the 2000 Winter Simulation Conference
J. A. Joines, R. R. Barton, K. Kang, and P. A. Fishwick, eds.

A VOICE ASSISTED SIMULATION-ANIMATION ARCHITECTURE

Raymond L. Smith, III

International Business Machines Corporation
3039 Cornwallis Road
Research Triangle Park, NC 27709, U.S.A.

Stephen D. Roberts

Department of Industrial Engineering
Campus Box 7906
North Carolina State University
Raleigh, NC 27695-7906, U.S.A.

ABSTRACT

This paper introduces a software architecture that has been used to enable voice assistance for a simulation-animation environment by integrating technologies that recognize spoken language input and generate spoken language output. Voice assisted technology has several features which make user navigation within complex software applications easier than traditional methods, such as key-typed commands or mouse manipulation. While this environment might be more friendly to an end user, several challenges exist to a developer tasked with integrating these extremely diversity technologies into a single software architecture that must operate with computational efficiency. We present the requirements and design for a proposed software architecture, referred to as the Voice Assisted Simulation-Animation Architecture (VASArch), that attempts to address these problems. We also present the implementation of a prototype for simulating a single-server system with exponentially distributed customer interarrival and service times, called VASArch(M/M/1), which was developed to demonstrate the feasibility of the proposed software architecture. The prototype offers a user the ability to interact with the simulation model environment by providing input through spoken commands, mouse manipulation, and keyboard entry. In addition, the prototype provides output, which includes statistical information, in spoken and visual form for user examination.

1 INTRODUCTION

Research in the area of Human-Computer Interaction (HCI) attempts to enhance the effectiveness and efficiency with which work and other activities are performed at the human-computer interface, while promoting desirable human values (Dix 1998). As a result, one technology offering promise for development of interactive applications in the future is conversational speech systems, which includes speech synthesis and speech recognition technologies (Bernsen 1998). The quest to communicate with machines using speech as the interface medium can be explained by the inherent advantages associated with spoken language. Human speech is perceived as being both natural and ubiquitous across the human population, and the development of communication with machines through speech has frequently been viewed as a natural progression in human-machine interaction (Roe 1994).

At present, applications of speech technology are used in manufacturing and inspection processes, aids for the sight- and mobility-impaired, aviation warning systems, telephone transaction systems, medical records dictation, and voice messaging system (Baber 1993). Rapid expansion of speech systems into new applications can be attributed to recent increases in recognition rates and memory capacity, decreases in cost, and the realized advantages these types of systems enjoy over more conventional input and output systems. As speech technologies and application design improve, conversational speech systems will begin to revolutionize the human-computer interface by allowing unrestricted, virtually error-free, input and retrieval of data (Taylor 1990).

In this paper we address the software architecture for a simulation-animation environment integrating technologies that recognize spoken language input and generate spoken language output. The merger of these diverse technologies, which possess drastically different computational demands and durations, introduces considerable complexity that the software architecture must address. Furthermore, since the application should support interactive behavior under real-time conditions, a number of activities must be performed which cannot be interrupted in the midst of execution. As a result, the architecture design must provide the ability to manage and control the execution of multiple activities, or tasks, within an application constrained by a single-processor computing environment. Throughout the remainder of this paper, we commonly refer to the definition of the proposed software architecture as the Voice Assisted Simulation-Animation Architecture (VASArch).

2 A SOFTWARE ARCHITECTURE DESIGN

The VASArch software architecture design includes (1) the system components, (2) the conceptual design, (3) a multiprogramming approach, and (4) the logical design.

2.1 System Components

The VASArch software architecture depends upon five major system components: (1) the Application Controller, (2) the Graphical User Interface, (3) the Simulation-Animation, (4) the Speech Recognition, and (5) the Speech Synthesis.

The Application Controller (AC) component is responsible for managing the activities of all other system components. It coordinates vital computer resources while promoting a unified and interactive presentation to the application user.

The Graphical User Interface (GUI) component is responsible for visually rendering a presentation window for the application and managing the display of information representative of the activities performed by the system components. This responsibility also includes management of tactile information received through keyboard or mouse.

The Simulation-Animation component is responsible for the execution of a simulation model representing a complex system and the collection of statistical information. It is also responsible for the display of a graphical presentation illustrating the dynamics of the transactions within the modeled system.

The Speech Recognition component is responsible for the capture of spoken utterances and real-time recognition processing to convert an utterance into decoded text. In this particular application we require technology that supports (1) speaker-independence, (2) continuous speech, and (3) a medium size vocabulary. We recommend use of the IBM Continuous Speech Series (ICSS) product to fulfill this requirement.

The Speech Synthesis component is responsible for producing natural sounding, continuous speech output. To fulfill this requirement, speech will be produced by dynamically concatenating prerecorded wave forms selected from a repository into an ordered play list.

