Distributed, Game-Based, Intelligent Tutoring Systems - The Next Step in Computer Based Training?

Jeff Craighead
Department of Computer Science
University of South Florida
Tampa, FL 33620
craighea@cse.usf.edu

ABSTRACT

This paper proposes what may be the step in computer based training, a distributed, game-based intelligent tutoring system, and discusses possible complications with creating such a system. Recent research has demonstrated a positive effect on learning by integrating intelligent tutoring systems within virtual worlds and video games as teaching aids. These systems cover many topics including reading and math for elementary students as well as computer programming, physics, and medicine for college level students. As well, there has been work demonstrating the positive outcomes of using web-based tutoring systems. While these systems have shown that computer based training is effective, none have investigated multi-user tutoring systems in which teams of players work together within a virtual world to solve more complex problems such as operating a multi-operator robot. Such a system could increase a students understanding of a subject by allowing them to discuss the problem in real time with the other players on their team.

KEYWORDS: Cooperative Games, Training, Intelligent Tutoring Systems

1. INTRODUCTION

Are distributed, game-based, intelligent tutoring systems (ITS) the next step in the evolution of computer based training applications? This paper proposes a framework and discusses the difficulties that must be overcome to create such a system. A distributed, game-based tutoring system would combine the concepts from game-based and web-based tutors with online multi-player gaming technologies. Such a system would provide anytime, anywhere access to a customized, immersive environment in which the student can (or must) work with peers to learn about and improve their skills in a given subject area.

Game-based and web-based intelligent tutors assess a student's skill level by watching the student attempt to solve one or more problems within a domain. Game-based tutors have the advantage of providing an immersive, interactive environment for the student to explore a problem; web-based tutors typically provide only a text based interface and possibly pictures, video, and diagrams. On the other hand, web-based tutors have the advantage of providing a student access to their customized program anytime, anywhere, so long as an internet connection is available; game-based tutors are typically installed on one machine and keep the student's profile locally. By providing an anytime, anywhere, game-based tutor, students will have better access to a more engaging learning environment.

Commercial computer games do not make any attempt to provide a learning environment, however there has been a great deal of work to connect players together, track individual and team performance, and share this information with other players in the hopes of sparking competition. Competition has been shown to be a factor in the popularity of online games. The goal of combining this aspect of multi-player gaming with game-based tutoring systems is to increase the student's desire to play the game and cooperate with their teammates.

The high level framework presented in this paper is for a multi-player, game-based, intelligent tutoring system based on the development of the Search and Rescue Game Environment (SARGE). Section 2 discusses the existing game-based intelligent tutoring systems and web-based intelligent tutoring systems for training and education. Section 3 discusses the motivation behind the investigation into distributed, game-based intelligent tutoring systems in the context of SARGE. Section 4 discusses the proposed framework with examples from SARGE. Section 5 discusses issues that must be overcome for a distributed, game-based ITS to be effective. Finally Section 6 provides a summary of the this work and discusses future work in the development SARGE.

2. RELATED WORK

This section reviews recent work on game-based and web-based tutoring systems. Students who used these systems showed improved results for a particular task versus students who did not use the system. Each system is described along with the results presented in the cited works. While there are other multi-player educational games such as Wang, et al.'s "Lecture Quiz"[1] which are used in the classroom, this section limits its focus to games that include intelligent tutoring systems. The section summary provides a discussion of the pros and cons of these systems.

2.1. Game-Based Tutors

Previous work by the author [2, 3] reviews the most popular commercial and open source robot simulators and develops a rating system to categorize each simulator based on its features. This section expands on this work by discussing several game and simulator based intelligent tutoring systems used for training in professional and educational environments[1]. The overwhelming majority of the work with game based intelligent tutoring systems has been focused on children in the primary and secondary education market leaving the professional market largely ignored. Because of the overwhelming support for games as educational tools for children, [4, 5, 6, 7, 8] the potential for training adults using similar techniques is worth investigating.

The simplest game based intelligent tutors provide after action reports, letting a student know what they can do better to improve their scores in the next round of play. More complex tutoring systems attempt to identify a student's problem areas and give suggestions or change the content of the curriculum over the course of a game to help the student understand a given topic. The challenge with the later is that it is often difficult to determine what the student is having a problem with and how to modify the game to correct the student's misunderstanding of the topic. In simple educational games this is much less of an issue as there are clearly defined steps to solving a math or science problem. However when more complex tasks are involved there may be no clear means of accomplishing a given task [9]. It may be necessary to break the task down into component parts and tutor the student on these smaller items. For instance, the case of flying a non-autonomous unmanned aerial vehicle (UAV) the tutor may determine that the student is having trouble maintaining altitude or hovering because of jerky input movements. The ITS could just give suggestions to the student or it could provide assistance by damping their input motions to prevent the student from becoming frustrated.

[1]An educational environment is defined as one that would be used in a primary or secondary educational institution. For example there are many video games targeted towards students in primary grades that teach basic reading, writing, arithmetic, science, etc.

