AIAA SciTech
5-9 January 2015, Kissimmee, Florida
AIAA Modeling and Simulation Technologies Conference

A Speech-Enabled Simulation Interface Agent for Airspace System Assessments

Hui-Ling Lu [*], Victor H. L. Cheng[†], Andrew Fong[‡], and Jimmy Nguyen[§]
Optimal Synthesis Inc., Los Altos, CA, 94022

Deborah S. Ballinger[**] and Steven E. Cowart[††]
NASA Ames Research Center, Moffett Field, CA 94035

and

Stephen Jones[‡‡]
SimPhonics Inc., Tampa, FL 33619

Extensive human-in-the-loop testing of the Next-Generation Air Transportation System (NextGen) concepts and technologies is typically required in a controlled lab environment before they can be integrated for evaluation in the field. The experiments tend to require the participation of a large number of subject-matter experts (SMEs) including air traffic controllers (ATC) and (pseudo-)pilots, where such requirements make the experiments costly and the logistics with so many participants make them difficult to plan. These experiments often are designed only to collect data from either ATC or the pilots, but seldom both; the counterpart is needed only to provide realism in communication between them. The paper illustrates our research in developing a Speech-Enabled Simulation Interface Agent (SESIA) to replace the non-essential human subjects in these experiments. SESIA can interact with the SMEs through voice communication, and interface with the simulation platform to perform the intended control. It has the benefit of reduced cost associated with the experiments and increased convenience in their planning, thus allowing the opportunities to schedule additional testing. In cases where a pseudo-pilot would normally represent multiple flights and communicate to the ATC with the same voice for all flights, SESIA will actually increase the realism of the experiments by allowing different voices to be synthesized to simulate different pilots.

I. Introduction

As concepts and technologies being developed for the Next-Generation Air Transportation System (NextGen) mature, the natural progression is to study their integration and evaluation in the operational environment. Before the concepts and technologies can be inserted into the National Airspace System (NAS) for evaluation in the field, they typically have to undergo extensive human-in-the-loop (HITL) testing in a controlled laboratory environment to identify and work out the issues. Depending on the particular concept/technology, the HITL experiments may involve subject-matter experts (SMEs) including air traffic controllers (ATC) and pilots. The laboratory environment would include realistic operational equipment such as appropriate ATC stations and flight-deck equipment. For assessments involving ATC as the primary test subjects, often pseudo-pilots would be controlling multiple flights working with some form of Pseudo-Aircraft System (PAS). One important system in this

[*] Senior Research Scientist, Optimal Synthesis Inc., Los Altos, CA 94022.
[†] Vice President, Optimal Synthesis Inc., Los Altos, CA 94022, AIAA Associate Fellow.
[‡] Research Engineer, Optimal Synthesis Inc., Los Altos, CA 94022.
[§] Research Engineer, Optimal Synthesis Inc., Los Altos, CA 94022.
[**] Project Manager, NASA Ames Research Center, Moffett Field, CA 94035.
[††] Manager, NASA Ames Research Center, Moffett Field, CA 94035.
[‡‡] President, SimPhonics Inc., Tampa, FL 33619.

laboratory environment is a realistic communication system for simulating radio communications among the controllers and pilots. In current-day operations, controllers and pilots communicate by voice over VHF radio. In the laboratory environment, this communications capability is typically provided by a dedicated communication system that allows connection of radio headsets with push-to-talk (PTT) capability to support communications among the operators.

To ease the cost and burden in hiring secondary SMEs to support the experiments, we set out to develop an automated speech-enabled simulation interface agent (SESIA) that can take the place of the secondary SMEs in communicating with the primary SMEs and interacting with the operational environment. In this context, the primary SMEs are the essential test subjects whereas the secondary SMEs are those required to support the experiment. As an example, in an experiment designed to test ATC workload and situation awareness associated with the operation of an ATC automation system, the ATC represents the primary SMEs, and pseudo-pilots would be the secondary SMEs. On the other hand, in a flight-deck experiment designed to test pilot workload and situation awareness associated with an advanced flight-deck concept, the pilots would be the primary SMEs, and the ATC may be secondary SMEs. The key point is that the experiment is designed to collect data from the primary SMEs but not from the secondary SMEs.

The automated speech agent technology is expected to provide the following benefits:

- Secondary SMEs are no longer necessary to support an experiment.
- This eliminates the cost associated with hiring the SMEs.
- The convenience and reduced cost resulting from elimination of the SMEs allow opportunities for scheduling additional testing involving only the primary SMEs.
- In a PAS environment, a pseudo-pilot typically would represent multiple flights and would communicate to the ATC with the same voice for all the flights, whereas an automated speech agent would allow different voices to simulate the pilots on different flights. This increases the realism of the experiments.

To demonstrate the capabilities of SESIA, specifically, we have developed a full-function prototype of a speech-enabled autonomous agent to interface with the communication system to recognize speech clearances and generate realistic speech responses, and to create actionable commands to be sent to a simulation platform to carry out the simulated pilot control of the aircraft.

Section II provides an overview of the functional requirements for building the SESIA. Section III provides a high-level description of the SESIA system including its functional components. This is followed in Section IV by more detailed descriptions of the different functional components in the implementation. Section V describes the experiment setup for feasibility demonstration of the SESIA. Concluding remarks are provided in Section VI.