2.2 VASArch Conceptual Design

The conceptual design model, shown in Figure 1, describes how the major system component functions should be organized and assembled together. Figure 1 carefully

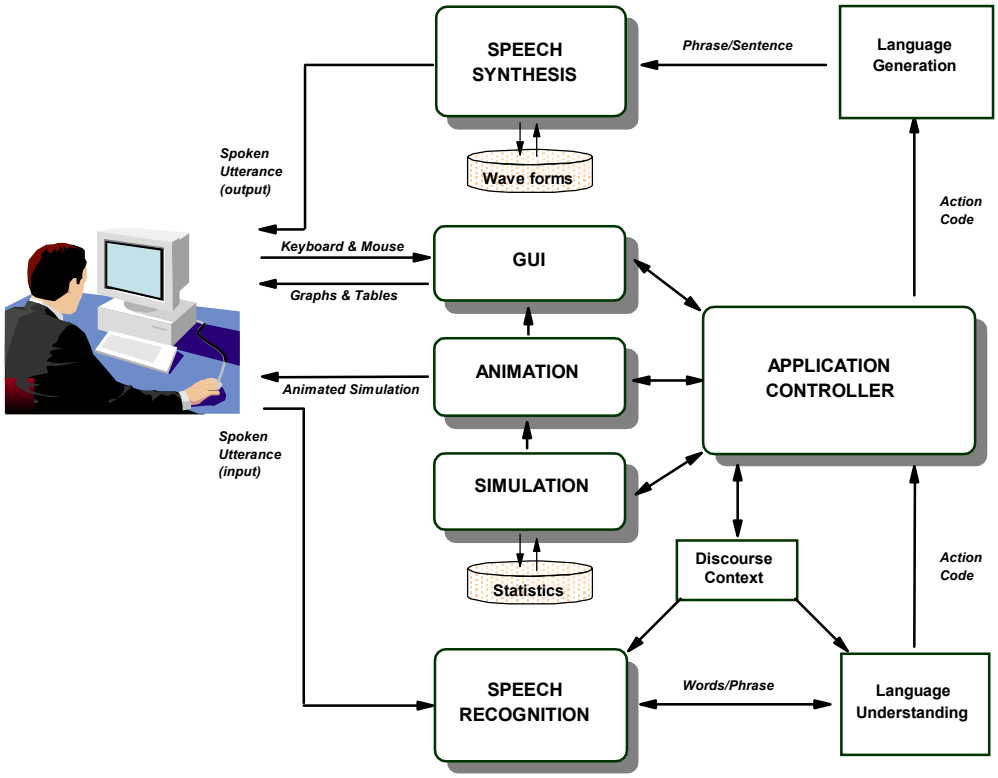

Figure 1: VASArch Conceptual Design

illustrates how information will be exchanged between the user and the major system components. In this particular case, the conceptual design model focuses on the creation of a multi-modal interface that will enable the free exchange of information between user and machine. This multi-modal interface will specifically offer the ability to interact with the simulation model through input received by keyboard, mouse, or spoken command. Users may select the input communication medium that best serves their need, or preference. Output from the application may be presented to the user in the form of a visual display, animation, or spoken language. The design will allow any combination of these methods to be used.

2.3 A Multiprogramming Approach

Traditional interactive speech systems perform their activities in a defined sequence of events. Users are prompted for their input, they speak their request, the system processes this request, some action is performed by the system, and a spoken utterance is generated. Once a response is given to the user almost no other activity is performed, except to wait for the next spoken command. In contrast, VASArch will execute an animated simulation continuously during the period waiting for a spoken request, while a spoken request is processed, and during the response to a spoken request. Events that occur as a result of a simulation event, or caused by a user event, may also be responded to either visually or through spoken output. This overlap in multi-modal input and output requires the execution of tasks in parallel (Dix 1998).

Multiprogramming techniques, particularly multitasking and multithreading capabilities, offer the ability to exploit the parallelism that exists between independent activities within the VASArch application (Stock 1995). This parallelism, in essence, reflects the division of work that must be performed. Multitasking and multithreading combined offer substantial benefit in situations where relatively independent activities occur which possess distinctly different computational intensities and durations. Typically, activities that have large differences in timing are more likely to be isolated in their operation from other threads of execution (Dorfman 1994, Kogan 1994). Based on these characteristics, the activities identified with the major system component functions are organized into six threads of execution within a single process. The corresponding threads include (1) the application controller, (2) the graphical user interface, (3) the simulation, (4) the animation, (5) the speech recognition, and (6) the speech synthesis.

The coordination of activities and allocation of computing resources between the threads is necessary to execute the application. Under this multiprogramming design, synchronization and communication protocols at the operating system level provide the ability to effectively manage the multiple threads of execution.

2.4 VASArch Logical Design

The logical design, shown in Figure 2, describes the flow of activities between the functional subcomponents located within each system component. More importantly, these flows of activities begin to reveal the relationships that must be established and maintained between the major system components.

The description of the logical design architecture has been arranged into five sub-sections that correspond to the major system components. The presentation order follows the general progression of activities through the application.

