Toward Automatic Verification of Multiagent Systems for Training Simulations

Ning Wang[1], David V. Pynadath[2], and Stacy C. Marsella[2]

[1] Curious Lab LLC, Westchester, CA USA
ningwang@curiouslab.com
[2] USC Institute for Creative Technologies, Playa Vista, CA USA
{pynadath,marsella}@ict.usc.edu

Abstract. Advances in multiagent systems have led to their successful application in experiential training simulations, where students learn by interacting with agents who represent people, groups, structures, etc. These multiagent simulations must model the training scenario so that the students' success is correlated with the degree to which they follow the intended pedagogy. As these simulations increase in size and richness, it becomes harder to guarantee that the agents accurately encode the pedagogy. Testing with human subjects provides the most accurate feedback, but it can explore only a limited subspace of simulation paths. In this paper, we present a mechanism for using human data to verify the degree to which the simulation encodes the intended pedagogy. Starting with an analysis of data from a deployed multiagent training simulation, we then present an automated mechanism for using the human data to generate a distribution appropriate for sampling simulation paths. By generalizing from a small set of human data, the automated approach can systematically explore a much larger space of possible training paths and verify the degree to which a multiagent training simulation adheres to its intended pedagogy.

Keywords: multiagent training simulation, serious games

1 Introduction

Virtual worlds inhabited by autonomous agents are increasingly being used for experiential training and education (e.g., [1, 5, 8, 12, 14]). These virtual worlds provide an engaging environment in which students develop skills that can transfer to real-world tasks. To faithfully capture unpredictable real-world settings, simulations are populated by synthetic agents that ideally exhibit the same kind of complex behaviors that humans would exhibit [8, 14]. The creation of these environments raises considerable challenges. Foremost, a student's experience in the environment must be consistent with pedagogical goals and doctrine. Notably, success and failure in the environment must be aligned with the skills and knowledge that the system is designed to teach.

From an instructional perspective, the use of complex multiagent virtual environments raises several concerns. The central question is what is the student learning—is it consistent with training doctrine and will it lead to improved student's performance?

Negative training can arise in training environments due to discrepancies between simulation and the real world, as well as discrepancies between simulation and pedagogical goals. With inaccurate models, undesirable strategies may instead appear effective, leading one to become overconfident in their likelihood of success. Strategies may also be locally successful in the simulation but violate broader pedagogical and doctrinal concerns and lead to failure in larger, real-world contexts. For example, while eliminating political opposition may succeed in a local urban simulation, it may profoundly violate doctrine by leading to very negative consequences in a more global context.

As these simulations increase in size and richness, it becomes harder to verify (let alone guarantee) that they accurately encode the pedagogy. Human subject playtesting provides accurate data. But it explores only a limited subspace of simulation paths due to the high cost, in time and money. Although multiagent systems support automatic exploration of many more paths than is possible with real people, the enormous space of possible simulation paths in any nontrivial training simulation prohibits an exhaustive exploration of all contingencies.

However, many of these contingencies are very unlikely to ever be realized by a student. Specifically, a student is highly unlikely to perform actions randomly without regard to their effects. Consequently, presuming a student is sampling from a uniform distribution of all possible action sequences is a poor starting point for evaluating a complex multiagent based social simulation.

We present an automated mechanism that instead tests only those paths that we can expect from real human behavior. We first analyze a multiagent training simulation already deployed in classrooms. The result shows that, while the vast majority of students received appropriate feedback from the multiagent system, some students were able to succeed despite violating the pedagogy. Given this motivating example, we then present an automated mechanism for using the human data to generate a distribution appropriate for sampling simulation paths. Our combined mechanism can thus systematically explore a much larger space of possible training paths and verify the degree to which a multiagent simulation adheres to its intended pedagogy.