Gómez-Martín, et.al. [7, 10] created a 3D role playing (JV^{2}M) game which teaches students how the Java Compiler and Virtual Machine (JVM) operate. The 3D environment is a town populated by structures representing various parts of the JVM and non-player characters (NPC) who are trapped inside the town. The user is charged with freeing the NPCs by executing increasingly difficult Java programs. One of the NPCs provides hints to the player if needed and can solve each problem if the player cannot. Additionally the paper identifies several typical game features that may help hold the users interest in the game such as interactive tutorials instead of long user manuals, the ability to save and load a game in the middle of play, checkpoints that alert the player to their accomplishment and provide a temporary save location, and a winning condition to let the player know they know everything the tutor can teach. These features are often left out of game-based tutoring systems.

Stottler, et.al. have a tactical simulator used by the U.S. Navy which features an intelligent tutoring system [11, 12, 13]. Their work on incorporating intelligent tutoring systems into simulations has shown that it is possible to create effective intelligent tutors for complex scenarios. The goal of these simulations is to teach students in the Navy's Surface Warfare Officers School surface warfare tactics and proper operation of equipment used by a Tactical Action Officer (TAO). Combined with 3 months of classroom courses, the students learn and practice tactics and operational procedures they will use onboard a real ship to make life and death decisions in times of hostility. The simulation presents the player with all the information that would normally be supplied to subordinate officers as well as a means of issuing commands in side panels around the main interface. This forces the student to somewhat take on the role of each of the subordinate positions to better understand their role. The main interface supplies only the information that would actually be available to the TAO. The intelligent tutor allows students to request an online evaluation of their use of various systems on the simulated ship. Additionally at the end of a mission the ITS summarizes the student's performance and provides suggestions on what scenarios the student should choose next.

Burmester, Stottler, and Hart [14] propose a proof-of-concept intelligent tutoring system for use in embedded training situations for the Future Combat System (FCS) C2 vehicle. Embedded training is defined as training using simulated scenarios within the actual vehicle used in the field; this allows soldiers to train while in the field and en-route to a mission. The ITS is used to take the place of an instructor since it is unlikely that an instructor would be present to present problems, analyze the soldiers battle plan and point out mistakes. In this case the soldier is assumed to have prerequisite knowledge on the use of the C2 vehicles battle planning software. The goal of the embedded training system is to make sure the soldier will make the correct tactical decisions in various combat scenarios. An example presented in the paper is the use of a UAV to scout an area obscured by high terrain. If the UAV is not used, this is considered an incorrect tactical decision. The ITS determines if a soldier is making the correct decisions by comparing his battle plan to that of several correct and incorrect plans created by an expert. In the case of a good plan by the soldier, the ITS chooses the expert plan most similar to that created by the soldier for alerting on a few incorrect decisions in an otherwise good plan, the ITS additionally explains why the alternative action in the expert plan was better than their own decision. In the case of a bad plan by the soldier ITS will show them the correct plan and explain to the solder why the correct expert plan is correct.

Siemer and Angelides [6, 15] define gaming-simulation as "a sequential decision-making exercise... providing an artificial but realistic environment that enables players to experience the consequences of their decisions thorough immediate response." In 1994 the pair incorporated an intelligent tutor into the "Metal Box Business Simulation Game". The game and intelligent tutor combination is referred to as INTUITION (INtelligent TUITION). The game has up to four players taking on various management roles at the fictional Metal Box Company. They must solve various day-to-day issues involved in the sales and manufacturing of the products produced by Metal Box. The game begins with intelligent tutor assuming the player is a novice. Subsequent games within a round[2] allow the intelligent tutoring system to adjust to the players skill. The ITS promotes experienced players to higher levels of management and assign the lower management role to another player or simulate the role. If the role is simulated by the computer it will purposely make some bad decisions which must be corrected by the experienced player. Additionally the game simulates competing companies which the players must beat in the metal box market. The players are required to make decisions at time steps corresponding to financial quarters in the simulation. At each player's decision step the ITS calculates all possible solutions and identifies the optimal solution. The player's decision is compared against that of the ITS and if it is determined that the player does not understand one of the concepts in its expert module the player is given some remedial lesson.

[2]A round is defined by Siemer as a series of games in which the intelligent tutor is not reset.

2.2. Web-Based Tutors

Web-based tutoring systems provide any time, any where access to a customized learning environment. The majority of new intelligent tutoring systems are web-based because of the portability of web-based applications. None of the current web-based tutoring systems feature game play. Instead the interfaces usually present static text and images which make up a series of questions which the student must answer. Some systems include a talking head to go along with the text to increase the players comfort with the system. Correct answers cause the tutor to advance the student to a higher skill level while incorrect answers cause the tutor to provide hints in an attempt to coax the correct answer out of the student. This section examines several web-based tutoring systems that use this approach. While none of these are directly applicable to an immersive game-based tutor, this will show that while the tutoring agents have become smarter, little has changed in the presentation of material over the last ten years.