2.4.1 Speech Recognition

The Speech Recognition system component performs a series of activities to convert a speech utterance into decoded text, as illustrated in Figure 2. First, the *analog to digital conversion* activity uses a microphone connected to the line input of the audio adapter to capture the speech input signal (utterance) into a format which can later be processed. The higher sampling rate improves recognition accuracy; however, this results in increased computational load and requires more memory for the internal speech data buffers (IBM 1993). Second, once a speech utterance has been captured into digital form it undergoes *feature extraction,* which is performed in a two-stage sequence known as *data conditioning and rate conversion*, and *vector-quantization*. *Data conditioning and rate conversion* are pre-process activities responsible for converting speech data from an external data format to an internal data format. *Vector-quantization* then compresses the internal data format from either 180 kilobits or 360 kilobits per second to only 3.2 kilobits per second producing 100 feature vectors for each second of digitized speech. Third, following vector-quantization the recognition engine performs the actual recognition of the speech signal through a *beam search* using the feature vectors, context word pair grammars and acoustic phoneme models. Finally, the *ICSS Application Programming Interface (API)* allows the client application to communicate with the speech recognition engine to control context loading, context switching, runtime parameters, data acquisition and return of decoded text. The Speech Recognition system component has responsibility for communicating with the recognition server through the ICSS APIs. It also has responsibility for communicating results to the system components through the Application Controller.

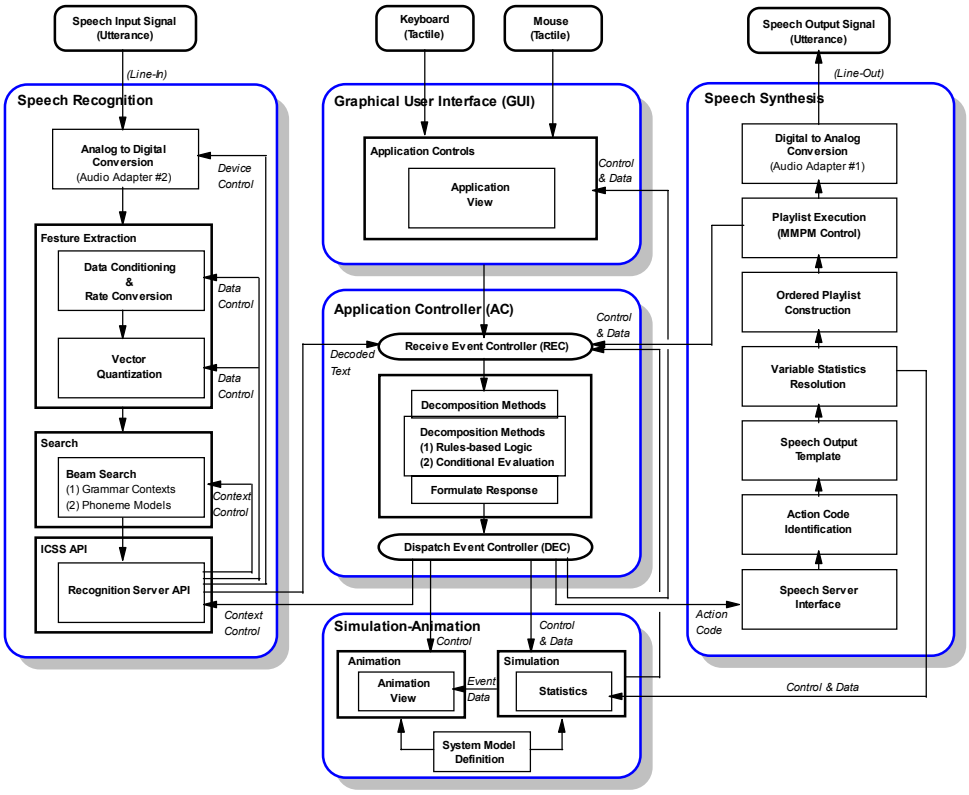

Figure 2: VASArch Logical Design

2.4.2 Graphical User Interface

The Graphical User Interface system component manages input received from tactile sources and generates display of the *application view*. In order to accomplish these activities we have utilized the facilities provided under OS/2's Presentation Manager environment (Burge 1993, IBM 1992, Petzold 1994). As Figure 2 illustrates, the graphical user interface first defines and sets up the controls for handling of keyboard and mouse input to be associated with the application view. Second, a graphical presentation is created for the application view and displayed to the user's screen. Third, the graphical user interface initiates handling methods for user events generated by input controls associated with the application view and those dispatched by the Application Controller to reflect changes introduced from other system components. This includes provision for the update and redraw of the graphical presentation for the application view. Lastly, the graphical user interface system component may selectively transmit information to the Application Controller regarding user events that impact other system components. The Application Controller then engages the respective system components needed to respond to the event.

2.4.3 Application Controller

The Application Controller system component coordinates overall program flow of control through a series of activities which receives input requests from all system components, decomposes the requests into meaningful associations, processes these associations into actionable requests and dispatches to the applicable system components. First, requests in the form of a system event are transmitted by a system component and received by the *Receive Event Controller* (REC) located within the Application Controller. The REC classifies the incoming system event type to determine the appropriate decomposition method. Event types related to the speech recognition process are more complex and require advanced methods to decompose the decoded text into meaningful associations. The associations are determined through traditional rule-based reasoning logic and conditional evaluations. Event types not involved with the speech recognition process require similar, but less sophisticated decomposition methods in which the system events are translated into an activity matrix for conditional evaluation. Information resulting from both event types is then brought together and evaluated using traditional rule-based logic in order to generate the application controller's response, which consists of an action code assignment and impacted system components identification. Finally, the

Dispatch Event Controller (DEC) communicates the action codes and synchronizes activity with the identified system components.