2 PsychSim and UrbanSim

While our methodology applies to many agent-based simulations, we use PsychSim as our example architecture [7, 11]. PsychSim is a social simulation tool for modeling a diverse set of entities (e.g., people, groups, structures), each with its own goals, private beliefs, and mental models about other entities. Each agent generates its beliefs and behavior by solving a partially observable Markov decision problem (POMDP) [4].

Multiple training simulations use PsychSim to generate behavior for the people, groups, and environment that students interact with to practice skills in a safe but realistic setting. The Tactical Language Training System helps students acquire communicative skills in foreign languages and cultures, where PsychSim agents represented villagers with whom the student develops rapport through conversation [13]. BiLAT uses PsychSim agents to engage students in bilateral negotiations in face-to-face meetings within a specific cultural context [5]. PsychSim agents also teach people to avoid risky behavior by simulating situations with pressure to engage in such behavior [6, 9].

In this paper, we focus on UrbanSim, a simulation-based training system that has been deployed to teach stabilization operations in post-conflict urban environments [8]. The student directs multiple military units to execute operations in the context of a fictional urban scenario. The student's goal is to make progress along multiple dimensions (e.g., economic, political, security), called Lines Of Effort (LOEs). PsychSim agents generate the behavior for people, groups, and structures, as well as computing the effects of the students' decisions on their states. In the scenario used in this paper, there were 88 such agents and 6 real-valued LOEs derived from their states. The students give commands to 11 units under their control, after which PsychSim agents observe the commands' effects, choose their own counteractions, and observe those counteractions' effects. This cycle repeats for 15 rounds, with the students getting feedback each round through their LOE scores and a partial view of the scenario state.

3 Evaluation of Pedagogy

Although UrbanSim has been successfully deployed in classroom, the question remains about how well the multiagent component correctly encodes the intended pedagogy. That pedagogy relates to the strategies in selecting commands to give to units based on current state of the world and phase of the mission. The goal of this training simulation is for the students' scores to be positively correlated with how well their action choices satisfy the intended pedagogy. UrbanSim gives students more than 3000 possible ways to deploy their 11 units for each of the 15 rounds, thus producing 10^{26} possible strategies. Given the impracticality of exhaustive enumeration of that strategy space using agent-based simulation, we instead used playtesting to explore only a subset.

3.1 Study Population

We recruited 58 participants (56 male, 2 female) from a US metropolitan area. 35% of them are between 18 and 35, 14% are between 36 and 45 and 16% are above 45 years of age. 11% of the participants have high school education or GED, 79% have some college education or college degree, 10% have some graduate education or a graduate degree. 21% of the participants spend 1-4 hours using computer daily, 79% spend more than 5 hours. 6% of the participants have not or only played video games several times in the past year, 9% play video games monthly, 28% play weekly and 58% play video games daily. 70% of the participants did not spend any time in active military duty.

3.2 Experiment Manipulation and Procedure

When UrbanSim is deployed in the classroom, students are first shown a usability video about basic operations in UrbanSim and then a pedagogy video on the desirable strategies to use in UrbanSim. In the pedagogy video, participants are taught to:

1. Consider a non-aggressive approach as an alternative to the oft-preferred aggressive approach. For example, attacking a group is an aggressive action while hosting a meeting with the local mayor is a non-aggressive action.

2. Direct units under command to carry out Clear actions first, then Hold actions and finally Build actions. Clear, Hold and Build are not types of actions, but effects of an action. The Clear effect of an action is to remove potential danger in an area. The Hold effect is to protect an area that has danger already removed. The Build effect is to help a secured area recover and prosper. Each action has a weighted effect on Clear, Hold and Build, e.g. advising a local mayor can affect both Hold and Build.
3. Plan ahead instead of being purely reactionary, e.g discouraging "Whack-a-Mole".

