INTERNATIONAL JOURNAL OF INTELLIGENT CONTROL AND SYSTEMS
VOL. 14, NO. 1, MARCH 2009, 115-122

Virtual Operating Room Team Training via Computer–Based Agents

Emre Baydogan, Lee A. Belfore, II, Mark Scerbo* and Saurav Mazumdar
Department of Electrical and Computer Engineering , Department of Psychology*
Old Dominion University
Norfolk, VA 23529
{ebayd001, lbelfore, mscerbo, smazu001}@odu.edu

***Abstract*—The Virtual Operating Room (VOR) is an operating room training environment capable of supporting team training for surgical procedures. One of the challenges of creating such a team training environment is assembling a suitable team of participants for a training session. In the VOR, simulated agents may be substituted for actual participants. This provides the ability to conduct training sessions with a few or even one participant. In this paper, the VOR computer-based agent architecture is presented. The agent automaton in VOR interacts with others agents present in the environment through specific inputs and outputs such as voice recognition, environment sensors, and equipment simulators. In the VOR simulation architecture, the primary components in the system such as voice recognition and rendering are decoupled and therefore allow for modular development and abstraction. The potential interactions between the simulated agents and the trainee(s) are based on an underlying scenario script. The script provides the potential sequence of steps, interchanges, branching (deviations) that may occur in the training.**

***Index Terms*—Virtual Operating Room, Team-Training, Agent-Based Simulation, Laparoscopic Cholecystectomy, Medical Simulation**

1. INTRODUCTION

THE use of simulation in training applications is viewed as a vital step in the training process without putting the safety of people or property at risk. Furthermore, simulation provides the opportunity to create scenarios to match the desired training goals, whereas "on the job" training approaches rely on suitiable cases. Indeed, the length of medical residencies has been limited to 80 hours per week by the Accreditation Council for Graduate Medical Education [1]. This reduction has advantages and disadvantages. Among the disadvantages, medical residents will see fewer cases, therefore potentially compromising their training. Simulation based training provides the opportunity to restore some of these missed training opportunities.

1.1. Medical Simulation

In the medical domain, the core technology is the use of interactive three–dimensional (3D) visualization, or virtual environment (VE), for training personnel, and also for improving the quality of patient care in emergency situations, hospitals, and battlefields. The experience of VEs can be immersive or augmented with a head–mounted display (HMD), on a 3D video monitor, or in a room size CAVE; stand alone, distributed, or Internet-based. Thus, it is possible to customize the required VE experience for the respective health care provider or patient. Most importantly, the benefit of medical simulation is that training exercises and scenarios can be created and performed without putting patients at risk. Furthermore, medical simulations can provide performance measurements that, if properly designed, may be used on assessing the level of proficiency for the trained scenario. VEs are used in diagnosis, therapy, education, and training [2].

1.2. VOR

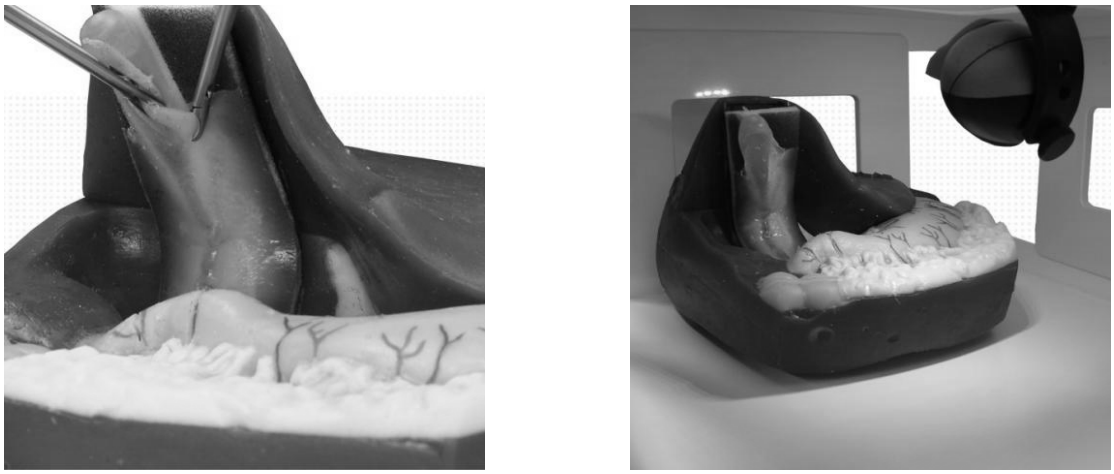
Most current medical simulators target specific procedural skills (e.g., airway management, laparoscopy). Although training on these simulators is an ideal way to practice procedures, they are not designed to address errors resulting from inappropriate equipment design, poor judgment, or team interactions in the OR.

The Virtual Operating Room (VOR) is a fully immersive virtual environment designed to augment procedural simulator training with a simulated OR. The VOR is modeled on a standard OR and outfitted with both real and virtual instruments as well as commercial medical simulators. The VOR does not replicate existing medical simulation technology, but instead provides the OR context in which to study performance.