ELM-ART was the one of the first web-based tutoring systems [16]. Created in 1996 to bring the LISP tutoring system ELM-PE to the web, ELM-ART used a very simple interface consisting of hyperlinks and forms. The system presented material from a LISP textbook with colored hyperlinks to suggest which topics were within the students level of understanding. At the end of a section, the student was required to complete one or more short programs in LISP which required an understanding of the material presented. A correct implementation would advance the students recorded skill level which would in turn cause the section hyperlink colors to change. A wrong answer would cause the tutor to show an error message with an explanation of what was wrong with the code. Further incorrect answers would prompt more detailed error messages. ELM-ART was created just after the popularity of the Web began to rise, yet as will be shown in the following works, little has changed in web-based tutoring interface design.

Wijekumar and Meyer (2006) present a web-based intelligent tutor which teaches students about the Structure Strategy for reading comprehension. The system is called the Intelligent Tutoring system for Structure Strategy (ITSS). ITSS is based on a Flash application that uses a talking head and a simple window to hold the users attention (ITSS is targeted at middle school students). Students are presented with a text which they must read, then answer questions about that text. The tutor then identifies portions of the text according to the Structure Strategy to help a student correct an incorrect answer. [17]

Chipman, et al. (2007) present the third version of the AutoTutor intelligent tutoring system and its architecture in [18]. AutoTutor is a natural language tutor that allows students to respond in their own words. AutoTutor attempts to hold a conversation with the student to test them on a given topic. The current version of AutoTutor is based on a series of modules which can be replaced individually depending on the topic, language, etc. A state table is passed between each module in a specific order controlled by a central manager module called the Hub. The modules individually modify the state table so that at the end of the processing the system has analyzed the students response and generated a new question or hint. The state table is then sent to the client through a translator. This separates the interface from the tutor allowing a student to potentially access the same course through a complex application with animated talking heads or through a simple web page with forms. While this modular architecture is valuable for the type of material that AutoTutor is typically used to teach (math, science, and reading) it is unclear whether this would benefit a game-based tutor where the tutoring system must be able to monitor an immersive environment in real time.

The list of similar web-based tutoring systems goes on, two other new systems include SlideTutor (2006), a tutoring system that teaches medical students to identify skin diseases based on images of skin samples and [19] and COMET (2007) which is a distributed, multi-user tutoring system for medical students designed around the problem-based learning model. COMET allows students to discuss the problem together as well as receive hints from the tutoring system in a whiteboard area [20]. COMET is the closest existing system to the framework presented in Section 4 however the environment used, a text and image based system, makes the tutoring agent exactly like the other systems discussed.

2.3. Related Work Summary

This section discussed the implementation of many game and web based tutoring systems. None of the systems presented address the problems that will be encountered in the development of a distributed, multi-player game-based tutor which include cheating, match making, student and expert model creation and world synchronization. The game-based tutors presented in this section all assume that only a single human player is in the environment or as in the case of INTUITION uses a turn based game in which a students decision can be considered against a static game board, effectively making it a single player game as far as the tutor is concerned. The web-based tutors all use simple text based interfaces to teach typical classroom topics such as reading, math, and science. These are also turn based where one question equals one turn. These properties require the tutor to have knowledge of only a small, specific domain. Conversely a tutor which is used in an immersive environment in which the students can experiment and play in the world to complete a vague, high level assignment must have a much broader knowledge of what should happen and how the task should be accomplished.

On the other hand by looking at the group of these works as a whole we can identify common features, which should probably be included in any future game-base ITS. Gómez-Martín's work on JV^{2}M shows that an immersive tutoring system can go beyond text based after action reports using a sidekick type character that provides help during play. In a game such as SARGE where the players are cooperating in a team a helper agent may be incorporated as a teammate. Or instead of providing help directly via an in-game helper, the plan for SARGE is to make subtle changes to the environment to provide hints by directing the players attention to a specific area in the world or a specific control for the robot. However, in the case that the player still fails at a given task, Stottler has shown the utility of after action reports, sometimes the best way to get a point across is to be blunt. While the post game breakdown is definitely not a fun element of a game, the entertainment of after action reports could possibly be enhanced by having them given as part of the game narrative by a character.

The problems faced when developing a distributed, multi-player ITS are discussed further in Section 5. While problems such as match making can be handled using a Master Server as discussed in Section 4, solutions to problems such as maintaining a consistent world state given that multiple tutors make modifications to the world require further research.