2.4.4 Simulation-Animation

The Simulation-Animation system component performs a series of activities that characterizes simulated transactions within a modeled system in graphical animated representation. As illustrated in Figure 2, the system component consists of two separate subcomponents represented as *simulation* and *animation*. Both the simulation and animation subcomponents are dependent on the *system model definition* that is created and stored externally. Using this definition the simulation subcomponent sets up the simulated system, initializes predefined parameters, and notifies the animation subcomponent to create the animation view. The animation subcomponent uses the Graphical Programming Interface (GPI) facilities to create an animation view derived, but separate, from the *GUI application view* (Knight 1995). The animation view displays the animated representation of the system model. The simulation subcomponent then initiates handling of incoming control and data dispatched from the Application Controller. When a simulation event occurs the simulation subcomponent records statistical information and communicates the updated event and state information to the animation subcomponent. Since the animation view is derived from the application view, presentation control messages from the graphical user interface are received through the Application Controller. The simulation subcomponent handles all simulation events and has principal responsibility for communications with the Application Controller when an event impacts other system components. In addition to these primary activities, the Speech Synthesis system component may on occasion require access to state and statistical information to resolve an unknown variable in an outbound speech utterance.

2.4.5 Speech Synthesis

The Speech Synthesis system component performs a series of activities to convert an action code assigned by the Application Controller into a computer generated speech utterance, as illustrated in Figure 2. First, the *speech server interface* receives a request dispatched from the Application Controller to generate a speech utterance. Second, the speech server interface then interprets the request by performing an indexed reference lookup for the action code to locate the appropriate *speech output template*. The speech output template identifies a combination of fixed and variable messages that must be assembled to generate a meaningful speech utterance. In a majority of cases the speech output template may consist only of fixed messages that can be produced simply through the play back of a single waveform. However, when variable messages are encountered the process becomes more complex since the Speech Synthesis system component must retrieve state or statistical information from the simulation subcomponent. Once information for the entire speech output template has been populated, representative waveforms are retrieved and concatenated within *ordered playlist construction*. Finally, an outbound speech utterance is generated when the play list is sent to the audio adapter connected through the line output to a pair of acoustical speakers for digital to analog conversion. The audio adapter for playback must be properly configured with device-drivers setup under the MMPM environment. The Speech Synthesis system component notifies the Application Controller once a request has been completed

3 THE VASARCH(M/M/1) PROTOTYPE

We developed a prototype to demonstrate the feasibility of constructing an operational application based on the proposed software architecture. In this prototype, we use the M/M/1 queuing system as the system modeled for the simulation and animation. Thus, we commonly refer to the prototype as VASArch(M/M/1) throughout the remainder of the paper. The prototype has served as an invaluable resource for experimentation purposes to identify potential design problems, determine possible improvements, challenge performance issues and verify stability of the proposed software architecture. Both the prototype and the proposed software architecture have evolved through an iterative series of modifications and enhancements based on its performance and results.

3.1 Prototype Assumptions

A few simplifying assumptions were made to the software architecture design since considerable investment in time and resources would otherwise be required to develop a comprehensive prototype. The modifications applied do not reduce core application function, but do limit the scalability and presentation quality. The most notable modifications include (1) the consolidation of the graphical user interface and animation functions; (2) the use of bitmap images within the animation rather than more sophisticated graphics capabilities; (3) the reports and graphs that display statistical information are simulated presentations that do not make use of actual statistical data that is captured; (4) the configuration of the model simulated cannot be readily changed to represent more complex systems.

3.2 Prototype Construction

The operating system requirements and technical specifications are dependent primarily on a strong foundation for program control in multiprogramming environments. The requirements and technical specifications can be summarized in five dependencies that must be supported by the operating system. They include the ability to (1) create and support multiple threads of execution; (2) manage multiple task, as represented by multiple threads originating from a single process, over one or more processors; (3) preempt a task with possession of a specific processor for immediate reallocation to a prioritized task; (4) allow multiple threads access to the same portion of code with stable performance and memory space protection; and (5) manipulate large memory objects.

At the time of our research, the operating system that best met these criteria, with proven results, was IBM's OS/2. IBM OS/2 Warp had well demonstrated its stability and performance through wide acceptance in the business place with use in real-time, mission-critical applications, such as financial banking, logistics management and production operations (Kogan 1994). Furthermore, the development tools available were considered well developed and robust. Therefore, we selected IBM OS/2 as the operating system on which to develop the VASArch(M/M/1) prototype.