The initial surgical procedure selected for development was a Laparoscopic Cholecystectomy (gall bladder removal). This procedure was chosen because it is fundamental to any surgical training program and there are several commercial simulators (E.g. Simulab Laparoscopic Colysystectomy with LapTrainer, see Figure 1) that support it. The knowledge base for each virtual agent was obtained from genuine surgeons, nurses, and anesthetists using cognitive and hierarchical task analyses. The surgical procedure was analyzed and divided into a sequence of tasks for each member of the surgical team including the attending surgeon, anesthetist, circulating nurse, and scrub technician. The resulting workflows were then used to create a script for the VOR providing a framework for coordinating movements and dialogue for each agent and a library of statements trainees were expected to utter.

Manuscript received April 1, 2009. This work was supported by the Commonwealth of Virginia. This paper is extended from Baydogan et al. "Simulation Architecture for Virtual Operating Room Training" published at *SCS Agent-Directed Simulation Symposium*, San Diego, CA, USA, March 2009.

E. Baydogan is currently with the National Capital Area Medical Simulation Center at Uniformed Services University of the Health Sciences.



(a) Gall Bladder Model (b) Gall Bladder Model with LapTrainer

Fig. 1. Simulab Laparoscopic Colycystectomy Model (Copyright© Simulab Corporation 2009 – permission pending)

The dialogue structure was developed from the task analyses. Different personalities for the virtual agents were generated using the Big Five model [3] as a foundation. Statements that addressed the procedural steps were consistent across personality types. However, words, sentence structures, and the manner in which statements were communicated were changed to fit a particular personality type. For example, a helpful agent might say, "Could you please turn off the lights?", whereas an arrogant agent might say, "Why are the lights still on?"

The Virtual Operating Room (VOR) is an immersive training environment wherein medical teams can be trained to perform surgical procedures [4], [5]. The principal goal of the VOR is to support team training in an operating room. The VOR allows for interaction between actual participants, virtual agents in the absence of actual participants, or a team comprising of actual participants interacting with virtual agents (Figure 2).

Fig. 2. A snapshot of the VOR: trainee, scrub tech, attending surgeon (agent), nurse (agent), patient (mannequin)

The VOR provides a simulated operating room environment. The objective of the VOR is not to build or simulate from scratch all components that may be a part of the training exercise in an operating room but to reuse available medical simulators, tools and devices and incorporate them into the simulated environment to provide team training. Some of the components that may be found in the VOR are surgical simulators, anesthetist workbench and a mannequin. Furthermore, VOR provides capabilities to simulate components such as the patient, other members of the surgical team and their interactions.

In this paper, an architecture to support the VOR is described. The simulation is agent–based. An agent substitutes for any absent team member or any other member who is required for the training. The agent automaton in VOR interacts with others agents present in the environment through specific inputs and outputs such as voice recognition, environment sensors, and equipment simulators. The agent is provided with a behavior that simulates the desired personality of the team member. The potential interactions between the simulated agents and the trainee(s) are based on an underlying scenario script. The script provides the potential sequence of steps, interchanges, branching (deviations) that may occur in the training. The initialization script in the VOR is created using XML(Extensible Markup Language).

2. Agent–Based VOR Simulation

In Figure 3, a general view of the agents in the simulation is presented. As previously stated, an agent may represent any team member or participant required in the team training scenario. In the VOR simulation, each agent is an independent, autonomous state machine. The agent has a 3D visual representation that the trainee can interact with. Each agent has its own set of animations that may be triggered based on pre-specified input. The trainee may interact with the agents using speech. Each agent may be given a behavior to simulate the personality of the patient or team member. Agent behaviors can also be modified to reflect different personalities. The personality differences are implemented by substituting different

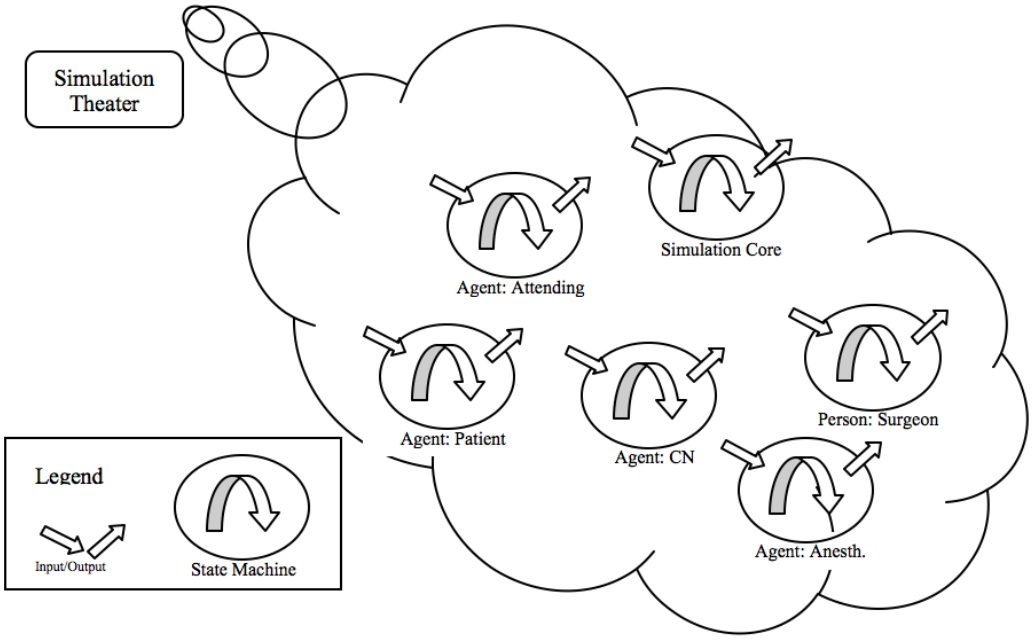


Fig. 3. Agent-based view of the VOR. Each entity (E.g. Person, and Patient) has a state machine of its own. (CN: Circulating Nurse, Anesth: Anesthesiologist)

animations and modifications to speech generation to reflect the change. In speech generation, changing pitch, volume, and rate allows us to add personality and mood.