II. Overview of Functional Requirements of the SESIA Prototype

In view of the nature of the research projects prominently centered around the air traffic management (ATM) theme at NASA Ames Research Center, the SESIA system focuses on concepts and experiments involving air traffic controllers (ATC) as test subjects, and pilots and pseudo-pilots will play supporting roles in the simulations. Consequently, the SESIA system is developed to serve the pseudo-pilot functions in a laboratory environment and is expected to interface to flight-deck simulations, which will receive aircraft control commands converted from ATC clearances by SESIA. Its design has the following functions to support simulation of human communication by voice and controlling the aircraft simulation platform:

- Communication system interface with voice communication system.
- Speech recognition capability to allow the agent to convert voice messages (e.g., clearances issued by ATC) to text.
- Generation of actionable commands based on the recognized text to drive the simulation.
- Interface to simulation platform to communicate the actionable commands to the simulation platform.
- Generation of "voice" responses in text form in response to received messages.
- Text-to-speech capability for converting the response text to speech for communication back through the voice communication system.

III. System-Level Design of SESIA

Figure 1 depicts the high-level structure of SESIA's operational environment showing how SESIA may be connected to the voice communication system and the simulation platform. Functionally, the SESIA concept will involve the necessary interfaces as well as the core processing for recognizing the speech from the voice communication system, converting the recognized speech into actionable commands for transmission to the

simulation platform, and generating the speech response for communication back through the voice communication system. Figure 2 illustrates the functional block diagram that contains the envisioned high-level functional subsystems of SESIA. More detailed description of the speech agent is provided in Section IV.

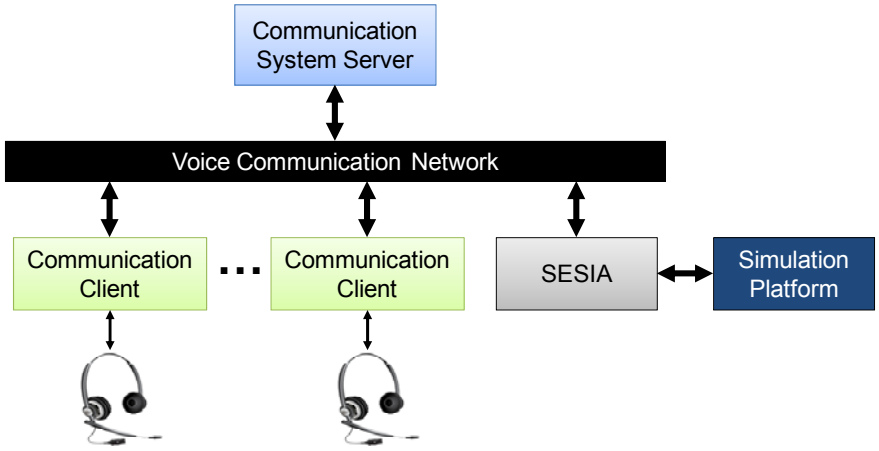

Figure 1. High-Level Structure of SESIA Operational Environment

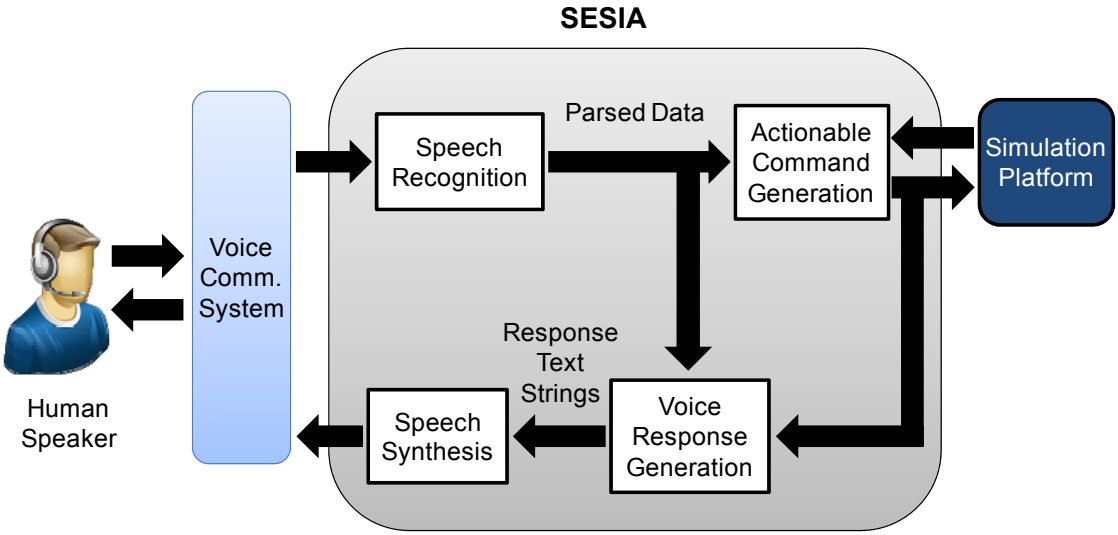

Figure 2. Functional Block Diagram Showing SESIA Functions

IV. Implementation of SESIA Functions

The objective of the automated speech agent is to provide a seamless speech interface to perform human-like functions in human-in-the-loop simulations. As shown in Figure 2, several functions are required to construct the SESIA. Implementations of each function are detailed below.