The code released for the prototype was prepared using the IBM VisualAge C++ for OS/2 Programming Tool Kit v3.0. We enabled speech recognition capability by integrating features of the IBM Continuous Speech Series (ICSS) for OS/2 product into the VASArch prototype. We enabled speech generation capability by integrating customized features provided under the IBM Multimedia Program Manager (MMPM). We enabled animation capability through use of the IBM Presentation Manager (PM) environment. The simulation model used to drive the prototype was written in a object-oriented C++ design customized specifically for the M/M/1 queuing system.

The hardware used to develop and operate the VASArch prototype consisted of a standard multimedia personal computer configured with an Intel P5-100 Pentium microprocessor and 32 megabytes of memory installed. Multimedia features require the ability to receive audio input through a microphone, and produce audio output through a speaker concurrently using independent channels. Since at the time of our research most audio adapters were incapable of performing both duties concurrently, the personal computer was equipped with a second audio adapter; one to maintain the audio output channel, and one to maintain the audio input channel. Figure 3 illustrates the computer hardware configuration for the audio adapters, microphone, and digital stereo

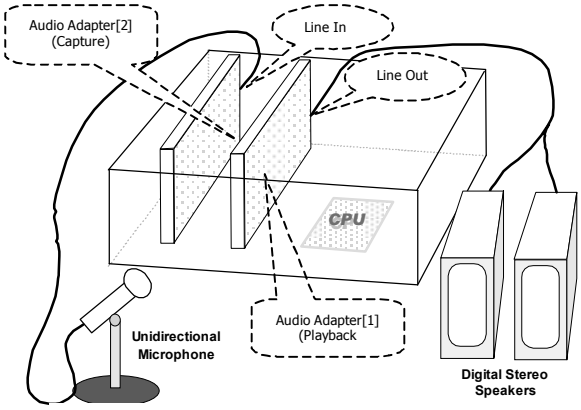

Figure 3: Computer Hardware Configuration

speaker. The audio adapter used to perform speech synthesis was a Creative Labs SoundBlaster 16 ISA ASP adapter (ordinal device 1). A pair of digital stereo speakers was connected to this adapter through its speaker output line connector. The audio adapter used to perform speech recognition was an IBM Audio Capture Playback Adapter ISA adapter (ordinal device 2). A unidirectional microphone was connected to this adapter through its microphone input line connector. In the future, we anticipate the introduction of audio adapters that will provide full duplex capability in a single audio adapter.

3.3 Modeled System

In the M/M/1 queuing system, customers arrive at a single-server service station in accordance with a Poisson process — that is, the times between successive arrivals are independent exponential random variables. Upon arrival each customer goes directly into service if the server is available and, if not, the customer joins the queue. When the server finishes serving a customer, the customer leaves the system, and the next customer in queue, if there is any, enters service. The successive service times are assumed to be independent exponential random variables.

Use of the M/M/1 queuing system model provides three adjustable parameters within the simulation. First, the arrival rate may be varied to control the frequency at which customers enter the system. Second, the service rate may be varied to control the rate at which customers are served by the resource once in the system. Last, the state of the server resource may be changed between an operational, or inoperative state.

Figure 4 provides a screen capture from the simulation-animation browser. The animation illustrates the M/M/1 queuing system model with the depiction of customers waiting in queue, a customer in service and an active server resource. While in an operational state, the server resource may be located in one of two positions in the animation, idle