The Simulation Theater is the central simulation controller that coordinates the interactions between the agents and between the agents and the environment. It handles generated events, time management and simulation flow.

A generic component view of the VOR architecture is provided in Figure 4. The speech recognition component recognizes speech and is the primary mode of input to the simulation. Based on this primary input, the agents may interact with the trainee. The agents recognize specific keywords uttered by the trainee and then communicate using speech, where the speech is predefined for the respective keywords and is stored in a textual format. The speech is produced using these texts via text to speech component. The agents in turn can produce events which may change its own state, state of other agents or the central state machine.

3. VOR Simulation Architecture: A Package View and Implementation

In Figure 4, the package view of the simulation architecture for the VOR is shown. Figure 5 shows modules of the simulation theater that constitutes all connectible elements in VOR.

Note that the elements such as LapTrainer [6], which does not have connectivity to other elements, are not shown. The modules are discussed subsequently.

3.1. Core and Modules

In this subsection, the core of the simulation as well as the major modules are described. The core is the Simulation Theater and the modules are the rendering, speech recognition, text to speech, and simulation console modules.

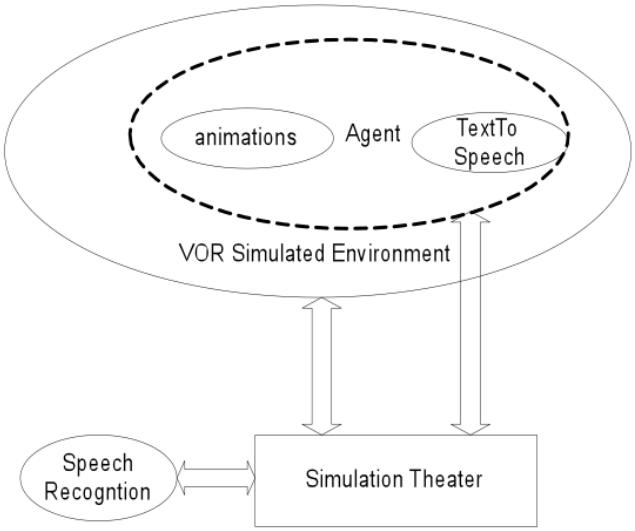


Fig. 4. VOR Simulation Architecture. A Component View

3.1.1) Core (Simulation Theater): Simulation Theater is the central controller and communicator among the modules that are connected as well as provider for computer generated responses. The Simulation Theater also handles the events triggered in the system and therefore is in control of the simulation flow that is dictated by either initialization parameter or the user input. Moreover, simulation output consisting of logs and assessment results are managed and stored by the Simulation Theater for later processing.

One important property of the simulation core is that the core allows portable modules. Hence, modules defined in simulation initialization dictate the structure of the simulation.

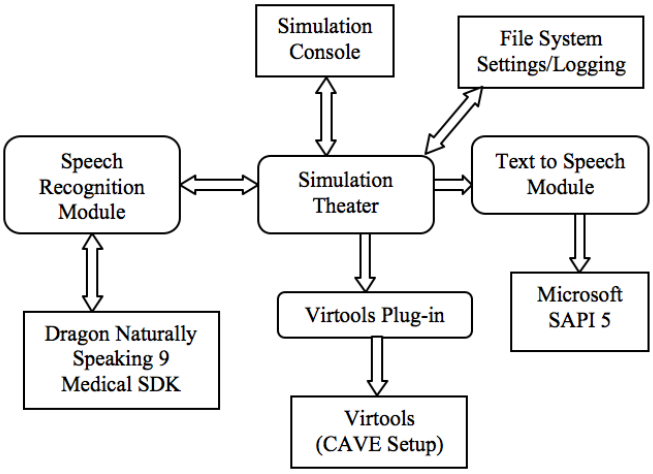


Fig. 5. VOR Architecture. A Package View

Simulation designer can add or remove a device for an individual scenario.

3.1.2) Rendering (Virtools): The rendering for the environment that wraps and constitutes the VOR is provided by Virtools from Dassault Systems [7]. Virtools is capable of rendering the models, created by other design tools (I.e. 3D Studio Max from AutoDesk), to C.A.V.E. (Cave Automatic Virtual Environment) [8] display system. The rendered environment includes computer controlled agents, virtual room of the medical operation and necessary medical equipment. In order to change environment, including agent actions, appearance and sounds of the virtual room, a Virtools Building Block is implemented enabling communication from Simulation Theater. A Virtools Building Block is a plugin implemented on Virtools SDK to run on each rendering client. The Universal Datagram Protocol (UDP) is used to listen command and parameters sent to Virtools via network. Current C.A.V.E. setup has 3 rendering screens. Therefore, 4 Virtools clients exist including the controller.

