Serious Games for Team Training and Knowledge Retention for Long-Duration Space Missions

Sowmya Ramachandran
Stottler Henke Associates, Inc.
1650 S. Amphlett Blvd., Suite 300
San Mateo, CA 94402
Sowmya@StottlerHenke.com

Bart Presnell
Stottler Henke Associates, Inc.
1650 S. Amphlett Blvd., Suite 300
San Mateo, CA 94402
BPresnell@StottlerHenke.com

Rob Richards
Stottler Henke Associates, Inc.
1650 S. Amphlett Blvd., Suite 300
San Mateo, CA 94402
Richards@StottlerHenke.com

***Abstract*—NASA's Johnson Space Center is working with Stottler Henke to improve teamwork skills for long-duration space missions. Numerous factors, however, make instilling teamwork skills into deep-space astronauts extremely challenging. First, due to communication difficulties and mission duration, teams must be trained in a wide variety of domains—from maintenance procedures throughout the ship, to extremely thorough medical training, to science evaluations. Second, there may be a significant delay between when such knowledge is acquired and when it is needed; for instance, several years may pass between an astronaut learning how to diagnose, and to treat, a collapsed lung. Third, training must anticipate a huge range of conditions; the sheer length of deep-space missions when compared to shuttle missions vastly increases the number, and scope, of possible situations that could arise, making it virtually impossible to train for every scenario. To face the myriad challenges of deep-space flight, inflight training must be readily available.**

Our solution is a low-fidelity simulation using game-based training that leverages our experience with intelligent tutoring system (ITS) technology. Game-based training is engaging—it provides an entertaining break from the more onerous aspects of deep-space flight. Moreover, it incentivizes practice of amorphous, but critical, teamwork skills, motivating crews to practice more frequently for either mission-related team training or generalizable team skills training. Finally, a game-based solution addresses the problem of finding the physical space in which to train when a small crew is constrained in a cramped environment for extended periods of time.

This paper reviews the process of creating games based on firefighting on the International Space Station (ISS) and Medical Training. We review the process of analyzing the firefighting procedure to determine the teamwork skills involved; how those skills were mapped to game mechanics; what changes were made to try to make the experience more engaging; and finally, whether these changes were effective.

TABLE OF CONTENTS

1. INTRODUCTION

Instilling teamwork skills into astronauts serving on long-duration space missions presents many challenges, starting with the vast array of domains—from maintenance procedures throughout the ship, to extremely thorough medical training, to science evaluations that crew members must learn due to communication difficulties and mission duration. In addition, there may be a significant delay between when such knowledge is acquired and when it is needed, meaning skills must regularly be refreshed; furthermore, the sheer length of deep-space missions vastly increases the number, and scope, of possible situations that may arise, making it virtually impossible to train for every scenario. To face the myriad challenges of deep-space flight, inflight training must be readily available.

Exploration crews have continually expressed a need for more chances to learn to work together as a team prior to flight, coupled with the need to retain proficiency with limited room to practice during flight. Our solution leverages Intelligent Tutoring System (ITS) technology in conjunction with low-fidelity simulation using game-based training, thereby providing engaging game-based training with the pedagogical benefits of ITSs. Our ITS approach stresses near-term or just-in-time development and maintenance of specific teamwork knowledge, skills, and attitudes (KSAs) that will be applied to imminent tasks. Primary aims are practice and analysis of the teamwork skills that are critical not only for success in particular tasks, but also in maintaining an effective team given challenging circumstances.

Background

The work that the project team has undertaken in developing game-based training can be associated with research conducted on the topic of simulation-based training (SBT). SBT refers to training interventions that use many different types of simulations in order for providing opportunities to teach and administer practice opportunities for learning skills. SBT is an effective approach for taskwork and teamwork training, and is applied in multiple domains including aviation, military, and healthcare [1].

One common misconception is that the characteristics of the simulation—primarily the fidelity, or realism, of the

978-1-4799-5380-6/15/$31.00 ©2016 IEEE

technology utilized for training—have the most impact on training effectiveness. However, based on validation studies of training simulations possessing varying degrees of fidelity, a majority of training experts agree that realism plays a minor role in training effectiveness, with a need for more emphasis on that actual training content [2] [3].

There are two critical requirements for using SBT: (1) applying valid, reliable, and effective metrics to monitor trainee performance, and (2) implementing event triggers—target opportunities for practicing targeted training skills—can help in the training process. Metrics must provide diagnosable criteria regarding the learning performance of trainees, as well as determine what needs to be focused on for improving trainees for future cases. One reason for using event triggers is the measurement of team performance is the most effective for capturing data at the individual and team level [4]. Additionally, event triggers can be designed in order to elicit certain skills associated with training.