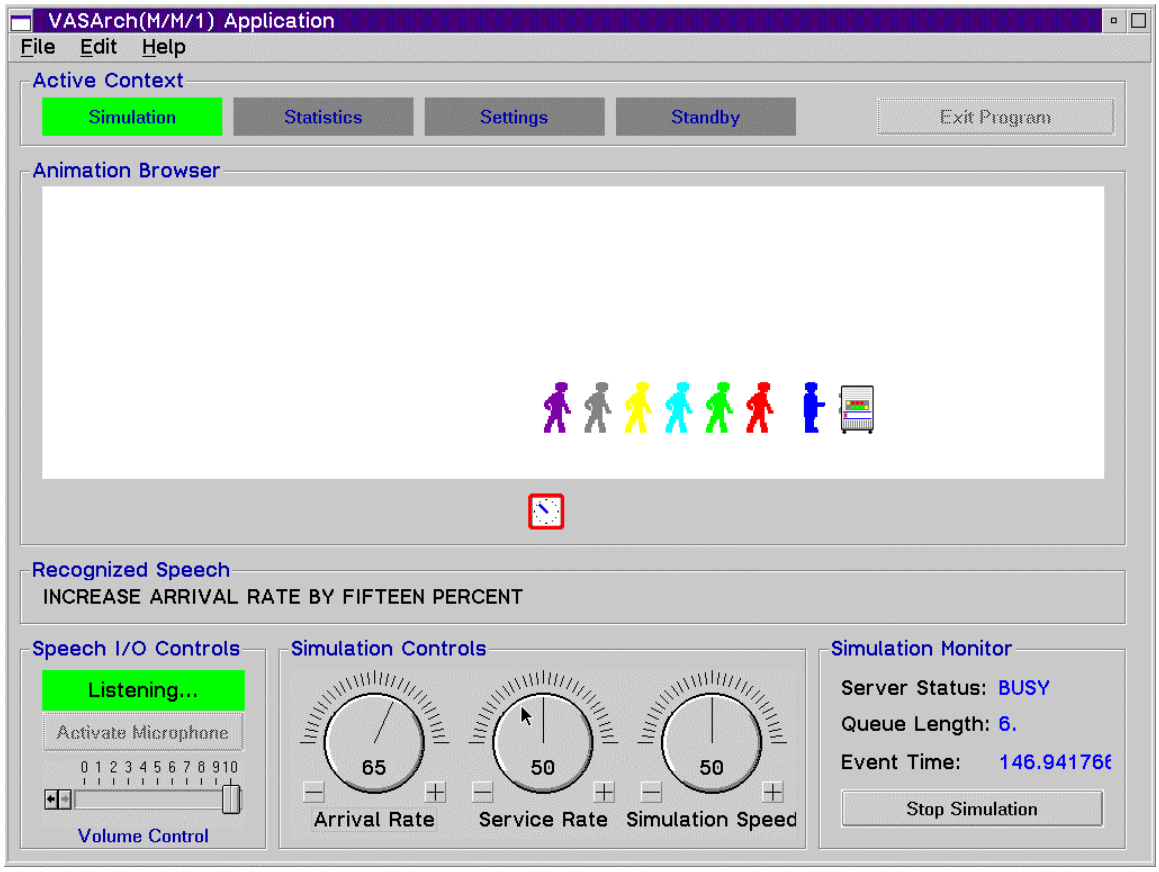

Figure 4: VASArch(M/M/1) Application Console

or busy. If there are no customers waiting to be served or in service, the server retreats to the idle position located to the right. The server will move into the busy position, located to the left, to perform service when a new customer arrives. While in an inoperative state, the server resides in the idle position located to the right.

3.4 User Interface Features and Functions

The user interface features and functions associated with the VASArch(M/M/1) prototype encompass the graphical user interface, animation presentation, speech recognition, and speech output. Throughout this section we describe the user interface features and functions from the perspective of the graphical user interface and animation presentation illustrated in Figure 4. The graphical user interface allows the user to manipulate window controls, such as push buttons, circular dials and horizontal sliders, through mouse and keyboard interaction. Speech recognition allows the user not only to manipulate these same window controls associated with the graphical user interface, but also to extend the unseen capability using spoken commands.

Explanation of the user interface features and functions is provided with respect to identifiable groups within the graphical user interface. A group in the graphical user interface is defined by a light gray borderline that surrounds control and presentation elements, and has a text label.

3.4.1 Active Context Group

The *active context* group, located near the top of the window frame, contains four indicators that visually identify the active context used for speech recognition that corresponds to the simulation, statistics, settings, or standby speech mode. Individual contexts are used by VASArch(M/M/1) in order to improve speech recognition efficiency and accuracy. Transition between speech recognition operating speech modes associated with a context occurs through use of keywords embedded in the structure of the spoken command. The operating speech mode can only be changed only through spoken commands, and not by mouse or keyboard manipulation.

The *exit program* push button contained within the active context group, located at the upper right corner of the window frame, terminates execution of the program when selected by mouse or keyboard. As an alternative, users may also terminate execution of the program by uttering any one of the spoken commands “exit program,” “end program,” or “quit application.”

3.4.2 Animation Browser Group

The *animation browser* group, located in the upper half of the window frame, displays the graphical animation representing the simulation of the single-server queuing system model. Individual customers and their movement through the system are distinguished by the assignment of different colors within the animation. The activity of the server resource is distinguished through animated movement and change of position. In addition, the animation browser group contains an animated simulation clock that rotates clockwise during execution. The animation browser group does not contain any controls that can be manipulated either by mouse, keyboard or spoken utterance.

3.4.3 Recognized Speech Group

The *recognized speech* group, located below the animation browser near the middle of the window frame, displays utterances recognized as a spoken command in the text field. If an utterance cannot be identified in the active context the application notifies the user of a problem through a message handled by speech synthesis and displays an error message in the text field.

3.4.4 Speech I/O Controls Group

The *speech I/O controls* group, located at the lower left corner of the window frame, consists of three components that include the speech recognition status indicator, the activate microphone push button control, and the volume level horizontal slider control. The speech recognition status indicator distinguishes whether the application is (1) "listening..." for the next spoken command; (2) "waiting..." to process an acquired spoken utterance; or (3) "standing by..." monitoring all spoken utterances for the attention getting word, also known as hibernation. Selection of the "activate microphone" push button control by mouse or keyboard interrupts the hibernation state and resumes the listening state with the active context in the simulation mode. The volume level horizontal slider control may be manipulated by mouse, keyboard, or spoken command to adjust the speech synthesis output.

3.4.5 Simulation Controls Group

The *simulation controls* group, located at the lower center of the window frame, contains three circular dial controls representing the adjustment mechanisms for the arrival rate at which customers enter the system, the service rate at which customers are serviced by the resource, and the simulation speed at which the animation-simulation is presented. Each circular dial control may be adjusted by mouse, keyboard or spoken command over a range of values from a minimum of 0 to a maximum of 100. The value on any circular dial control represents a relative magnitude for the corresponding rate. For example, the specification of a 0 value for the dial control corresponds to the minimum allowable rate, which is still greater than 0.