3.1.3) Speech Recognition: The speech recognition engine used in the speech recognition module of VOR is the Dragon Naturally Speaking 9 Medical (DNS9M) [9] by Nuance Incorporation. DNS9M capabilities are used via an SDK provided by the vendor (DNS9M Client Edition). The module triggers recognition via DNS9M and communicates to the Simulation Theater over network connection, supplying portability of DNS9M installation to a dedicated computer system for the sake of performance issues. DNS9M requires an initial training session for each trainee to increase recognition performance.

The speech recognition module sends the raw recognition results after each utterance from specified trainee. Each result is associated with its probability of recognition provided from DNS9M. Additionally, the simulation core has the control over number of recognition results sent from the module. For example, if the simulation core requested 4 results, the module would send 4 results having the most probability of recognition. This provides the analyzing state machine a better chance of catching ambiguous utterances (E.g. “I scream” and “ice cream”).

It is also possible to have multiple speech recognition modules running simultaneously. Thus, the system have the ability to serve multiple trainees.

3.1.4) Text to Speech: Computer controlled agents interact with real persons in VOR through speech. Speech text may vary on the basis of medical procedure, task of the agent, and the personality of the agent. As a result, speech text is pre-stored per training scenario and converted to speech for an agent by using a computer generated speech engine. For the purpose of the VOR, Microsoft’s Speech API (MSAPI) version 5.0 is selected. Any speech generation tool that support MSAPI 5.0 can be integrated to VOR without any modification other than supplying SAPI name. Currently, several different voices from Neospeech and Cepstral are used to vocalize agents of the VOR. Similar to Virtools Building Block the TTS module uses UDP to listen to Simulation Theater and receive the text to be used in generation of the speech. The text communicated is MSAPI XML coded text, which wraps all the necessary settings for selection voice, and personality as well as the text for speech.

3.1.5) Simulation Console: The Simulation Console provides real time intervention capabilities to the simulation theater. In particular, it provides starting, stopping, loading different simulation setups, simulation logging and access to simulation authoring. A GUI was created to enable easy access to the above capabilities. In case the authoring option is selected, the GUI can scale the displaying resolution to adjust to the number of agents or devices in the simulation.

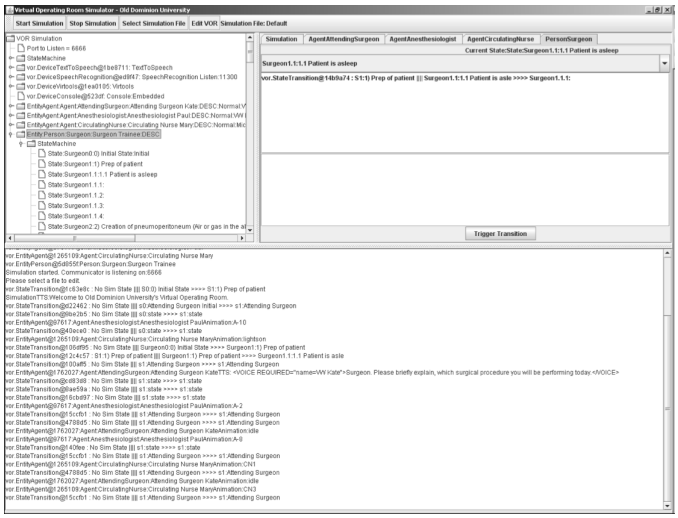


Fig. 6. A Snapshot of Simulation Console User Interface

For better control of the simulation scenario, manual overrides for simulation state and transitions are enabled through simulation console. Simulation console user can observe simulation state, ongoing input/output traffic and past/possible transitions. A snapshot of the simulation console user interface is shown in Figure 6.

3.2. Implementation

In the following sections, some of the implementation details are given for the VOR implementation framework and VOR simulation initialization strategy.

3.2.1) VOR Implementation Framework: The VOR Simulation Theater is implemented in Java. A framework, called the `vor` (Package), is created. The primary objective of the framework is to be able to adapt different simulation requirements. There are several key pieces of the framework including the Simulation Core, Devices, Entities, Communicator, Logger, and Inputs/Outputs. Entities represent both real and virtual people and are further distinguished by three entity types: Agent, Person, and Patient. `vor.EntityAgent` represents computer generated characters in the simulation. Agents process inputs generated from devices or entities in accordance to their state machine [10], [11]. This may result in an output action such as text to speech (TTS) from the agents. To produce TTS Agents would use `vor.DeviceTextToSpeech` after generating personality added text. An actual trainee in VOR, is represented by `vor.EntityPerson`. This is a state machine that processes inputs coming from speech recognition module.

Simulation Core is the main structure that includes all Devices, Entities, Communicator, and is responsible for event management. It also stores simulation state data used by other structures. The `vor.Communicator` is the network manager that manages the UDP connection.

Devices represent different components of the simulation. Each one knows how to manage its local source (E.g. equipment, sensors, etc.) or how to communicate with its peer module that manages the remote source. For example, `vor.DeviceTextToSpeech` can connect to TTS module to deliver Speech XML texts. `Vor.DeviceConsole` creates a user interface that will accept console inputs from the computer operator to manipulate the simulation flow in real time. All devices inherit some common capabilities and properties such as UDP communication, and handling inputs and outputs. VOR employs a decentralized method to calculate the actions of the simulation. By having separate state machines, easy modification, insertion, and removal of entities are supported.