A. Speech Recognition System Development

The speech recognition and language understanding component is constructed based on our recently developed speech recognition system for Navy operations in carrier air-traffic control. This Navy project addressed the feasibility research on utilizing the speech recognition technology for carrier air traffic control. The speech recognition capability enables seamless integration of the Navy Unmanned Combat Air Systems (N-UCAS) with aircraft carrier operations.

The resulting recognition system was shown to achieve 98.7% average sentence recognition based on the database recorded aboard carrier CVN-76 under extremely noisy flight operations. The speech recognition technology developed is a state-of-the-art Hidden-Markov-Model-based recognition system based on the flexible open-source Sphinx-4 speech recognition decoder[1]. The Sphinx-4 system was developed in the Java programming

language, and thus can be deployed to a large variety of development platforms. Figure 3 depicts the block diagram of the developed speech recognition system. An innovative front-end processing approach was developed to reduce the impact of noisy background. Enhanced recognition accuracy was obtained via adapting acoustic models, a recognizer lexicon and a language model. The recognition system also includes two notable features that enhance the recognition accuracy significantly: a rapid speaker adaptation technique which adapts the acoustic model using only one short enrollment sentence and a dynamic language model that incorporates aircraft ID information from the flight plan. The use of the aircraft ID information addresses the problematic aircraft ID recognition issue reported by several prior studies[2].

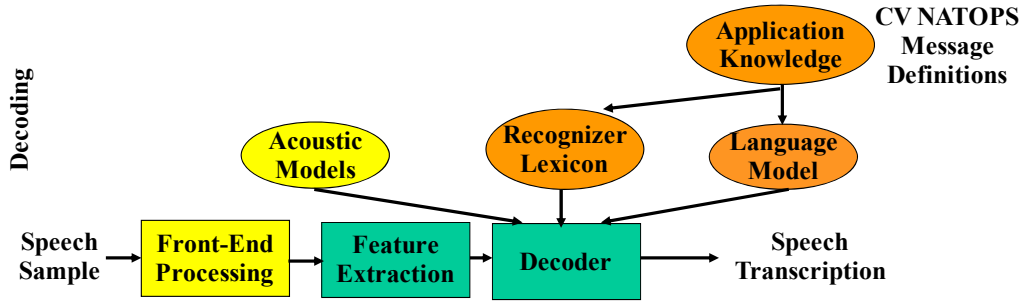

Figure 3. Simplified Block Diagram of the Speech Recognition System

For a command-and-control task, a graph-driven grammar can greatly reduce the recognition errors by applying language restrictions effectively. Figure 4 shows an example of the graph-driven grammar implementation, in which "On" and "Off" indicate radio on and radio off for press-to-talk radio communication scenarios. The Java Speech API Grammar Format (JSGF) is used to specify this graph-driven grammar. The JSGF format specifies a semantic Context-Free Grammar (CFG). With the tagging functionality as shown in Figure 5, the use of the JSGF grammar allows us to extract semantic information from the recognition results directly. Hence, tight coupling of speech recognition and language understanding is easily achieved through CFG implemented via the JSGF.

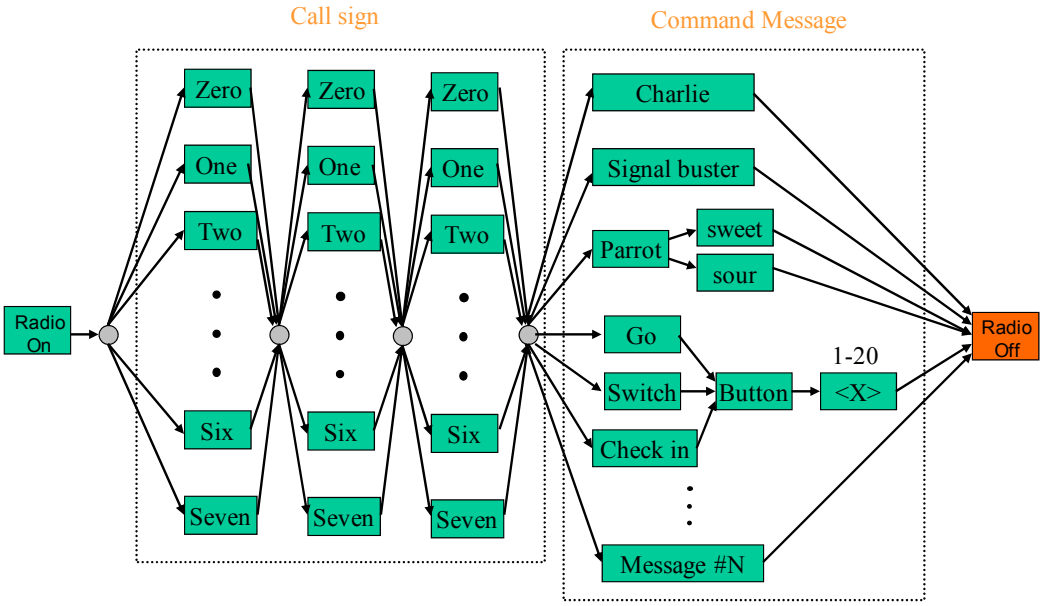

Figure 4. Example of the Graph-Driven Grammar

```
#JSGF V1.0;

/**
 * JSGF Grammar for Departure Controller
 */

grammar Departure;

public <commands> = <singleCommand>;

<singleCommand> = <octal> <octal> <octal> <action>;

<action> = <acknowledge> | <generalAction> | <departureAction>;// | <situationalAction>;

//UL1 - UL7
<acknowledge> = roger {UL3};

//UL11 - UL60
<generalAction> = (radar contact [lost] [<one_to_fifty> miles]) {UL11}
        | ([turn] (left | right) [heading] <radialNum>) {UL12}
        | (fly (heading <radialNum> {UL13} | present heading {UL14}))
```

tag

Figure 5. Example of JSGF (Java Speech API Grammar Format)