3. SARGE: MOTIVATION AND BACKGROUND

This section discusses the motivation for distributed, game-based intelligent tutoring systems in the context of our work on SARGE. An interest in intelligent tutoring systems for complex environments grew out of observing novice Unmanned Aerial Vehicle (UAV) operators struggle and give up on their initial training in simulation. These operators were expected to master the UAV in simulation before operating the real UAV because it is safer and more cost effective than risking the destruction of a real UAV. The eureka idea was "It would be great if the simulator could determine the user's skill level and adjust the flight model or provide other assistance to prevent the user from becoming frustrated and giving up altogether." This notion, combined with previous work on improving performance of distributed teams of operators spawned the initial idea for SARGE and the framework presented in Section 4. Figure 1 shows one of the SARGE environments under construction with multiple robots being controlled by players, this particular environment is a model of the iSSRT testbed at the University of South Florida. Figure 2 shows a close up an iRobot Packbot model that is used in SARGE.

The goal for SARGE is to create a game-based tutoring system with the following properties, which are not combined in current intelligent tutoring systems:

1. Easily Distributable - The target market for SARGE is law enforcement and education, this required SARGE to be easy to install, use and obtain for a large number of people with varying degrees of knowledge of computers. To facilitate these needs, SARGE offers point and click installation and execution as well as a full graphical in-game menu system.

2. Multi-Player - Allow players to train at any time with other players over the internet. Our focus on team operations comes from prior research that shows that teams of operators tend to perform better on a task than a single operator [21]; and that in practice, several people are required to operate search and rescue and law enforcement robots effectively [22]. As such, a training game should provide the opportunity for the players to cooperate on a task as they do in the real world.

3. Multi-Platform - With the desire to reach as broad an audience as possible, multi-platform capabilities were a must. Windows and Macintosh computers make up all but a small fraction of the computer systems in our target market thus these are the platforms supported. Game consoles such as the XBox, Playstation or Wii would have been useful platforms as well, however the console makers generally do not cooperate with small developers.

4. Open Source - In the interest of collaboration and furthering development of SARGE it was decided that all source material, including art, would be provided on SourceForge.

5. Easy Content Creation - Given that all source materials are provided for SARGE, and to increase the pace of the initial development, an engine that has a simple, quick workflow was desired.

Initially SARGE was to be based on USARSim [23, 24], the open source robot simulation mod for UnrealTournament 2004. This decision was made because USARSim seems to be the most widely used robot simulator, is open source and is multi-platform. However after a short time we found that extending USARSim to include game features and new content to be quite slow due to the lack of documentation for the Unreal engine and buggy world editor. Additionally the physics engine was not performing as well as we had hoped. Running USARSim required a good knowledge of the command line - making distribution to non-computer-savvy-users difficult, and only source code was provided in the source download on the USARSim site on SourceForge, the art that makes up the worlds and robots was not available. This led to a search for a new game engine that would better meet the design criteria.

The Unity game engine, from Unity Technologies [25] was chosen after two months of searching for a new engine. During this process an evaluation of many other popular engines was conducted including the Torque engine from Garage Games, Ogre combined with OpenDynamicsEngine (ODE), OpenSceneGraph combined with ODE, CryEngine from Crytek, and the Source engine from Valve. Unity won out due to its simple workflow for content and code creation, complete API documentation, and the multiple methods of distribution of compiled games. Unity games can be compiled to run on Windows and Macintosh as both stand alone executables and as a webplayer that runs in a web browser, similar to Adobe Flash-based applications,which makes distribution to a wide audience simple. As well, Unity provides a built in networking library to facilitate the creation of multiplayer games.

SARGE has been in development for nearly one year and is available on SourceForge as a multi-robot simulator. The next step for development is to integrate several tutorials and a multi-player, multi-robot game scenario, as well as the intelligent tutoring system into the package.

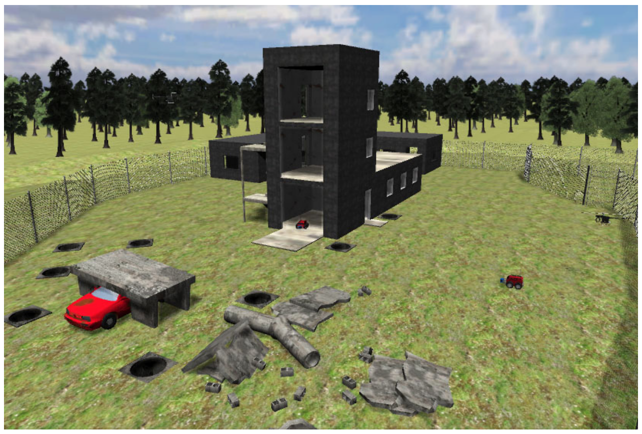

Figure 1. Screenshot From SARGE Showing Multiple Robots Being Controlled In The Environment

Figure 2. Screenshot From SARGE Showing The iRobot Packbot Model

4. DISTRIBUTED ITS FRAMEWORK

This section discusses the proposed framework and uses the SARGE implementation as an example. The framework is composed of three elements: an immersive game, a master server to store player information and handle multi-player collaboration, and the intelligent tutoring agent. Existing systems tend to combine two of the three components. For instance JV^{2}M combines an immersive game and an intelligent tutor, however there is no multi-user support. Another example is AutoTutor, which can store player information in a central repository, providing support for multiple users each with their own instance of a quiz (AutoTutor does not use an immersive game). The framework presented in this section combines all three components to create a more robust system that could prove to be a better tutor for more complex tasks that require multiple participants. Figure 3 shows a high level view of the framework components for SARGE.