Our approach is based on the belief that there are generic teamwork skills that can be trained [5] [6]. Regardless of domain, these skills are required to be an effective member of a team. Different domains and team structures may place more stress on different skills, but the basic skills remain constant. We base our training and metrics on the 4-C model of teamwork. In this model, there are 4 basic skills.

Cohesion: Degree to which team members exhibit interpersonal attraction, group pride, and commitment to the team task.

Coordination: Enactment of behavioral mechanisms necessary to perform a task and transform team resources into outcomes.

Communication: Transactional process by which team members can send and receive information simultaneously.

Cognition: Shared understanding among team members, developed as a result of team member interaction--familiarity with teammate knowledge, skills, and abilities. Cognition includes knowledge of roles and responsibilities as well as of team mission objectives and norms.

2. TECHNICAL APPROACH

The focus of our technical design and implementation has been on supporting the need for domain independence. Our two primary areas of concern are game platform and performance metrics. For the game platform, the architecture uses a very generic system of objects and properties with a collection of actions that can alter the object properties. This approach does place limits on the type and scale of games that can be created. Additionally, until we finish development of authoring tools, the creation of individual games could be more time-consuming than developing a system focused on a particular domain or type of game. However, by choosing this type of domain model, we can support a large number of different domains in developing a wide variety of games, but also in being able to analyze performance in a wide variety of games [7].

The same approach was taken with the development of performance metrics. While there is significant work on domain- specific analysis of teamwork, we wanted to avoid having to generate new performance metrics for each domain. For example, there are existing checklists for assessing medical team communication, but we did not want the tool to be dependent on its ability to analyze the domain-specific situation. For example, we did not want to have to author rigid rules such as in a medical game with a cardiac arrest event, wherein the bedside nurse is needed to supply the latest blood gas measurements to the on-call doctor. Instead, we structured our metrics to use the type of the ration of information requests vs information supplied to make an assessment of how effective communication has been. For example, in the above scenario, we use a ratio of information-supplying actions vs. information request actions to assess how the player in the bedside nurse role is supplying the needed information. This is based on the belief that if the nurse needed to supply the latest blood gas measurements, the nurse would receive a request for that information. This approach allows us to assess how well players are supplying the needed information without having to create domain-specific analyzers to review each communication sent.

We describe below the specifics of the game platform and the elements used in our domain-independent performance assessments.

Game Platform

The overall platform is a basic client server architecture built on existing commercial platforms. This approach provides maximum flexibility in how the system can be deployed as well as flexibility in which platforms can be supported.

Overall System Design

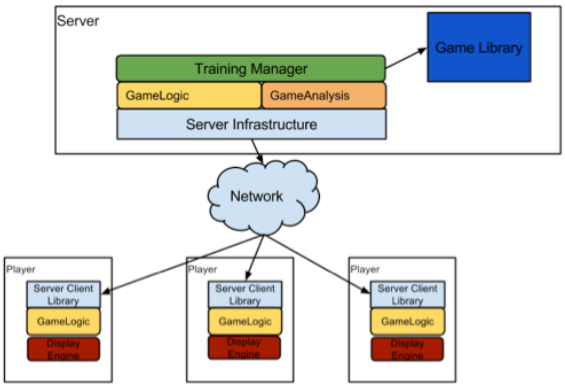


Figure 1. System Organization

Development of a Formal Simulation Structure

We formalized the simulation structure in order to allow

easier construction of future domains and to enable analysis of actions for performance assessment and review.

The simulation world consists of two classes of entities: Actors and Game Objects. Each Game Object is associated with Properties. Actions are links between entities. A Source and a Sink define an Action. The Source property of an Action defines who or what initiates the action. The Sink property defines who or what is affected by the action. There are four types of actions:

- Actor to Object: This represents an action taken by personnel that affects one or more properties of an Object. E.g., extinguish fire, shut down electric subsystem.

- Object to Actor: This represents information about Object properties passed to personnel, e.g., fire port code, sensor readings.

- Object to Object: This represents automatic processes in the simulation, e.g., shutdown of electric subsystem resulting in putting out a fire.

- Actor to Actor: This represents information and communication between personnel, e.g., passing code from assistant to firefighter. Once the simulation model is defined, a communication model or a model of how information is passed around can be generated. Much of the communication model can be generated automatically from the simulation structure definition. Figure 2 below shows an example of a communication model. The Game Engine tracks the frequency of activation of each link as a game unfolds. The communication model is useful for assessing team communication in-scenario. It is also useful for determining coverage. For example, if players never trigger some communication links, then future training may need to be adjusted to direct the players towards these links.