B. Actionable Command Generation

For converting recognized speech text to actionable commands in the simulation platform, the semantic information of the spoken command has to be extracted. In fact, semantic information such as call sign, requested action, and the arguments associated with the action are readily extracted since this information is embedded in the language model specified by JSGF by using the command tagging approach. Command tagging in the grammar file allows one to map the spoken command to each actionable command. Figure 5 shows an excerpt of a JSGF grammar for the departure controller in the Carrier ATC environment with example of command tagging. With the command tagging, the speech recognition output could be parsed according to the tagging number. The actionable command type and its corresponding arguments could then be determined.

C. Voice Response Generation

The response generation component creates text string input to the text-to-speech (TTS) engine. In the most simplified case, the response generator would simply "read back" the control command message extracted by the speech recognizer. However, for realistic simulations, naturalness of the responses is critical. The objective of this component is to mimic the pilot response by adding common variations to the control command. The rules of common variations in ATC application include dropping and shorten call sign ID, inserting greetings, and swapping orders of command messages. Currently, SESIA has a simplified implementation that creates randomness of the common variations for the response generation component. In the event that the controller mistakenly gives commands to a wrong flight not in the flight schedule, the response is silent, which is what one would expect when the flight does not exist. To make the read-back response realistic, the design of a rule-based read-back response module based on the human pseudo-pilot responses in previous recorded ATC simulation experiments is underway.

D. Speech Synthesis FreeTTS

The text-to-speech speech synthesis component converts the text string generated from the response component to voice output. Most state-of-the-art speech synthesizers are based on the concatenative speech synthesis paradigm, in which synthetic speech is created by concatenating pieces of pre-recorded speech that are stored in a database. Systems differ in the size of the stored speech units; a system designed for synthesizing speech in unlimited domain stores phones or di-phones units. For limited domains, the storage of entire words or sentences allows for high-quality output. The implementation of SESIA's speech synthesis component is based on the FreeTTS open-source software[3] that supports both unlimited-domain and limited-domain applications[4].

FreeTTS is a speech synthesis system written entirely in the Java programming language. Its features include an extensive API documentation, partial support for Java speech API, and full support for importing synthetic voices created from FestVox[5], a well-known project by Carnegie Mellon University which aims to support building new synthetic voices. FreeTTS system is very user-friendly. It does not require any system parameter configurations or

module configurations. Furthermore, FreeTTS can be easily integrated with other software due to its well-designed API. Figure 6 shows the overall architecture of the FreeTTS[3].

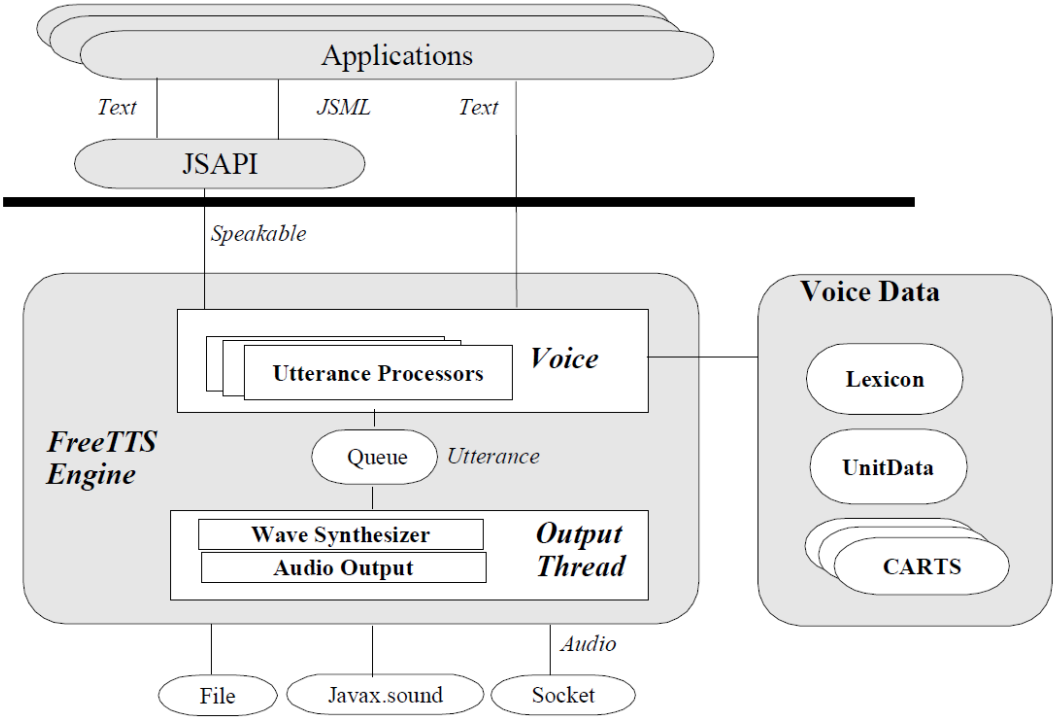

Figure 6. FreeTTS Architecture[3]

Apparently, the performance of FreeTTS depends on the prebuilt synthetic voices. Festival[6] and FestVox are the main software responsible for creating these voices. FestVox provides a high-level systematic way to streamline the new voice creation process using the Festival engine. Using a set of recorded prompts, we created Festvox text-to-speech models for different voices. The Festvox models were then converted to FreeTTS models using the FreeTTS conversion tool. Currently, eight voices were created to simulate multiple virtual pseudo pilots. All voices sound natural for most commands covered under the current extracted ATC grammar.