3.4.6 Simulation Monitor Group

The *simulation monitor* group, located at the lower right corner of the window frame, consists of three separate text fields which display real-time information related to the execution of the simulation system model for the server status, queue length, and event time. In addition, the simulation monitor group contains a push button control that may be activated by mouse, keyboard, or spoken command to start or stop execution of the simulation and associated animation.

3.5 Voice Assisted Operation

Figure 4 provides an illustration of a graphical user interface that should appear not too unfamiliar to experienced computer users. Based on our description of the user interface features and functions, most users should be able to readily navigate the application through mouse and keyboard interaction with limited instruction. Not shown in Figure 4 are the wide-range of transactions that can also be performed through speech interaction. In this section we emphasize the use of speech recognition and speech synthesis interaction to perform similar types of activities.

Speech recognition and speech synthesis interactions are organized into four structured speech modes to promote increased recognition accuracy and improved response time efficiency. The four speech modes, known as the simulation mode, statistics mode, settings mode, and hibernation mode, correspond to the indicators located in the active context group. The indicator for the active speech mode in the active context group is highlighted with a green background to provide the user a visual clue to guide word choice. The transition between the speech modes occurs transparently to the user when keywords embedded in a command statement are spoken. We further describe the details each of the speech modes in the sections that follow.

3.5.1 Simulation Speech Mode

The *simulation speech mode* provides the ability to adjust the arrival rate, service rate, simulation speed, volume control, and server resource state for the simulation model. For example, once the simulation-animation has been started, we adjust the arrival rate for customers entering the system by uttering the spoken command "increase arrival rate by fifteen percent." As a result, we observe in Figure 4 an increase in the arrival rate from fifty percent to sixty-five

percent as represented by the circular dial control contained with the simulation controls group. Following the increase, customers begin to accumulate in the queue waiting for service since the arrival rate exceeds the service rate.

3.5.2 Statistics Speech Mode

The *statistics speech mode* provides the ability for the user to display reports and graphs generated from statistical data captured with each simulation event. Furthermore, the statistics speech mode allows the user to ask a general question through speech recognition and receive a reply provided through speech synthesis. For example, a user may inquire, "what is the percentage utilization for the resource?" and the application may respond, "the resource utilization is eighty-five percent."

After transition into the statistics speech mode, the corresponding indicator within the active context group for statistics is highlighted with a green background. In Figure 5 we illustrate a request to produce a observation-based report by uttering the spoken command "display report on the customer time in system." As a result, a window dialog containing the report for the customer time in system is presented.

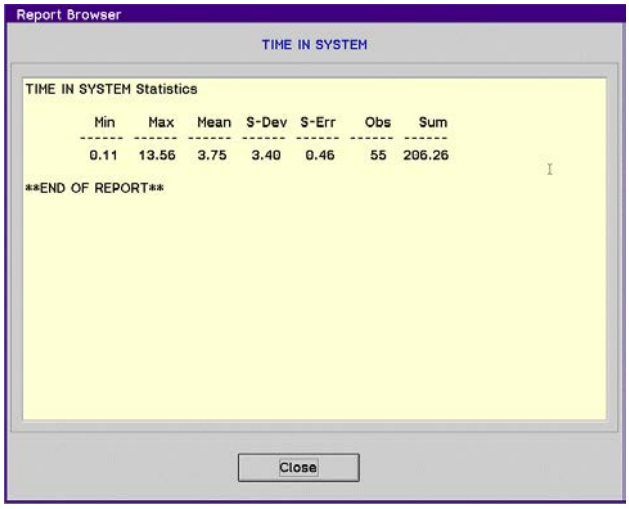

Figure 5: Spoken Command: "Display Report on the Customer Time in System"

3.5.3 Settings Speech Mode

The *settings speech mode* provides the ability for the user to adjust internal parameters associated with the system model through mouse, keyboard, or spoken command. Transition from another speech mode to the settings speech mode may be initiated through one of several spoken commands that include "investigate settings." After transition into the settings speech mode the corresponding indicator located in the active context group is highlighted with a green background. Adjustments are performed through a displayable control panel or directly manipulated by spoken command. The control panel is revealed when the user utters the spoken command "display control panel." Once displayed the user may adjust the maximum number of customers allowable in queue, or the threshold limit for a rare event that results in an action when trigged. Alternatively, the user may change these settings directly through spoken commands without opening the control panel.

3.5.4 Hibernation Speech Mode

The *hibernation speech mode* provides the ability for the user to reinitiate spoken interaction from a suspended state by continuously monitoring all spoken utterances for an attention getting word or phrase. Entry into the hibernation speech mode may occur because (1) the user initiated a request for suspension by spoken command, such as "suspend conversation"; (2) the user exceeded the maximum allowable time to provide spoken input; or (3) three or more consecutive errors were encountered related to speech interaction. The "standby" indicator located in the active context group is highlighted with a green background to alert users of the continuous monitoring for the attention getting word or phrase while in the hibernation speech mode. Utterance of the attention-getting phrase "wake-up-computer" will discontinue hibernation and result in the restoration of the last active speech mode to resume conversation.