3.2.2) VOR Simulation Initialization Strategy: The VOR XML data model for defining training exercises and agent automatons consists of an underlying VOR XML schema [12], [13]. XML schema usage allows for automatic class binding and a priori input data validation. Figure 7 describes the process of populating VOR specific controller classes.

Java classes are generated dynamically based on the VOR XML schema. These classes can then be populated by unmarshalling (JAXB: Java Architecture for XML Binding [14]) the simulation initialization input file. To isolate the VOR architecture from input changes, objects dynamically generated and populated from the input strategy are not used directly by the VOR Application. Dedicated VOR Java classes poll these Java objects to instantiate and populate themselves.

Table I and Table II show examples of device initialization and entity definition in XML representation respectively. Device *Virtools* has multiple clients to communicate and communication parameters (I.e. host name, port number) are defined in respective XML element. *Attending Surgeon Kate* is an agent entity and her mood is *Normal*. She is using text-to-speech engine*VW Kate*. Her state machine is defined in terms of states and transitions inside entity element of the XML listing.

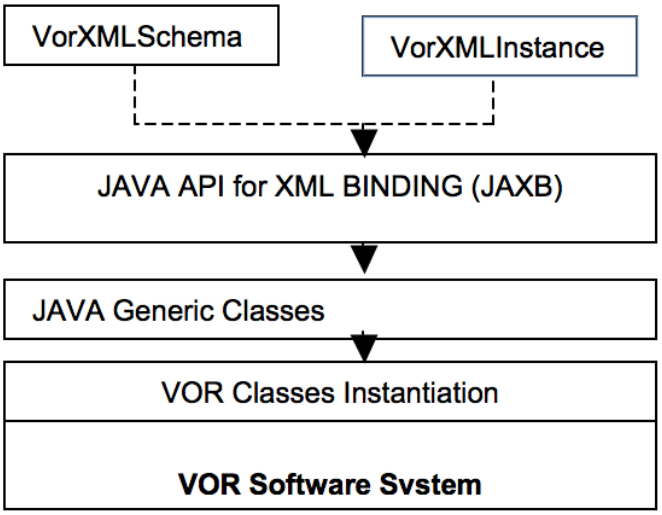


Fig. 7. Simulation Initialization Strategy

TABLE I
DEVICE (MODULE) OPTIONS EXAMPLE LISTING

```
<?xml version="1.0" encoding="UTF-8"
        standalone="yes"?>
<VorSimulationOptions xmlns="VorSimulationOptions"
  xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance"
  xsi:schemaLocation=
    "schema/vor_SimulationOptionsSchema.xsd">
    <SimOptions PortToListen="6666" />
    <Device Type="TextToSpeech" Embedded="false">
        <Client Port="11100" Address="vor2"/>
    </Device>
    <Device Type="SpeechRecognition"
        ListenPort="11300" Embedded="false">
            <Client Port="11200" Address="vor2"/>
    </Device>
    <Device Type="Virtools" Embedded="false">
        <Client Port="11400" Address="controller"/>
        <Client Port="11400" Address="centerwall"/>
        <Client Port="11400" Address="leftwall"/>
        <Client Port="11400" Address="rightwall"/>
    </Device>
    <Device Type="Console" Embedded="true">
    </Device>
    <Device Type="Wiimote" Embedded="true">
    </Device>
</VorSimulationOptions>
```

3.3. Agent Design

The team environment in VOR requires independence of the entities. Thus, agent-based models were selected to represent members of the surgical team in the VOR.

3.3.1) Simulation Agents: VOR employs a decentralized method to calculate the actions of the simulation. Agent behavior, visual animations, and speech are decided in their respective decision making mechanism in the entity state machine. By having separate state machines, easy modification, insertion, and removal of entities are granted (Figure 3). The interactions among entities are supplied by means of inputs and outputs, depicted in Figure 8. Separation of entities enables a dynamic exchange of agents, e.g. substitute a real person for an agent and vise verse, in the middle of the training exercise. In this exchange, all other other elements and aspects of the scenario remain the same.

TABLE II
ENTITY ELEMENT EXAMPLE LISTING

```
<Entity Type="Agent" TTSName="VW Kate"
        Name="Attending Surgeon Kate"
        Mood="Normal" EntityName="AttendingSurgeon"
        Description="DESC">
  <StateMachine>
    <State Name="S0" Initial="true" Description=". . ." />
    <State Name="S1" Description=". . ." />
    <Transition>
      <CurrentState Name="Si"/>
      <NextState Name="Sj"/>
      <Input Type="SpeechRecognition" Text="word 2" />
      <Input Type="Internal" Text="internalMessage 1" />
      <Output Type="Internal" Text="internalMessage 2" />
      <Output Type="Animation" Text="theAnimation" />
      <Output Type="TextToSpeech" Text="say this" />
    </Transition>
  </StateMachine>
</Entity>
```

Agent behaviors can also be modified to reflect different personalities. The personality differences are implemented by substituting different animations and modifying speech generation to reflect the change in personality. In speech generation, changing pitch, volume, and rate allows us to add personality and mood changes.