4.1. The Game

The first element in the system is an immersive game which must include support for multiple players to allow teams of students to work together within the virtual environment. The multi-player aspect of the game should not only allow players to cooperate on a task but provide some means of competition between individual and teams of players. In the interest of increasing the desire to learn, an educational game should provide competition in the form that allows players with higher skill to receive a higher score than their lower skilled teammates and competitors. Figure 3 shows that an instance of a SARGE game feeds user input into both the tutoring agent and the game world, the game world state is then periodically sent over a network to other instances[3] within the game to maintain consistency across instances.

[3]An instance is a copy of the game running on a player's computer.

Vorderer, et. al [26] examines the role of competition in the enjoyment of video games by surveying game players about their game playing preferences. Vorderer's work looks at two types of competition, competition against the game itself and competition against other players which they call social competition. Looking at non-social competition, they found that players prefer games that provide many possibilities for the player to explore the virtual world combined with a strong need to meet some goal to advance. In their survey participants were asked to rate their expected enjoyment from playing the described game scenarios. The scenarios ranged from wandering around a virtual world with no tools and no goals to having lots of weapons and monsters approaching. Wandering with no goal received the lowest score while having lots of weapons and approaching monsters received the highest rating. The survey investigating social competition was similar, participants were asked questions about their competitive nature and to rate a list of games in terms of their competitive nature, however the results were only slightly statistically significant. The survey showed that players with a higher computer-game-specific competitive nature[4] were more likely to want to play competitive computer games. As well the survey showed that players who believe they can overcome any opposition in a computer game were more likely to choose competitive computer games.

If we combine these findings we see that social-competition itself does not necessarily bring fun to a game but is important to keep the interest of many players. Providing specific goals and multiple means of achieving said goals is the most important. Thus a game for a multi-player tutoring system must use goals that require and reward teamwork. Obviously the subject covered by the tutor will heavily influence the game content, however the idea of using competition to keep the player interested is a must. A simple example that is used in many commercial shooter games is a "capture the flag" scenario. The goal of capture the flag is to capture the opposing teams flag while preventing them from capturing your team's flag. This requires the teams to self organize into a defense and offense and the arenas are typically large and provide many paths for players to travel between their home base and the opposing teams base.

In SARGE the game will provide competition in both forms. Competition against the game is given in the form of tasks the user must accomplish. For instance one scenario under development for SARGE requires two play-

[4]Players which the survey determined were more competitive while playing computer games, but had only an average score for general social competitiveness.

ers operating a miniature unmanned aerial vehicle (MAV) to identify void spaces in a building collapse that may be searchable using a small ground vehicle (UGV). The players are scored based on both time and number of identified void spaces. Social competition while less important for the "fun" of the game is useful for comparing players' relative skill levels. Social competition is facilitated using an online score board and ranking system, commonly called a ladder. The SARGE ladder is under development and is based on Microsoft's X-Box Live system called TrueSkill [27].

4.2. The Master Server

The master game server provides several functions within this framework. Its primary function is to store player progress data and modules for the intelligent tutoring system. Player progress data includes scores for exams or trials, topics mastered, total hours played, improvement statistics, the student model for the ITS, etc. The ITS modules contain the expert model for the subjects covered by the game as well as any other information needed by the ITS. Additionally the master server facilitates connections between players. As is found in many commercial systems, the server should provide the game client with a means for the player to advertise a hosted game or join a game hosted by another player. Optionally the server could provide web-based access to player account information and publish high scores in an effort to foster competition.

The SARGE server has several components itself. The first component is based on the Master Server application that is distributed by Unity Technologies for use with their Unity game engine. This component keeps track of all game hosts available online. A game host is an instance of a game that is waiting for other players to connect. The Master Server allows players to choose a game host with favorable ping times or a game that is hosted by real-world teammates. The second component of the server is used by the intelligent tutoring agent. This component stores player login information along with scores, total time played, the ladder ranking, and skill levels for the various robot operating skills SARGE attempts to teach. The third component is a standard web server which allows players to view their performance statistics and high scores outside of the game.

4.3. The Intelligent Tutor

The intelligent tutoring system is composed of one or more software agents that monitor a users progress while working problems and provide suggestions when the user encounters problems, rearrange the curriculum, and adjust the parameters of the environment to provide the user with new material that they are mostly likely to successfully learn. Every intelligent tutoring system takes a different approach to implementation as this is heavily influenced by the type of material it is designed to teach. The SARGE tutoring system is designed to teach players to perform specific tasks using several robotic vehicles. The SARGE tutor is the first intelligent tutoring system to modify the environment in order to provide specific challenges which the player must learn to overcome. This requires the tutor to not only have knowledge of which skills the user has mastered and a method of measuring these skills which involve driving in various environments, targeting, and manipulating, but knowledge of how the environment must be modified to teach these skills. To do this the tutor must monitor the local player as well as the state of the world including the positions of other players so that the adjustments maintain consistency across instances.