E. SESIA Interface with Voice Communication System

The laboratory environment selected for testing SESIA uses a SimPhonics system as the platform for simulating radio communications. The integration between SESIA and the SimPhonics voice communication system is via the Distributed Interactive Simulation protocol (DIS, IEEE-1278.x)[7] established as a standard for binary exchange of information in military simulations. The interoperability of the standard relies primarily on a consistent format for information on the wire. Part of the DIS protocol deals with radio simulation, so that simulated entities can communicate with simulated radio and intercoms.

DIS sends and receives many different types of Protocol Data Units (PDUs) over socket connections in a unicast, multicast, or broadcast mode. Two types of PDUs are used under the DIS radio: the transmitter PDU and the signal PDU. The transmitter PDU contains information about the simulated transmitter and the signal PDU contains the actual voice data for transmission.

Figure 7 shows a typical DIS radio transmission. Transmitter PDUs are sent out at regular intervals while the signal PDUs are only sent when audio data is being sent, e.g., when the PTT is active. A transmitter PDU is also generated at the beginning and at the end of the signal PDU sequence (on inactive-to-active PTT transitions and active-to-inactive PTT transitions). Since SimPhonics voice communication supports the DIS protocol, SESIA's voice communication component is configured as a DIS radio so that it can receive voice signal for speech recognition and transmit synthesized voice after text-to-speech conversion. In addition, with the transmitter PDU, the PTT signal can also be obtained from the DIS radio.

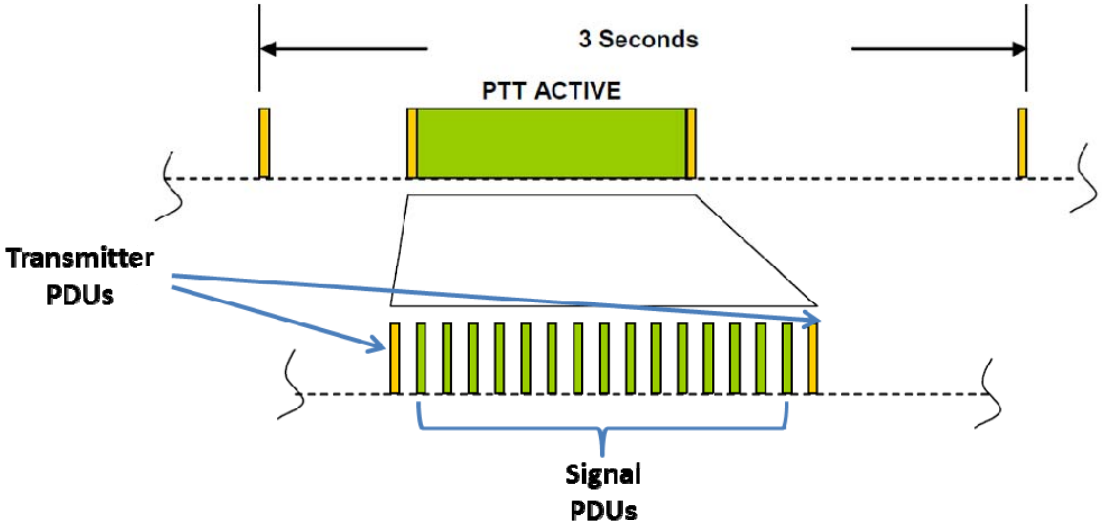

Figure 7. DIS Radio PDU[8]

The construction of the DIS radio interface is based on the open-DIS open-source implementation. Open-DIS[9, 10] is an effort to make available to the modeling and simulation community a full implementation of the IEEE DIS protocol in both C++ and Java. Since our agent implementation is in Java, it makes the integration more straightforward. Previously, the radio communication PDU was not fully implemented. Through collaboration with the MOVES Institute at the Naval Postgraduate School to complete the radio communication PDU implementation, the DIS radio capability is enabled, which allows SESIA to link with the SimPhonics voice communication system.

F. SESIA Interface with Simulation Platform

The laboratory environment selected for testing SESIA uses the Multi Aircraft Control System (MACS) developed by NASA as the platform for simulating air traffic in the NAS. Software interface between SESIA and MACS is necessary to integrate the recognized actionable control commands from "intelligent" rule-based response subsystem with the aircraft simulations.

For this effort, we have brought up the MACS software for demonstrating the communication between the controller station and the pilot station. This setup supports verification of the communication between the SESIA speech agent and MACS.

The message types for controlling MACS operations typically include speed/heading/altitude changes, radio channel selection, and waypoint manipulations. Figure 8 shows a screenshot of the MACS GUI for pseudo-pilots. Most clearances are linked to three panels: radio handoff, mode control, and the FMS route. Here we overlaid several related clearances with these panels. The radio handoff allows the pseudo-pilot to change the radio communication channel. The mode control panel (MCP) illustrates three major message types that command the pilot to change speed, heading, and altitude. The FMS route panel enables the clearances related to waypoint manipulations. Utilizing the extracted actionable commands and their associated arguments from the rule-based response subsystem, the extracted actionable commands are mapped to the command function implementations in MACS. To activate the MACS actions, we have developed a proprietary communication interface between MACS and SESIA via a Socket Communication approach with the goal to provide real-time communications between MACS and SESIA.