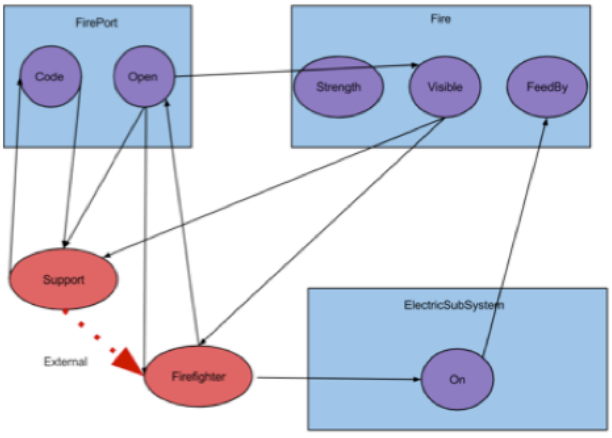


Figure 2. Example Communication Model

Game Engine

Significant changes have been made to the current game engine during this project. First, we have made a significant change in the environment: We designed the initial effort to be dark to limit the information available to any one player and to force communication. We determined this dark environment, however, to be too confusing to players. A first-person perspective within a maze environment has since been found to be sufficient to keep any one player from having complete knowledge of the game environment. Changes have also been made to supply a mini-map display of the environment. This map has simplified navigation within the environment. The map is limited to only the information the player had discovered or had been shared with the player, which makes the domain significantly more a test of sharing information effectively than of individual maze-solving skills.

Features have been added to allow the players to perform explicit information-based actions. Rather than allowing just unstructured communication via the chat window, the game allows for explicit information actions. There are four types of information actions:

Send Information

a. Example – Source of a fire

Request Information

b. Example – The Code For A Port

Assign a task

c. Example – Shut down a component

Acknowledge

d. Example – Acknowledge any of the incoming communications

The final change to the game engine has been the inclusion of the golden snitch. This is a random object that appears within the world. The snitch is an entirely individual action. The goal of this functionality is to provide a distraction from the main goal of extinguishing the fires. This current functionality is temporary in nature and will ultimately be replaced by an action more applicable to the domain.

Metrics

Over this project, we have also expanded the metrics used to evaluate team performance. First, we implemented infrastructure to support directly querying the players about their knowledge of the expected team behavior. The query questions and responses are derived from the existing communication models, meaning we expect this functionality should translate seamlessly to additional domains. The frequency of the query adapts to the players' performance. The system focuses queries on those concepts and players that have shown some errors.

The second set of metrics is based on the number and ratio of action types. We have added the ability to tag actions with type information. The tag types are:

- TEAM
- INDIVIDUAL
- INFORMATION
- REQUEST
- ASSIGNMENT
- ACKNOWLEDGEMENT

The metrics being measured are: INFORMATION vs. REQUEST – The ratio of information being supplied against the amount of information being requested. If more information is being supplied than is requested, this shows the team is anticipating the needs of other team members.

TEAM vs. INFORMATION – The ratio of team-focused actions vs. individual actions. This serves as a measure of team commitment and focus.

INFORMATION + ASSIGNMENT vs. ACKNOWLEDGMENT – The ratio of information provided versus the number of acknowledgments. This serves as a measure of the focus on closed-loop communication.

TEAM vs. ASSIGNMENT – The ratio of team goal actions vs. the number of assignment actions. This serves as a measure of how focused the team is on coordination.

Furthermore, we have embedded research-based team performance assessment metrics within the fire-suppression game, in collaboration with our consultant, Dr. Eduardo Salas, who offered recommendations for assessment metrics and scoring criteria based on a thorough analysis of team training research.

Following the 4Cs model of teamwork, the assessments metrics we have added fall into the following categories:

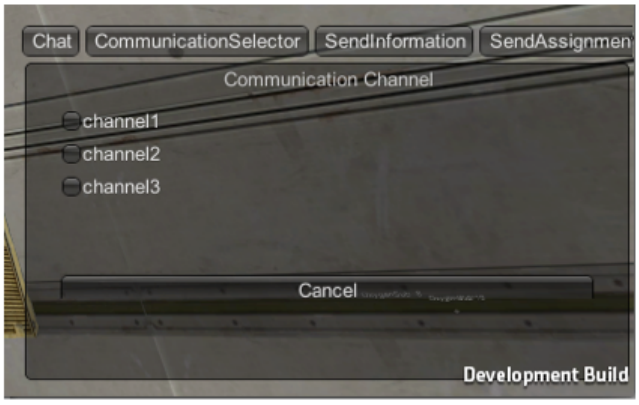


Figure 3. Communication Channel Selector

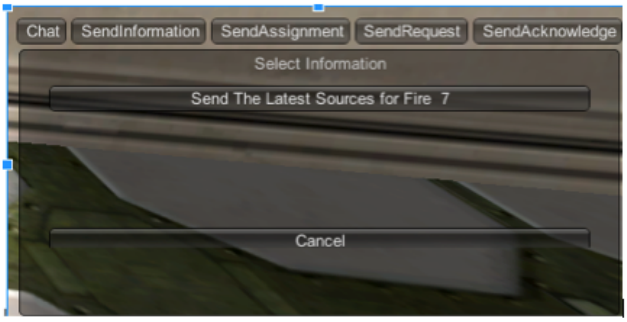


Figure 4. Send Information Screen

Players can also acknowledge receipt of information from their teammates, as seen in Figure 5 below.

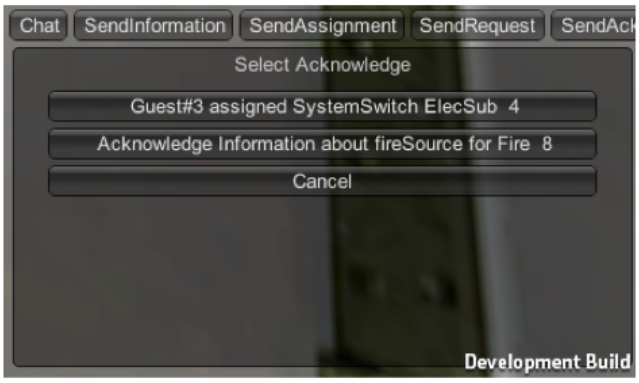


Figure 5. Screen For Acknowledging Message

The following scores are computed to assess a player's team-communication skills:

- The ratio of the number of "acknowledge" communications a player sends to the total number of communications that team member sends and receives.

- The ratio of the number of information requests sent by each member of the team compared to the number of information messages sent by each team member. The ratio is a team-wide metric showing how effective the team is at efficiently providing information.

Cohesion—Cohesion is assessed by introducing the notion of individual game goals—in addition to team goals. The team goal for the fire-suppression game is to put out all the fires in the space station. The new version of the game additionally includes individual goals in the form of experiments that players are responsible for. Periodically, players will be called upon to monitor or perform some actions related to their experiments. Sometimes attending to the individual goals will conflict with attending to the team goals, and a player's sense of cohesion with the rest of the team will be reflected in how they choose to resolve this type of conflict. For instance, do they compromise on the individual goal in favor of the team goals—or vice versa? A player's score on the cohesion dimension of team

performance is computed as a ratio of individual actions to the number of team actions they perform.

Cognition—Team cognition is related to players' mastery of their own tasks and knowledge of others'. It also relates to situation awareness. To assess this dimension of team performance, we introduced direct querying mechanisms whereby each player is given a multiple-choice question to indicate their knowledge or situation awareness. The frequency of these questions is modulated by the player's performance both on the game and on prior multiple-choice questions. The questions are auto-generated based on the model of actions represented within the game.

Coordination—Coordination is a measure of how effectively a team orchestrates its activities to achieve its goal, and a large part of coordination consists of knowing when and to whom to delegate and when to execute a task oneself. To assess coordination, we have introduced a scoring metric that reflects the ratio of the number of assignment actions performed by a player to the number of execution actions. It is also based on measuring the percentage of valid assignments made by a player. Invalid assignments are those a player is unable to perform. For example, assigning a firefighter to determine the source of a fire would be invalid.

We have, furthermore, refined the assessment metric to include measure of communication efficiency. The goal: to find an assessment that rewards both the effectiveness of the communication as well as the efficiency of information sent. Our previous metric used the numbers of requests for information in ratio to the number of actions performed. While this was a reasonable measure of how well team members were supplying the information needed by other teammates, it also rewarded behavior wherein significant amounts of extraneous information were being delivered.

The most successful metric we have developed is a simple weighted sum between the ratio of actions to requests as well as the ratio between actions to bits of information sent. This simple formula seems to reward the desired behaviors. This formula also allows the metric to be adjusted to different domains where the costs of requests or of sending information may vary.

3. EDUCATIONAL TOOLS

The technical approach to the game engine and metrics allows us to develop training games for a variety of domains and to assess team behavior in these games. However, this is only a partial solution—we must also support the educational process. There are two key sets of tools we need to supply in support of the educational goals. The first is a debriefing tool. The current standard for most effective team-based training involves allowing highly trained proctors to conduct the post-game debriefings. Research has shown the quality of the debriefing to be critical to the effectiveness of the game as teamwork training tool. Our initial approach is to provide tools for the team to review performance, rather than initially attempt to fully automate the debriefing process.