4.4. Framework Summary

This section presented a high level framework for game-based intelligent tutoring systems. The framework is composed of three basic units: an immersive video game, an intelligent tutoring agent, and a master server. The game and tutoring system are combined into a single package which is referred to as an instance. Each instance communicates with the master server which stores a players progress in the game (the student model) and performance statistics. The master server is also used for match making, pairing players of similar skill.

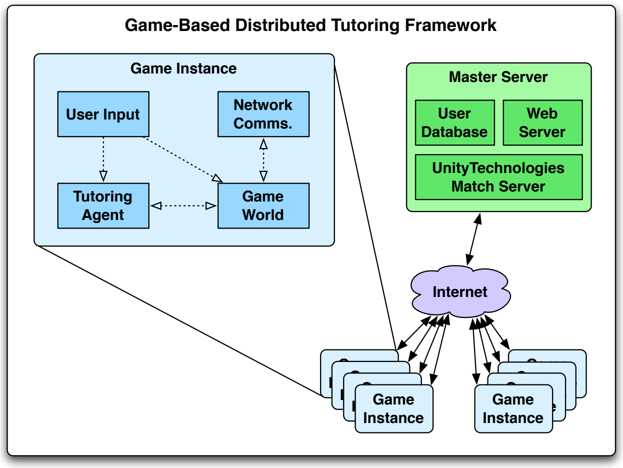

Figure 3. Overview of Distributed Tutoring Framework

5. DIFFICULTIES FACED BY A DISTRIBUTED ITS

This section discusses potential difficulties that will be faced when developing a distributed, game-based, intelligent tutoring system. Some of these difficulties are common in any online game, others are specific to tutoring systems. This section covers five specific issues: cheating, match making, “gaming the game”, defining the correct expert and student model for the ITS, and maintaining a consistent world state. There are bound to be implementation specific issues that are not covered in this section as they have not been encountered in the development of SARGE.

Two difficulties that will be faced that are common to any online game are cheating and match making. The problem of preventing cheating in online games is a topic of recent research [28, 29, 30, 31, 32, 33] because cheating has become a major problem. Cheating in online games is usually accomplished with a man-in-the-middle attack or client hacking. The most common cheating attacks can be prevented using encrypted traffic, which prevents the man-in-the-middle and snooping attacks. Client hacking is more difficult to prevent, and typically a second application is used to monitor the local memory space of the game for unauthorized changes. The game *World of Warcraft* uses a this type of monitoring system, if a player is found to be cheating they are kicked from the game and may be banned from playing again.

Match making is a task handled by the Master Server, players are matched with opponents and teammates that are at a similar skill level. This prevents new players from becoming frustrated with their lack of skill compared to expert players. However, there is no guarantee that there will be another player of similar skill available when a player is looking to play. This presents a problem, especially for educational and training games that may not have a large number of players. There are two solutions to this problem; one is to have a large number of platforms on which the game runs, thus hopefully increasing the number of available players; the other is to provide an agent which plays the game along with the player. The later method however is less desirable for a game that is supposed to train teams of players. A solution for classroom based training is to set aside time during training sessions in which the game is used.

Tutoring systems that have been used in elementary classrooms are often less effective than possible because the students “game the game”. This is the term for the students intentionally selecting incorrect answers in order to provoke an entertaining response from the tutoring system. The best solution to this is to create a game that is interesting enough to hold the player’s attention or to detect that this misuse is happening and take corrective action.

Creating the correct expert model for the ITS is the most important issue faced when developing any ITS. Without a proper expert model it is impossible to correctly judge the performance of the student. Creating an expert model involves a thorough analysis of the knowledge that is to be taught. If the knowledge covers a specific task, a task analysis must be performed to identify the key elements of the task. This knowledge must then be represented in such a way that a student’s performance can be judged against the expert model.

Creating a student model for a game based tutor is equally or more difficult than creating an expert model because this must be done on the fly. New users can be assumed to have no knowledge of a subject but the system must quickly identify the knowledge level of the student. This can be difficult in games that allow the user to freely explore a virtual world while accomplishing an assigned task. The point at which the system considers a topic as learned must be carefully considered. For example, in SARGE the student must learn to climb up and down stairs using a robot. Depending on the size of the robot this may involve changing the configuration of the vehicle to maintain stability. How many time must a student successfully climb the stairs, and how stable must the platform be while climbing for the system to consider the student a master of stair climbing? For a tutoring system that simply asks questions which the student must answer, the student model is easier to create since a fixed percentage of correct answers could be used to identify a topic as learned.