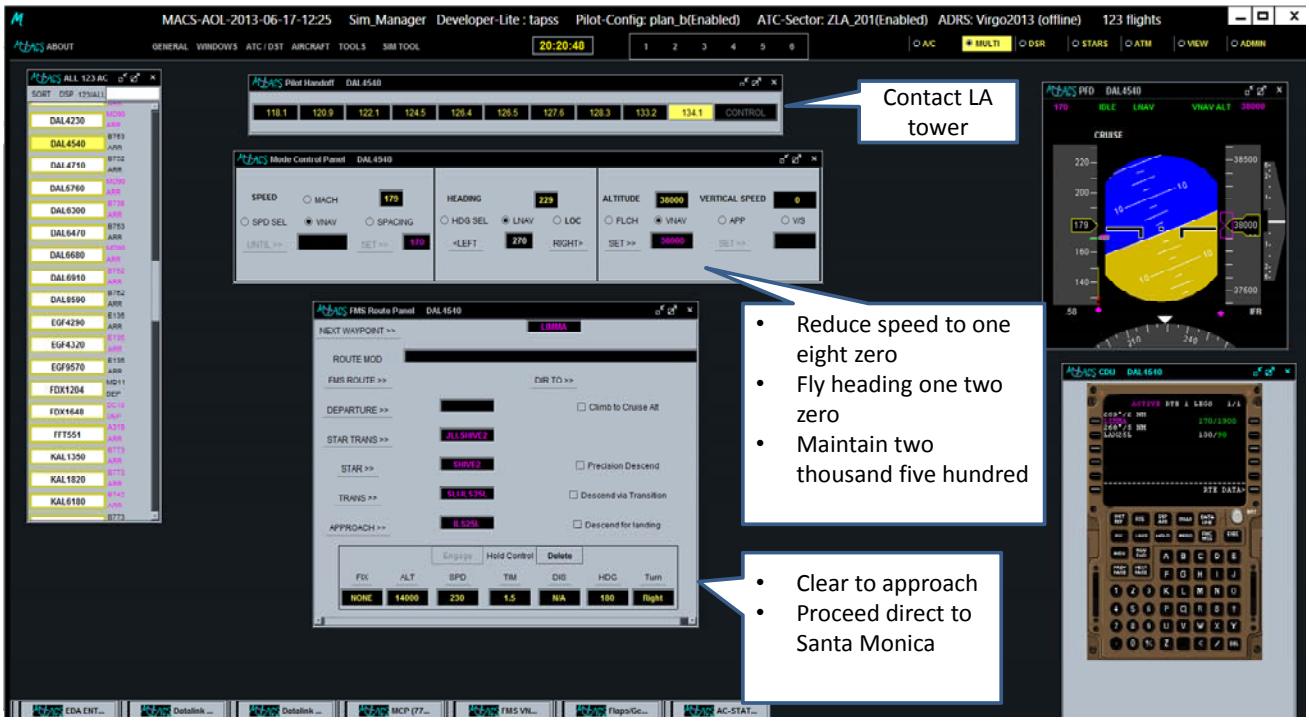

Figure 8. Linking between MACS Commands and Clearances

Figure 9 shows the functional diagram of the two-way socket communication operations. MACS socket thread is implemented under SESIA and SESIA socket thread is implemented under MACS for exchanging information between SESIA and MACS. MACS socket thread also maintains and stores data for flights under control. Two-way communications is needed since MACS needs to retrieve actionable commands from SESIA. On the other hand, MACS might need to send data to SESIA when the controller requests information or if the pseudo-pilot needs to check in with the controller. Figure 10 depicts one example of two-way communications between SESIA and MACS. The recognized clearance is converted to actionable commands via the GetMACSInput function and then is sent to MACS. Concurrently, MACS could transfer aircraft data to SESIA.