The second area we must address is determining the appropriate content to use for training teams [8]. We are developing two approaches to determining the appropriate content. In the first approach, we look to match existing games with descriptions of new domains; the second approach is to expand the scripting capabilities. These scripting capabilities will have a twofold focus: First, we will provide the ability to adjust the actions and information available to each role. Second, we will add capabilities to control the type of in-game events to be generated. These changes will allow educators to better tailor the game to the skills they wish to focus on [9].

Visual Debriefing Tools—We have included visualization of assessment metrics to facilitate end-of-the-game debriefings (Figure 6). These allow visualizations of how the performance assessment metrics change over time and are generated from game performance data logs after the completion of a game.

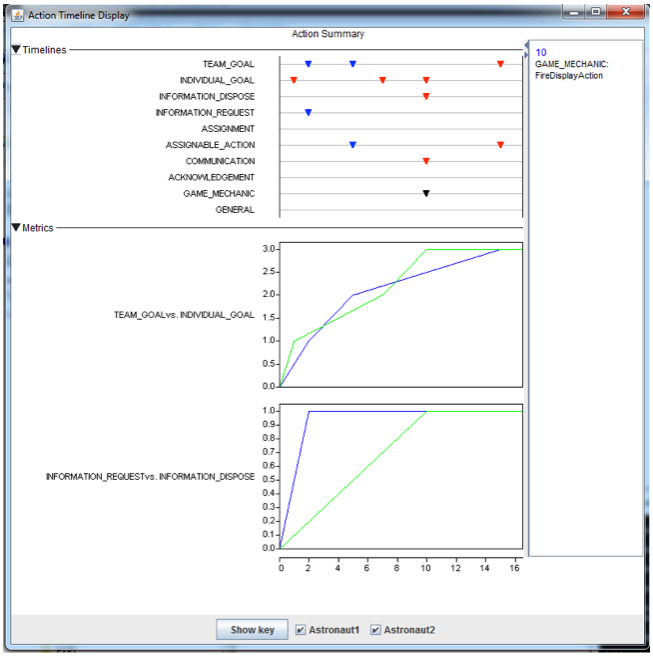


Figure 6. Debrief Tool Showing Events And Metric Value Over Course Of Simulation

Content Selection

Game Matching and Retrieval—One of the innovations of this effort has been an approach to auto-generating games for new team configurations by modifying existing games. Critical to this approach are algorithms for mapping game characteristics to team configurations and for matching team configurations to identify similarities and differences. We have developed a graph-matching algorithm to determine the "best match" between existing game domains and new training domains. The algorithm extracts features in each

domain and uses these to measure similarity. The model of roles and objects and the required information communication between them automatically construct the feature set. We have implemented a basic matching function.

The matching algorithm determines the features in a desired team structure and interactions. These features are based on the number of roles, the actions available to each role, and finally on the information that is gathered, used, or passed by each action. These features are then compared to the existing library of games. For each existing domain, a score is calculated as to whether the existing domain matches each feature of the new domain. These feature-matching scores are combined to determine an overall matching score. Once the existing domain that most closely matches the new domain is found, a matching is determined between the roles of the new domain and the existing domain. The existing domain can then be used as the basis of training for the new domain.

For example, the team may need to train for an exploration task, in which one team member must monitor a sensor reading while providing instructions to another team member operating a tool. This instance of team interaction can be based on the firefighting game, as opposed to another game where all players know all the information, and the interaction is based on efficiently assigning tasks.

Event Scripting to Test Specific Skills

While there is great value in allowing the system to match content to the particular team configuration, there is a need to allow team trainers to generate scenarios that train teams in very specific scenarios. To support this requirement, we have developed a Domain Specific Language that allows trainers to configure the domain-specific behaviors as well as specify the team configuration, the actions and information available to a specific player, the metrics to use for a scenario, and the events to trigger the team-specific behaviors.

For Cohesion, there are two script elements that are of particular interest. First, there is the ability to add new tasks to particular players using simple commands. This mechanism allows for scenarios where an individual team member can be overloaded, and an event is triggered where supporting behavior from teammates is needed.

The scripting allows the success of the team to be measured in several ways. First, trainers can specify metrics based on task state, either by measuring task statistics or insuring that specific actions have been performed for each task. Second, trainers can specify metrics that measure the actions taken. The trainer can either measure the delay between actions or the prevalence of certain actions. For example, in a medical domain, the trainer can specify reports that detail for the number of patient transfer actions that occur, or they can configure the scenario to report the delay between the patient being added and a patient being transferred. Finally, the scripting language allows for the specification of the surveys that can be given to the players. These can be surveys that allow players to rate either the overall performance—or specific aspects of the performance—of the team or individual. The trainer will receive reports on how the team responded.