Maintaining a consistent world state is always an issue with networked games due to lag, however in the case of distributed, game-based tutoring systems there will be additional complications due to the individual tutoring agents making modifications to the environment. This in itself is not much of an issue, the environmental changes can be synchronized between all clients. The problem arises when two tutoring agents want to make conflicting changes. In this case the changes must be prioritized and managed by the host instance or by peer-to-peer communication between tutoring agents.

The five difficulties described in this section are only the major hurdles foreseen in the development of SARGE so far, there are bound to be others that have yet to be encountered in the development process. Additionally many of the challenges that involve the tutoring agents will be heavily influenced by the topic the system is designed to teach.

6. DISCUSSION AND SUMMARY

This paper presented a framework for a distributed, game-based intelligent tutoring system. This framework is derived from the Search and Rescue Game Environment (SARGE), a distributed, game-based tutoring system to train distributed teams of robot operators in the law enforcement and urban search and rescue fields. We discussed the motivation for the development of such a system as well as several major hurdles that must be overcome for distributed, game-based tutor to be successful. While many tutoring systems exist, none have yet attempted to examine the utility of immersive, multi-user environments. The majority of tutoring systems focus on a single user learning well understood, structured topics such as reading comprehension, math, science, or computer programming. SARGE is being developed to examine an unstructured topic, field robot operation, where robot operators typically learn on the job or in rare organized training events. We believe this group of target users and applications are the next step in the evolution of distributed, game-based, multi-user intelligent tutoring systems.

Future work on SARGE will include a validation study to verify the utility of game based training and the use of the intelligent tutoring system for robot operators. Additional vehicles and games will be created which broaden the scope of training scenarios including bomb disposal, search and rescue, and post-disaster building inspection.

SARGE is available on SourceForge at http://sarge.sourceforge.net. Updates are published regularly.

REFERENCES

[1] A. I. Wang, T. fsdahl, and O. K. Mrch-Storstein, "Lecture quiz - a mobile game concept for lectures," in *The 11th IASTED International Conference on Software Engineering and Application (SEA 2007)*, November 2007.

[2] J. Craighead, R. Murphy, J. Burke, and B. Goldiez, "A robot classification system for hri," in *Proceedings of the 2007 International Symposium on Collaborative Technologies and Systems (CTS)*, May 2007, p. 93.

[3] ——, "A survey of commercial and open source unmanned vehicle simulators," in *Proceedings of the 2007 International Conference on Robotics and Automation (ICRA)*, April 2007, p. 852.

[4] J. VOGEL, D. VOGEL, J. CANNON-BOWERS, C. BOWERS, K. MUSE, and M. WRIGHT, "COMPUTER GAMING AND INTERACTIVE SIMULATIONS FOR LEARNING: A META-ANALYSIS," *Journal of Educational Computing Research*, Vol. 34, No. 3, pp. 229–243, 2006.

[5] A. Mitchell and C. Savill-Smith, *The Use of Computer and Video Games for Learning: A Review of the Literature*. Learning and Skills Development Agency, 2004. [Online]. Available: http://www.lsda.org.uk/files/PDF/1529.pdf

[6] J. Siemer and M. C. Angelides, "Embedding an intelligent tutoring system in a business gaming-simulation environment," in *WSC '94: Proceedings of the 26th conference on Winter simulation*. San Diego, CA, USA: Society for Computer Simulation International, 1994, pp. 1399–1406.

[7] M. Gómez-Martín, P. Gómez-Martín, and P. González-Calero, "Game-Driven Intelligent Tutoring Systems," *Proceedings of the Third International Conference on Entertainment Computing (ICEC)*, pp. 108–113, 2003.

[8] A. L. Alexander, T. Brunyé, J. Sidman, and S. A. Weil, "From gaming to training: A review of studies on fidelity, immersion, presence, and buy-in and their effects on transfer in pc-based simulations and games," DARWARS, Tech. Rep., November 2005.

[9] R. Jensen, D. Chen, and M. Nolan, "Automatic Causal Explanation Analysis for Combined Arms Training AAR," *The Interservice/Industry Training, Simulation & Education Conference (I/ITSEC)*, Vol. 2005, 2005.

[10] M. A. Gómez-Martín, P. P. Gómez-Martín, and P. A. González-Calero, "Game based learning beyond simulations," in *Proceedings of the European Conference on Games-Based Learning (ECGBL'07)*, D. Remenyi, Ed. Academic Conferences Limited Reading, 2007, pp. 89–96.

[11] R. Stottler and L. Vinkavich, "TACTICAL ACTION OFFICER INTELLIGENT TUTORING SYSTEM (TAO ITS)," Vol. 2000. NTSA, 2000.

[12] D. Stottler, N. Harmon, and P. Michalak, "Transitioning An ITS Developed For Schoolhouse Use To The Fleet-TAO ITS, A Case Study," Vol. 2001. NTSA, 2001.

[13] R. Stottler and B. Pike, "An Embedded Training Solution: FBCB2/Tactical Decision Making Intelligent Tutoring System," Vol. 2002. NTSA, 2002.