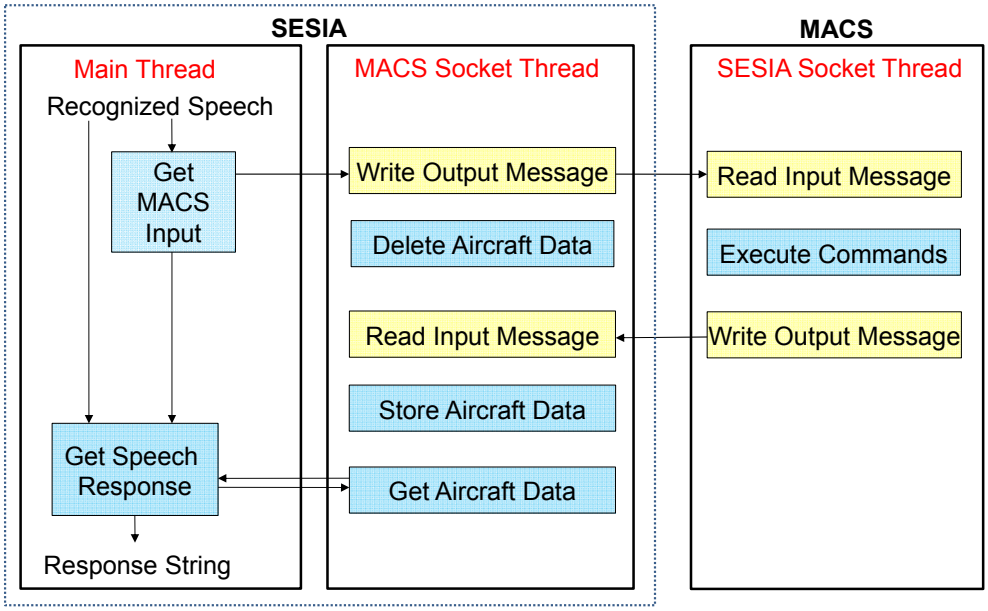

Figure 9. Two-Way Socket Communications between SESIA and MACS

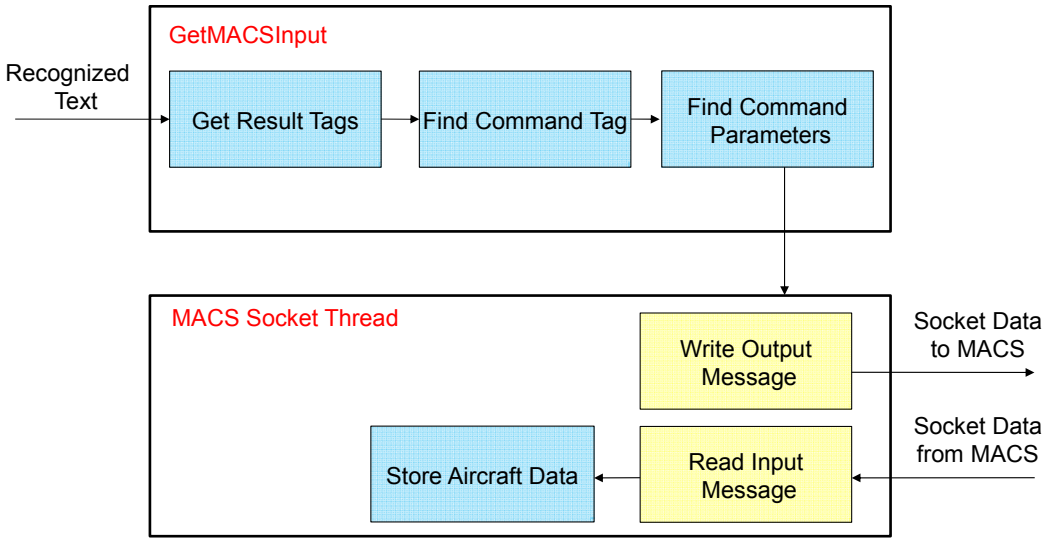

Figure 10. Two-Way Socket Communication Example

V. Feasibility Demonstration of the Automated Speech Agent

Figure 11 shows the current lab setup for testing the SESIA integration with both the MACS simulation environment and the SimPhonics Voice Communication system. One SimPhonics voice communication box sits at the right, and the leftmost desktop was configured to run MACS. The SESIA machine sits in the middle.

The SimPhonics computer is connected with a Jack Box. When the computer is booted, the V+ Run Time System automatically starts in multicast mode. To accept the audio from the headset connected through the Jack Box, the SimPhonics computer settings must be modified to receive audio input from the Jack Box. Using this setting, the audio from the SimPhonics Box was sent to the SESIA agent that recognized the audio and sent the synthesized speech response back to the SimPhonics Box successfully.