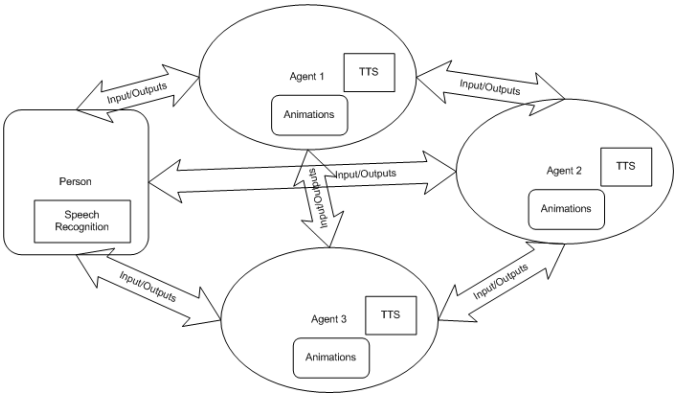


Fig. 8. Agent and Person Interactions

3.3.2) Simulation Authoring: Authoring is accomplished by editing the XML input file for the simulation. To simplify the process, a simple user interface has been implemented. XML file data is displayed in a tree structure simplifying editing simulation components initialization data. Moreover, a visual state machine editor is created enabling drag and drop methodology. However, the user, who is creating the scenario, has to modify XML initialization file manually to define scenario structure, relationship of the entities and simulation parameters. The visual state machine editor helps the user to author the state machine part of the entities. Then, for advanced features of the scenario, state machines are refined by manually editing the XML scenario file. The efforts to enhance scenario authoring are described further in §4.1.

4. FUTURE WORK

There are several improvements and enhancements that can be made to the VOR. The enhancements addressed here are scenario authoring, scenario diversity and interfacing advanced devices.

4.1. Authoring Enhancements

One of the most important future enhancements of the simulation environment is improving how the new scenarios created for VOR simulations. The scenarios for VOR simulations are created mainly by educators in the fields of surgery, anesthesiology and nursing. Since these educators do not have necessary computer and engineering knowledge, the authoring process is a collaboration between educators and VOR developers. Improvements in authoring process would render the need of engineers and computer scientists unnecessary. Thus, future enhancements should target visually enabled computer tools in creating and testing of VOR simulation scenarios. The computer tools would enable scenario creation in terms of meaningful parts (E.g. conversation, Q&A, etc.) rather than automaton states and transitions.

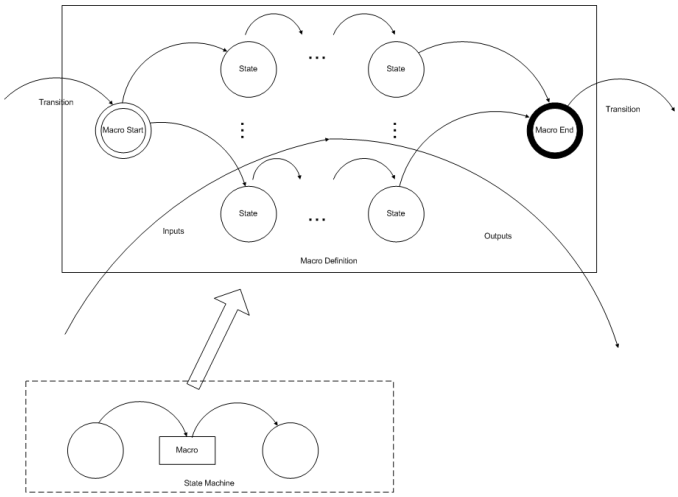


Fig. 9. Macro Definition Applied to a State Machine

The current design for scenario authoring consists of a crude visual editor of the state machine and manual modification of the XML initialization file. One of the enhancements to authoring system is envisioned as an advanced visual state machine editor. The editor would enable user to create stages of the simulation, conversations among agents and trainee, questions and answers section and device triggered actions. These scenario parts would be created by macros which allocate states and transitions necessary to the specific part of the scenario (E.g. conversation between attending surgeon and the trainee). Figure 9 shows concept of macro definitions applied to a state machine. Macro definitions would take parameters such as interacting agents, conversation text, timings between conversation pieces and possible trainee answers. Then, macro definition would create series of states and their transitions to represent desired part of the scenario (E.g. conversation).

4.2. Interfacing with Controllable Skills Simulators

The initial testing and assessment of VOR environment included a crude Laparoscopic Cholecystectomy simulator[15]

from Simulab Corporation (see Figure 1). The simulator is basically a visual model designed for the dissection and removal of the gall bladder. It does not have any type of interface to feed the system about surgical process. Therefore, computer agents in VOR could not use such information unless it is provided by manually via use of simulation console (§3.1.5).

As stated earlier, the simulation architecture supports plugging new devices into simulation theater. Each new type of simulation device, which can interface to a computer system, would have to be implemented as a module in order to connect to the simulation. A complex simulator, such as Laerdal SimMan® [16] could be used for team training of anesthesiologists. In this case, a module that would communicate with the SimMan is necessary. This module would convert the signals from SimMan to inputs for the agents and the simulation core. Similarly, adjustment signals to SimMan can be send from agents or the simulation core.