[14] G. Burmester, D. Stottler, and J. Hart, "Embedded Training Intelligent Tutoring Systems (ITS) for the Future Combat Systems (FCS) Command and Control (C2) Vehicle," Vol. 2002. NTSA, 2002.

[15] J. Siemer and M. Angelides, "Evaluating intelligent tutoring with gaming-simulations," *Proceedings of the 27th conference on Winter simulation*, pp. 1376–1383, 1995.

[16] P. Brusilovsky, E. Schwarz, and G. Weber, "ELM-ART: An intelligent tutoring system on World Wide Web," *Intelligent Tutoring Systems. Lecture Notes in Computer Science*, Vol. 1086, pp. 12–14, 1996.

[17] K. Wijekumar and B. Meyer, "Design and Pilot of a Web-Based Intelligent Tutoring System to Improve Reading Comprehension in Middle School Students," *International Journal of Technology in Teaching and Learning*, Vol. 2, No. 1, 2006.

[18] P. Chipman, A. Olney, and A. Graesser, "The Autotutor 3 Architecture," *Web Information Systems and Technologies*, Vol. 1, pp. 323–332, 2007.

[19] R. Crowley and O. Medvedeva, "An intelligent tutoring system for visual classification problem solving," *Artificial Intelligence in Medicine*, Vol. 36, No. 1, pp. 85–117, 2006.

[20] S. Suebnukarn and P. Haddawy, “Comet: A collaborative tutoring system for medical problem-based learning,” *Intelligent Systems*, Vol. 22, No. 4, pp. 70–77, 2007.

[21] J. L. Burke, “Rsvp: An investigation of the effects of remote shared visual presence on team process and performance in urban search rescue teams,” Ph.D. dissertation, University of South Florida, 2006.

[22] K. Pratt, “Analysis of vtol mav use during rescue and recovery operations following hurricane katrina,” Master’s thesis, University of South Florida, 2007.

[23] “Usarsim,” http://usarsim.sourceforge.net/. [Online]. Available: http://usarsim.sourceforge.net/

[24] S. Carpin, M. Lewis, J. Wang, S. Balakirsky, and C. Scrapper, “Usarsim: a robot simulator for research and education,” in *Proceedings of the 2007 International Conference on Robotics and Automation*, April 2007, pp. 1400–1405.

[25] “Unity,” http://www.unity3d.com. [Online]. Available: http://www.unity3d.com

[26] P. Vorderer, T. Hartmann, and C. Klimmt, “Explaining the enjoyment of playing video games: the role of competition,” in *ICEC ’03: Proceedings of the second international conference on Entertainment computing*. Pittsburgh, PA, USA: Carnegie Mellon University, 2003, pp. 1–9.

[27] R. Herbrich and T. Graepel, “TrueSkill (TM): A Bayesian Skill Rating System,” Tech. Rep. MSR-TR-2006-80, Microsoft Research, Microsoft Corporation, Redmond, WA, Tech. Rep., 2006.

[28] B. D. Chen and M. Maheswaran, “A cheat controlled protocol for centralized online multiplayer games,” in *NetGames ’04: Proceedings of 3rd ACM SIGCOMM workshop on Network and system support for games*. New York, NY, USA: ACM, 2004, pp. 139–143.

[29] M. Bei Di Chen; Maheswaran, “A fair synchronization protocol with cheat proofing for decentralized online multiplayer games,” *Network Computing and Applications, 2004. (NCA 2004). Proceedings. Third IEEE International Symposium on*, pp. 372–375, 30 Aug.-1 Sept. 2004.

[30] B. D. Chen and M. Maheswaran, “A cheat controlled protocol for centralized online multiplayer games,” in *NetGames ’04: Proceedings of 3rd ACM SIGCOMM workshop on Network and system support for games*. New York, NY, USA: ACM, 2004, pp. 139–143.

[31] J. Yan, “Security design in online games,” *Computer Security Applications Conference, 2003. Proceedings. 19th Annual*, pp. 286–295, 8-12 Dec. 2003.

[32] C. GauthierDickey, D. Zappala, V. Lo, and J. Marr, “Low latency and cheat-proof event ordering for peer-to-peer games,” in *NOSSDAV ’04: Proceedings of the 14th international workshop on Network and operating systems support for digital audio and video*. New York, NY, USA: ACM, 2004, pp. 134–139.

[33] P. Kabus, W. W. Terpstra, M. Cilia, and A. P. Buchmann, “Addressing cheating in distributed mmogs,” in *NetGames ’05: Proceedings of 4th ACM SIGCOMM workshop on Network and system support for games*. New York, NY, USA: ACM, 2005, pp. 1–6.

This work was sponsored, in part, by the US Army Research Laboratory under Cooperative Agreement W911NF-06-2-0041. The views and conclusions contained in this document are those of the authors and should not be interpreted as representing the official policies, either expressed or implied, of the ARL or the US Government. The US Government is authorized to reproduce and distribute reprints for Government purposes notwithstanding any copyright notation hereon.