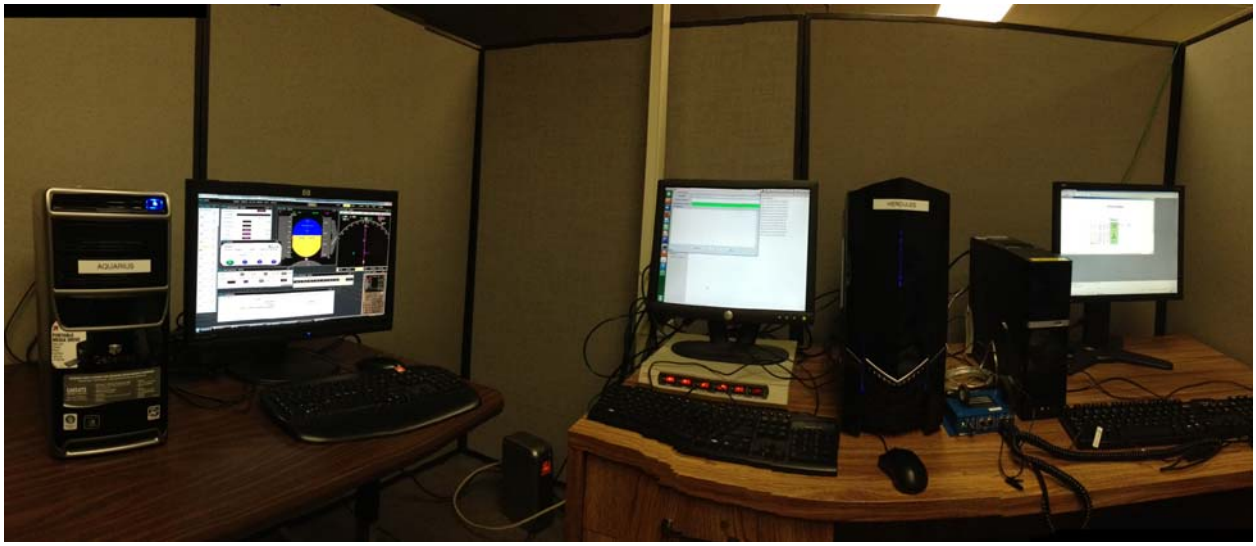

Figure 11. Lab Setup for the Integration Feasibility Study

To address the feasibility of connecting multiple SESIAs with multiple MACS stations, multiple SESIAs were tested to communicate with multiple MACS stations from a single computer. The SESIA socket thread on MACS is designed to operate as a server, while the MACS socket thread on SESIA is designed to act as client. During the connection process, a MACS station listens to port 9000 on the server machine for connection requests. An instance of SESIA connects to MACS by attempting a connection with the user-specified IP address of MACS and the known port number of the server. The port number of the client machine is selected by the operating system, so that

multiple socket connections can occur on the client machine. As a result, multiple instances of SESIA on a single computer can connect to different MACS stations on different computers without networking conflicts.

To perform this test, two instances of SESIA were started from a single computer. Each instance is run with different XML configuration files for different radio frequencies. The radio IDs and the IP addresses of the MACS stations are defined as command-line arguments for each instance. Shell script files were written for each instance to simplify the execution of the instances with the different arguments. First, two MACS stations connected to the MACS network server were started to initiate the simulation. Next, the VComm Radio was used for audio input to MACS stations. Audio commands were given over two radio frequencies simultaneously, and only the MACS station controlling specified aircraft would respond.

VI. Concluding Remarks

This paper documents our efforts in constructing a speech-enabled agent that aims to reduce the cost and enhance the functionality of human-in-the-loop (HITL) experiments of advanced airspace operational concepts in a laboratory environment. In particular, we address the issue of building an automated speech agent that can take the place of the secondary subject-matter experts (SMEs) in communicating with the primary SMEs and interacting with the simulation platform in HITL experiments. A full-function prototype of an automatic speech agent, SESIA, has been constructed under this research. SESIA is configured to mimic pseudo-pilots' operations and is capable of supporting experiments designed to test ATC workload and situation awareness associated with the operation of an ATC automation system.

SESIA will simplify the logistics in planning and performing HITL experiments by eliminating the need to recruit, train and organize test subjects who would otherwise be needed to support the experiments, even for preliminary experiments required to check out the systems to be evaluated. It will reduce the cost associated with HITL experiments and increase the opportunities in performing such experiments. It can replace pilot subjects in ATC-centric experiments, as well as ATC subjects in pilot-centric experiments. To deploy the SESIA in HITL experiments, performance assessment of the SESIA prototype in selected NASA facility is underway.

Acknowledgments

This development effort was performed under support from NASA Contract Number NNX13CA01C. The authors would like to thank Mr. Don McGregor and Prof. Don Brutzman in the MOVES Institute at the Naval Postgraduate School for their valuable supports on the open-DIS open-source software package. Thanks are also due to numerous facility managers and developers at NASA Ames Research Center who provided valuable information on NASA's facility and experiment needs to OSI researchers, including Dan Wilkins, Matt Gregory, Jacob Pfeiffer, Al Globus, Dr. Ronald Lehmer, Diane Carpenter, David Chin, Marty Pethtel, Fay Chinn, and Estela Buchmann.

References

[1]W. Walker, P. Lamere, P. Kwok, B. Raj, R. Singh, E. Gouvea, P. Wolf and J. Woelfel, "Sphinx-4: A Flexible Open Source Framework for Speech Recognition," *SMLI TR-2004-139*, Nov. 2004.

[2]J. M. Cordero, M. Dorado, and J. M. de Pablo, "Automated speech recognition in ATC environment," *Proceedings of the 2nd International Conference on Application and Theory of Automation in Command and Control Systems*, IRIT Press, 2012.

[3]Walker, W., P. Lamere, and P. Kwok, *FreeTTS - A Performance Case Study*, Technical Report TR-2002-114, Sun Microsystems, Menlo Park, CA, http://research.sun.com/techrep/2002/abstract-114.html, 2002.

[4]Black, A., and Taylor, P., "Automatically clustering similar units for unit selection in speech synthesis," *Eurospeech97*, vol. 2, pp. 601-604, 1997.

[5]Black, A. W., Lenzo, K., "Building Voices in the Festival Speech Synthesis System," http://www.festvox.org/bsv.

[6]Taylor, P., Black, A., and Caley, R., "The Architecture of the Festival Speech Synthesis System," *Proc. 3rd ESCA Workshop on Speech Synthesis*, pp. 147–151, Jenolan Caves, Australia, 1998.

[7]IEEE Standard 1278.1-1995 (and revisions), Standards Committee on Interactive Simulation (SCIS) of the IEEE Computer Society, Approved September 21, 1995.

[8]Stephen Jones, *Radio PDU Generator USER'S MANUAL*, June 2012.

[9]Open-DIS: An Open Source Implementation of the Distributed Interactive Simulation Protocol, http://open-dis.sourceforge.net/Open-DIS.html. (Last Access, Jan. 14th)

[10]D. McGregor and D. Brutzman. "Open-DIS: An open source implementation of the DIS protocol for C++ and Java," Simulator Interoperability Working Group (SISO) Fall Workshop, 2008.