Using controllable skills simulators inside the VOR would provide better measurement of the simulation state for the simulation core and agents. The VOR simulation would not only depend on trainee speech and simulation console inputs but also read-outs from the used skills simulators (E.g. SimMan, Surgical Simulator for Hysteroscopy[17], LaparoscopyVR Virtual-Reality System © [18] from Immersion Corporation, etc.)

4.3. Render Support with Other Technologies

VOR simulation currently uses Virtools from Dassault Systems as rendering environment for virtual reality component. The agent animations are triggered and rendered in C.A.V.E. by Virtools. Another possible enhancement to VOR simulation architecture is to add support for different rendering technologies. These rendering technologies may include OpenGL [19], [20], Direct3D [21], Java3D [22]. However, using systems such as VRJuggler [23], [24] and Diverse [25], [26] would be trivially simpler task than using lower level technologies like OpenGL and Direct3D. VRJuggler and Diverse provide rapid prototyping for C.A.V.E. like environments in terms of OpenGL applications [27] .

VOR simulation architecture allows modules to be used interchangeably, so that, porting VOR simulation among rendering technologies is trivial. Once the developer implements the module to communicate with rendering platform, the rendering capabilities of the platform is readily available to VOR simulations.

4.4. Application to Other Domains

It is apparent that we can use described agent architecture with other medical team training simulations. For example, operating room can be turned into an emergency room (ER) and ER simulations can be performed by assigning agents and trainee with ER member roles. Moreover, it is also possible to make the patient a part of the interaction among agents and trainee by creating scenarios that the patient is not under anesthesia. In such cases, the patient becomes an agent entity.

VOR team training via computer-based agent can be easily adapted to other domains in which team training is used [28]. Military team training is one of those domains that computer–based agent simulations can be used. A scenario targeting military training can easily be created via use of existing modules and architecture. Similar concept can be applied to team training domains such as education, health care, sports and business.

5. Conclusion

An architecture for VOR training environment is presented. The VOR allows simulated agents to interact with a trainee, and other simulated agents via speech. Each agent has its own state machine. The rendering of graphics and animation, agents, speech recognition, and text to speech functionality is decoupled from the simulation theater.

With the architecture, we provide components and modules that are pluggable, interchangeable agents and trainees, and rapid simulation development tools. The modules can use UDP to connect to Simulation Theater. Components can be created in the VOR Framework. Agents and trainees can be replaced by defining their behaviors via state machine description in a code-free manner. Non-programmers can define their virtual operating room simulation scenario through an easy to use graphical interface.

The VOR was designed to provide a coherent educational experience and facilitate training at both the individual and team levels. The virtual team members are based on a knowledge structure derived from experienced surgical team members and a personality generated from the Five Factor model. A library of virtual agents with unique knowledge structures and personalities can be created providing an unlimited pool of characters that would be difficult to achieve without trained actors. Thus, the VOR provides a forum for surgical team members to sharpen their interpersonal skills when interacting with other team members. Further, because the VOR is a computer-based application it allows instructors to generate an unlimited number of training scenarios and also provides tighter control and greater consistency over the presentation of training experiences. Moreover, all of these learning opportunities can be achieved without putting a single patient at risk.

Acknowledgment

The Virtual Operating Room is a joint project of Virginia Modeling Analysis and Simulation Center at Old Dominion University and Eastern Virginia Medical School. The authors would like to thank present and past project team members: Hector M. Garcia, Leonard J. Weireter, Jr., Michael W. Jackson, Amber Nalu, Elizabeth Newlin, James P. Bliss and Jennifer Seevinck.

References

[1] A. C. for Graduate Medical Education, "Resident duty hours: Common program requirements," http://www.acgme.org/acWebsite/dutyHours/dh_dutyHoursCommonPR07012007.pdf, 2007.

[2] R. M. Satava and S. B. Jones, "Medical Application of Virtual Reality," in *Handbook of Virtual Environments*, K. M. Stanney, Ed. Lawrence Erlbaum Associates, 2002, pp. 937–958.

[3] B. De Raad, *The Big Five Personality Factors: The Psycholexical Approach to Personality*. Hogrefe and Huber Publishers, 2000.

[4] M. W. Scerbo, L. A. Belfore, II, H. M. Garcia, L. J. Weireter, M. W. Jackson, A. Nalu, and E. Baydogan, "The Virtual Operating Room," in *IITSEC '06: The Interservice/Industry Training, Simulation & Education Conference*, vol. 2006(Conference Theme: Training 21st Century Joint Force), 2006.

[5] M. W. Scerbo, L. A. Belfore, II, H. M. Garcia, J. Leonard J. Weireter, M. W. Jackson, A. Nalu, E. Baydogan, J. P. Bliss, and J. Seevinck, "A Virtual Operating Room for Context-Relevant Training," in *Proceedings of the Human Factors and Ergonomics Society 51st Annual Meeting–2007*, vol. 51, 2007, pp. 507–511.

[6] "Laptrainer," http://www.simulab.com/SimuVision.htm.

[7] "Dassault systems 3dvia virtools," 2009. [Online]. Available: http://a2.media.3ds.com/products/3dvia/3dvia-virtools/welcome/

[8] C. Cruz-Neira, D. J. Sandin, T. A. DeFanti, R. V. Kenyon, and J. C. Hart, "The cave: audio visual experience automatic virtual environment," *Commun. ACM*, vol. 35, no. 6, pp. 64–72, 1992.