For Communication, the scripting language allows for configuring scenarios that force communication between players. Trainers can script which pieces of information particular players can receive. The scripting language also allows tasks and other items to be transferred between players. Both of these features are intended to force the players to communicate about the state and requirements of tasks.

There are two features for measuring the effectiveness of communication. First, the task status can be the metric. This does not measure whether the appropriate communication protocols are used, but it does measure the effectiveness of the communication. We also have a metric to analyze the patterns of communication: this analysis tries to determine the amount of closed-loop communication as well as determine which players are communicating.

For Cognition there are scriptable events that can stress the players' shared cognitive models. The script elements allow trainers to construct scenarios wherein an initial plan must change as resources are removed or new tasks are added. Additionally, communication capabilities can be removed from individual players—forcing players to make decisions without the ability to consult with other players.

To measure the cognitive performance, we can again use metrics on the state of the tasks as well as direct querying of the player. The scripting also allows for traditional task order surveys. These surveys ask each player to order the tasks involved in a procedure. The trainers can be given a report showing how similarly each player in the team ordered the tasks.

There are also mechanism to force Coordination between the players. The script makes available to the players different actions they can perform as well as allowing them access to different supplies. These script mechanisms allow the trainers to specify situations wherein players must coordinate actions so that all tasks can be completed.

To support the extensive scripting capabilities of the system, we have developed modifications to the popular open-source software development environment, Eclipse. We developed the scripting language by designing a Domain Specific Language in the Groovy programming language. Because the scripting language is essentially Groovy, we can take advantage of the code completion capabilities of Eclipse. In addition, we have developed code templates that can be used by the Eclipse editor. These templates are added into the context sensitive menus of the Eclipse text editor. Scenario authors can insert the templates and tab through

the elements in the templates. This approach makes it very easy to quickly develop a scenario.

4. FUTURE WORK

With these new tools fully functional, we plan to expand the library of games we have in two different ways. First, we intend to develop a game for a very specific domain. Then we will look at a very generic domain that allows for developing games that test a wide range of teamwork skills without any domain background.

New Medical Care Game Design

We first began development of the parameters for a new game in the area of medical care focused on diagnostic protocols for cardiovascular issues and designed to teach the importance of full assessment before diagnosis and treatments; of clear assignment of roles and responsibilities; and of closed-loop communications. Having defined the parameters of the game, we commenced development of the storyline and gameplay.

It has not been not our goal to develop a sophisticated, engaging, challenging high-fidelity simulation. Instead, offering large numbers of simple simulations provides the challenge and engagement. The game, we decided, should require players to alternate roles of assessing, treating, and monitoring patients. In the finalized overall design for the healthcare teamwork game, the focus of the game is on three modes of communication that are essential to a functioning medical team—either for deep space missions or teams in standard medical environments. The three types of communication we focus on are:

- Basic communication between nurses and doctors.
    - Providing practice at using SBAR communication. Nurses must supply four pieces of information in communication with doctors:
        - Situation;
        - Background;
        - Assessment; and
        - Recommendation
- Patient History.
    - Nurses must be able to quickly and efficiently transfer care of patients from one nurse to another. The game design is meant to ensure that these transfers happen, and happen quickly.
- Code Management.
    - Medical teams must be able to quickly manage these life-or-death situations. A significant part of the success of the code is having a team member be able to take over the code and effectively assign team tasks to other members. This is a teamwork skill critical to any medical team. Even highly trained, experienced medical teams typically take the time to review and practice code management.

Several mini-games comprise the overall game experience. The purpose of these mini-games is to mimic the workloads associated with actual medical environments. The transitions into and out of the mini-games provide opportunities and challenges in transferring patient history and care instructions between nurses. The mini-games are designed to accomplish two tasks: First, they are meant to mimic the types of skills needed for different nursing roles. For example, the patient management game has been envisioned as a Tetris-based game. The basic tasks to be done for each patient are relatively simple; the difficulty comes in managing the needs of a number of patients. The second goal of the mini-games is to provide an opportunity to refresh medical knowledge. For example, some of the mini-games may provide reminders about the steps and checks that must be performed. A game based on feeding in a central line, for example, would require the player to go through a simple version of the procedure:

> *Find the correct entry point.*
> *Feed the line into the central vena cava.*

There are three player roles. We expect that the majority of the players will be floor nurses. These nurses must transfer patient management when they switch from the central management game to other procedure games. While playing procedure mini-games, the nurses will not be able to manage other patients. This constraint is what forces significant transfer of patients between nurses. The second role is the charge nurse, who must take a basic history and coordinate the transfer of patients between nurses. The final role is that of the doctor. The doctor will take updates from nurses and require procedures to be done. The doctor can also change the management tasks associated with a patient.