4 RESULTS

The experience of speaking to your computer is quite awkward at first. Frequent mispronunciations, false starts, and ill constructed requests, make the initial experience less than desirable and sometimes downright frustrating. With a little practice, and confidence building, a reasonable performance level can be achieved. Unfortunately, as most have already discovered the recognition accuracy rate varies greatly by individual. We also concluded that the overall accuracy rate for speech recognition does not always match the claims. We experienced difficulties in the input of a long series of numbers, or complex numbers. As a result, we observed occasional word substitutions in the recognized phrases. Exceptionally good results were obtained when the sentence structures contained polysyllable words. Speech output should be used sparingly since it has a tendency to overwhelm new users.

5 SUMMARY AND CONCLUSION

The main findings of this research may be summarized as follows:

- The VASArch(M/M/1) prototype demonstrates a software architecture sufficient to support the

integration of speech recognition and speech synthesis technologies in conjunction with simulation-animation modeling software.

- The VASArch(M/M/1) prototype successfully managed the concurrent operation of all the functional system components in the software architecture using multithreaded and multitasking concepts.
- The VASArch(M/M/1) prototype obtained reasonable performance on a personal computer equipped with only a single Pentium-class microprocessor without the use of specialized hardware adapters, or coprocessors.

REFERENCES

Baber, Christopher and Janet M. Noyes (Eds.) (1993). *Interactive Speech Technology: Human Factors Issues in the Application of Speech Input/Output to Computers,* Bristol, UK: Taylor & Francis.

Bernsen, Niels Ole, Hans Dybkjaer, and Laila Dybkjaer. (1998). *Designing Interactive Speech Systems: From First Ideas to User Testing,* New York, NY: Springer-Verlag.

Burge, Thomas E., and Celi Joseph. (1993). *Advanced OS/2 Presentation Manager Programming*, New York, NY: John Wiley & Sons.

Dix, Alan, Janet Finlay, Gregory Abowd, and Russell Beale. (1998). *Human-Computer Interaction*: Second Edition. Hertfordshire, UK: Prentice Hall.

Dorfman, L., and M. J. Neuberger. (1994). *Effective Multithreading in OS/2*, New York, NY: McGraw-Hill.

Dutoit, Thierry. (1997). *An Introduction to Text-to-Speech Synthesis*, Boston, MA: Kluwer Academic Publishers.

IBM Corporation. (1993). *IBM Continuous Speech Series: Developer's Toolkit Technical Reference*, Second Edition, Armonk, NY: International Business Machines Corporation.

IBM Corporation. (1992). *IBM OS/2 Version 2.0, Presentation Manager Programming Guide*, OS/2 Technical Library, First Edition, Armonk, NY: International Business Machines Corporation.

Knight, Stephen A., and Jeffrey M. Ryan. (1995). *Programming the OS/2 WARP version 3 GPI*, New York:, NY John Wiley & Sons.

Kogan, and Dietel. (1994). *The Design of OS/2*, Second Edition, New York, NY: Addison-Wesley.

Stock, Marc. (1995). OS/2 Warp Control Program API, New York, NY: John Wiley & Sons.

Petzold, C. (1994). OS/2 Presentation Manager Programming, Emeryville, CA: Ziff-Davis Press.

Roe, David B., and Jay G. Wilpon (Eds.). (1994). *Voice Communications Between Humans and Machines*, Washington DC: National Academy Press.

Smith, R. L., (1999). A Voice Assisted Simulation-Animation Architecture. M.S. Thesis, Department of Industrial Engineering, North Carolina State University.

Taylor, M. M., F. Neel, and D. G. Bouwhuis. (1990). *The Structure of Multimodal Dialogue, Human Factors in Information Technology*, New York, NY: North-Holland.

Zue, V., R. Cole, and W. Ward. (1997). spoken language input," in *Human Language Technology*, Varile, G. and A. Zampolli (Eds.), New York, NY: Press Syndicate of the University of Cambridge, 1-70.

AUTHOR BIOGRAPHIES

RAYMOND L. SMITH III is a manager in e-business and Information Systems Development for the International Business Machines Corporation, Personal Systems Group, located at Research Triangle Park, NC. He received his B.S. degree in Industrial Engineering, and M.S. degree in Industrial Engineering and Operations Research, from North Carolina State University. He is a certified Project Management Professional (PMP). His interests include workflow management systems, decision analysis support systems, intelligent systems, and simulation applications. His email address is <rlsmith3@us.ibm.com>.

STEPHEN D. ROBERTS is a Professor in the Department of Industrial Engineering at North Carolina State University. Previously he was the Department Head and has served on the faculties of the University of Florida and Purdue University. He received his B.S.I.E.(with distinction), M.S.I.E., and Ph.D. from Purdue University. He was the recipient of the 1994 Distinguished Service Award. He has served as Proceedings Editor, Program Chair, and Board Member for WSC. His email address is <roberts@eos.ncsu.edu> and his WEB address is <www.ie.ncsu.edu/roberts>.