[9] "Dragon naturally speaking," 2009. [Online]. Available: http://www.nuance.com/naturallyspeaking/

[10] F. Wagner, R. Schmuki, T. Wagner, and P. Wolstenholme, *Modeling Software with Finite State Machines*. Boston, MA, USA: Auerbach Publications, 2006.

[11] D. Hristu-Varsakelis and W. S. Levine, Eds., *Handbook of Networked and Embedded Control Systems*. Birkhäuser, 2005.

[12] J. Roy and A. Ramanujan, "Xml schema language: taking xml to the next level," *IT Professional*, vol. 3, no. 2, pp. 37–40, Mar/Apr 2001.

[13] L. Buck, C. F. Goldfarb, and P. Prescod, "Datatypes for dtds," World Wide Web Consortium W3C, Tech. Rep., 2000.

[14] "Java architecture for xml binding," 2009. [Online]. Available: https://jaxb.dev.java.net/

[15] "Simulab cholecystectomy model," 2009. [Online]. Available: http://www.simulab.com/product/surgery/laparoscopic/cholecystectomy-model

[16] "Laerdal simman mannequin," 2009. [Online]. Available: http://www.laerdal.com/doc/7320252/SimMan.html

[17] K. Montgomery, L. Heinrichs, C. Bruyns, S. Wildermuth, C. Hasser, S. Ozenne, and D. Bailey, "Surgical simulator for hysteroscopy: a case study of visualization in surgical training," in *VIS '01: Proceedings of the conference on Visualization '01*. Washington, DC, USA: IEEE Computer Society, 2001, pp. 449–452.

[18] "Laparoscopyvr virtual-reality system," 2009. [Online]. Available: http://www.immersion.com/markets/medical/products/laparoscopy/index.html

[19] D. Shreiner, M. Woo, J. Neider, and T. Davis, *OpenGL Programming Guide: The Official Guide to Learning OpenGL*. Addison-Wesley Professional, 2005, pp. 7–19.

[20] "Opengl," 2006. [Online]. Available: http://www.opengl.org/

[21] "Direct3d," 2008. [Online]. Available: http://msdn.microsoft.com/en-us/directx/

[22] "Java3d," 2006. [Online]. Available: https://java3d.dev.java.net/

[23] A. Bierbaum, *VR Juggler: A Virtual Platform for Virtual Reality Application Development*. Iowa State University, 2000, ch. 1. [Online]. Available: http://www.vrjuggler.org/pub/Bierbaum-VRJuggler-MastersThesis-final.pdf

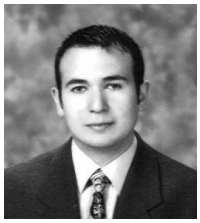
[24] "Vrjuggler," 2006. [Online]. Available: http://www.vrjuggler.org/

[25] L. Arsenault, J. Kelso, R. Kriz, and F. D. Neves, "Diverse: a software toolkit to integrate distributed simulations with heterogeneous virtual environments," http://www.diverse.vt.edu, pp. 1–31, 2001. [Online]. Available: citeseer.ist.psu.edu/arsenault01diverse.html

[26] "Diverse," 2006. [Online]. Available: http://diverse-vr.org/

[27] E. Baydogan, "Rapid prototyping for virtual environments," Ph.D. dissertation, Old Dominion University, 2008.

[28] E. Salas, R. L. Oser, J. A. Cannon-Bowers, and E. Daskarolis-Kring, "Team Training in Virtual Environments: An Event-based Approach," in *Handbook of Virtual Environments*, K. M. Stanney, Ed. Lawrence Erlbaum Associates, 2002, pp. 873–892.



Emre Baydogan, Ph.D. is a doctoral graduate from the Department of Electrical and Computer Engineering at Old Dominion University. He is currently a computer research scientist at National Capital Area Medical Simulation Center at Uniformed Services University of the Health Sciences. His research interests include virtual reality, 3D visualization, and medical simulation. He is a member of IEEE. His Web address is http://www.lions.odu.edu/~ebayd001/

Lee A. Belfore, II, Ph.D. is an associate professor in the Department of Electrical and Computer Engineering at Old Dominion University. He received his BS degree in electrical engineering from Virginia Tech, his MSE Degree in electrical engineering and computer science from Princeton University, and his Ph.D. degree from the University of Virginia in 1990 in electrical engineering. His research interests include virtual reality, medical modeling and simulation. Dr. Belfore is a Senior Member of the IEEE. His web address is http://www.lions.odu.edu/~lbelfore/

Mark W. Scerbo, Ph.D. is Professor of Human Factors Psychology at Old Dominion University. He received his Ph.D. in experimental psychology from the University of Cincinnati in 1987. He is a Fellow of the Human Factors and Ergonomics Society and has over 25 years of experience researching and designing systems that improve user performance in academic, military, and industrial environments. His current research interests are focused on user interaction with medical simulation technology.

Saurav Mazumdar, Ph.D. earned his M.S and PhD. Degrees in Electrical and Computer Engineering from Old Dominion University in 2002 and 2009 respectively. He earned is BE from Bangalore University in 2000. His primary interests are information retrieval, knowledge engineering and artificial intelligence.