The assessment makes use of our existing statistical measures. Also included are measures of the speed of play in the mini-games, allowing us to assess how much overall improvement is the result of improved teamwork and how much is the outcome of improvement with game mechanics.

Generic Team Skill Game

The envisioned game would be a simple grid-based navigation game. The goal of the game would be to navigate through the world to reach a required destination while completing sub-goals (i.e., gathering items) in the virtual environment. The world itself would consist of several different cells:

(1) Standard Cell – a cell that can be crossed and can contain items that can be picked up.

(2) Wall Cell – a cell that cannot be crossed and cannot contain items.

(3) Gate Cell – a cell that can be crossed when the gate is unlocked. It cannot be crossed when locked. It cannot contain items.

(4) Lock Cell – a cell that can unlock a gate for a limited period of time. It cannot contain items.

(5) Destination Cell – the goal of the navigation task. When all avatars reach this cell, the level ends.

The instructors would be given significant control over how the world is configured in order to manipulate complexity and difficulty, including:

Specify each cell type

(6) Place items within the world

(7) Connect lock cells to gates

(8) Adjust how long gates are open

(9) The number of avatars in the world

Using the new scripting flexibility and control over scenarios, instructors will be able to create teamwork practice opportunities based on their assessment of training objectives. For example, the specific workings of the lock and gate cells can create problems that test each of the teamwork skills. If the instructor configures the scenario so only Player 2 can see the gates, and only Player 3 can see the locks, they have created a scenario eliciting teamwork communication (i.e., the players must communicate what elements are where for the other team member to operate). On the other hand, if the world is configured so only Player 2 can operate the locks, the level becomes a coordination problem. When the instructor can configure the world so there is only one avatar and control over the avatar's actions and movements rotates between members of the team, and there is communication cutoff between teammates, the game becomes a cognition problem.

Improve Authoring Environment

Building upon the code completion tools within Eclipse has made the task of writing new scenarios relatively easy. However, there can still be significant cognitive demands in making sure the scenario scripting is logically consistent. We would like to build upon the tool infrastructure of Eclipse to allow a phase of logical analysis on the scripts. This analysis phase would be used to detect any logic errors in the scenario before testing the scenario within the game.

A second area of research is to generate scenarios based on high-level preferences from trainers. Our hope is to develop a system that can generate new scenarios based on the types of teamwork problems the trainers want to work on. We will generate these as scripts rather than directly within the game itself. This approach will allow the content generation to be insulated from the specific game engine as well as allowing trainers to build upon and fine-tune auto-generated scenarios.

5. CONCLUSION

This project has demonstrated that it is possible to develop a game framework that supports the assessment of teamwork in a domain-independent manner. While the framework does require that the new actions, objects, and properties are defined for new domains, the teamwork assessment can be done without having to specify complicated domain specific expert based assessment rules. This approach makes it significantly easier for this tool to adapt to a variety of new domains.

While this domain-independent assessment and platform is critical to the viability of this system, there is still significant work to be done to meet the needs of teamwork educators. To make the training process more efficient, we must allow educators the tools to control which teamwork skills are being tested and the level of challenge being issued.

Having both a domain-independent assessment as well as a sophisticated set of curriculum controls will allow team training to be applied to a variety of domains. Teamwork training has generally been a secondary concern, but with the expansion of communication and technology, more and more tasks have become cross-domain. Medical decisions must involve decisions made by members from different disciplines; nurses, surgeons, pathologists, pharmacists, social services, technologists, and administrators must all collaborate to make life-and-death decisions. Teamwork skills have become critical to making this diverse group with different expectations function in a highly stressful environment. Tools such as we are building will be critical additions to training in all of these domains.

REFERENCES

[1] Salas, E., Wilson, K. A., Burke, C. S., & Priest, H. A. (2005). Using simulation-based training to improve patient safety: what does it take? *Joint Commission Journal on Quality and Patient Safety*, *31* (7), 363-371.

[2] Jentsch, F., & Bowers, C. A. (1998). Evidence for the validity of pc-based simulations in studying aircrew coordination. *International Journal of Aviation Psychology*, *8*, 243-260.

[3] Maran, N. J., & Glavin, R. J. (2003). Low- to high-fidelity simulation - a continuum of medical education? *Medical Education*, *37*, Suppl 1:22-8.

[4] Salas, E., Rosen, M. A., Held, J. D., & Weissmuller, J. J. (2009). Performance measurements in simulation-based training: A review and best practices. *Simulation and Gaming*, *40* (3), 328-376.

[5] Salas, E., & Burke, C. S. (2002). Simulation for training is effective when… *Quality Safety Health Care, 11*, 119-120.

[6] Salas, E., Oser, R. L., Cannon-Bowers, J. A., & Daskarolis-Kring, E. (2002). Team training in virtual environments: An event-based approach. In K. M. Stanney (Ed.), *Handbook of virtual environments: Design, implementation, and applications* (pp. 873-892). Mahwah, NJ: Lawrence Erlbaum Associates.

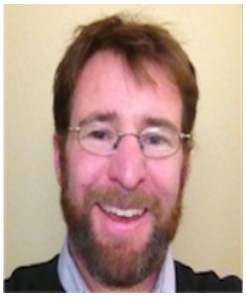
[7] Rieber, L. P. (1996). Seriously considering play: Designing interactive learning environments based on the blending of microworlds, simulations, and games. *Educational Technology Research and Development, 44*, 43-58.

[8] Weaver, S. J., Wildman, J. L., & Salas, E. (2009). How to build expert teams: Best practices. In R. J. Burke & C. L. Cooper (Eds.), *The Peak Performing Organization* (pp. 129-156). New York: Routledge.

[9] Volpe, C., Cannon-Bowers, J. A., Salas, E., & Spector, P. E. (1996). The impact of cross-training on team functioning: an empirical investigation. *Human Factors, 38*, 87-100.

BIOGRAPHY

Sowmya Ramachandran *received a Ph.D. in Computer Science from University of Texas at Austin in 1998. Dr. Ramachandran's interests focus on intelligent training and education technology, including intelligent tutoring and intelligent synthetic agents for simulations. She is also interested in issues of motivation and metacognition. Experience with military and private industry gives Dr. Ramachandran a unique perspective on the needs and requirements of the ultimate end-users and their constraints. She contributes expertise in AI, instructional systems, probabilistic reasoning, and knowledge management. She has developed ITSs for a range of topics including reading comprehension, high school Algebra, helicopter piloting, and healthcare domains. She has participated in workshops organized by the Learning Federation, a division of the Federation of American Scientists, to lay out a roadmap for critical future research and funding in the areas of ITSs and virtual patient simulations. She has developed a general-purpose authoring framework for rapid development of ITSs, which was used to develop an intelligent tutor for training Navy Tactical Action Officers. She has also developed tools and technologies for training emergency first responders.*



Bart Presnell *received an MS in Computer Science from Georgia Institute of Technology in 2004. Mr. Presnell has performed as a lead software engineer on numerous behavior modeling projects at Stottler Henke. His research interests include applying these behavior cognitive models to game-based training systems, most recently completing work on a behavior model for air combat tactics. Past projects have included efforts to create an online virtual world aimed at reducing childhood obesity (www.creature101.com) and an automated planner to model a variety of red forces in an effects-based air campaign simulator. The planner allowed cognitive models based preferences to guide search in both hierarchical task network planning as well as operator-based planning. Mr. Presnell's research interests focus on agent architectures, agent-based simulation, planning, and multi-agent coordination. He is particularly focused on applying these technologies to entertainment and educational interactive simulations. Past research efforts have involved team coordination for Aibo Robots in the Robocup competition and modeling communication behavior of emergency operation personnel during a crisis situation. Prior to joining*

Stottler Henke, he worked for 7 years as a software engineer for game development studios such as Electronic Arts and Stormfront Studios. He created Agent architectures and behaviors for agents in a variety of games, including sports, driving, puzzle, action, and combat titles.

Robert Richards *received a Ph.D. in Mechnical Engineering from Stanford University in 1995. Dr. Richards is managing and has managed multiple projects for both commercial and government clients, including various intelligent-tutoring-system-based training projects. Dr. Richards is the Principal Scientist and Manager of Stottler Henke's Navy helicopter training contract, OMIA. OMIA is a PC-based desktop training system that teaches crewmembers the Navy's new MH-60R and MH-60S helicopters. Dr. Richards has taken OMIA from a Research and Development SBIR project to a deployed training tool that has been awarded over $6 million in Phase III funding. He was also the PI for INCOT, an Air Force project that developed automated tools for network layout. These projects exemplify his wide range of research and application area interests, including: training system development; applying automation and artificial intelligence techniques; and decision support tool development for life-critical situations. Dr. Richards has publications in all of these domains.*