To encourage a greater diversity of strategies, one group of participants watches only the usability video (NoInstruction) and a second group watches both videos (WithInstruction). NoInstruction participants first fill out a consent form and a demographic background questionnaire, then watch the usability video. Next, they practice basic operations in UrbanSim for 15-20 minutes. After that, the participants interact with UrbanSim for 2 hours. Finally, they fill out the post-questionnaire. The procedure for the WithInstruction group is identical except that participants watch the pedagogy video following the usability video. There are 32 participants in the NoInstruction group and 26 participants in the WithInstruction group.

3.3 Measures

Demographic background questionnaire: asked questions about participant's age, education, video game experience, computer use experience and military background.

Post questionnaire: contains questions regarding the strategies that participants used in UrbanSim, perceived importance of people and groups in the scenario (e.g. police, tribes), perceived importance of the LOEs, self-efficacy of improving LOEs and their assessment of the effect of the training simulation actions on LOEs, e.g. the impact of patrolling a neighborhood on the economy, security, etc.

Training Simulation logs: captures the actions chosen by each participant for each unit for each turn, LOE scores before each turn was committed, final score of popular support, and final score for LOEs. We categorized participants' actions for each turn as whether they are Clear, Hold or Build actions and which LOEs they address.

3.4 Results

One participant's data was excluded from the analysis because the participant had no experience using a computer. A total of 57 participants' data are included in the analysis.

Encoding of Pedagogy The first aspect of the pedagogy is to consider non-aggressive action as an alternative to aggressive actions. So overall, we should observe participants performing more non-aggressive actions than aggressive actions.

Pedagogy 1: *Number of Non-aggressive Actions* $>$ *Number of Aggressive Actions*

The second aspect of the pedagogy is to follow a Clear $\rightarrow$ Hold $\rightarrow$ Build strategy. We summed up the number of actions carried out by the 11 units during the first third (turns 1 to 5), second third (turns 6 to 10) and last third (turns 11 to 15) of the game. We

then ranked the Clear, Hold and Build effect of all the actions in each third. If the effect on Clear is higher than Hold and Build, we then categorize that third as Clear focused. There are 171 thirds from 57 participants. Only 3% of the thirds are Hold focused, so we chose to ignore Hold and instead categorized only the Clear and Build effects of the actions of the first half of the game (turn 1 to 7) and second half (turn 8 to 15) of the game. Following this categorization, the pedagogy is still very clear: a student should secure an area through Clear actions before performing Build actions in that area.

Pedagogy 2: *Clear → Build*

Effect of Experiment Manipulation We conducted an ANOVA test on the percentage of non-aggressive actions participants took, and a CHI-Squared Goodness of Fit test on whether participants adhered to the two pedagogies, using the NoInstruction and WithInstruction groups as independent variables. Additionally, we compared the score on LOEs between two experiment groups using the ANOVA test. Results show that there was no significant difference between our two experiment groups on participants' use of non-aggressive (NA) actions ($N = 57$, $p = .45$), whether they followed the pedagogy ($N = 54$, $p = .53$) and their performance on LOEs ($N = 47$, $p = .78$).

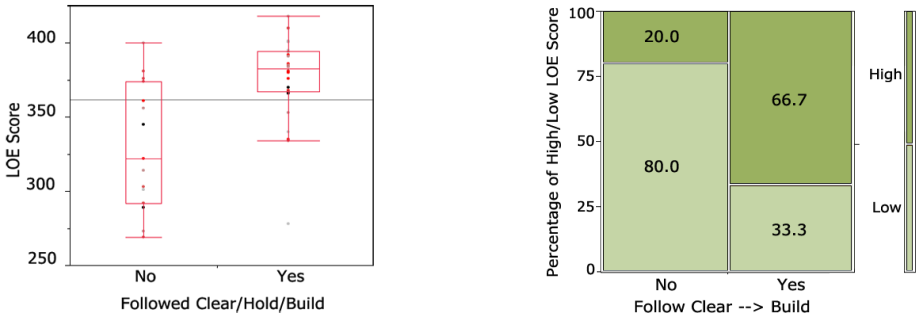
		No Instruction	With Instruction
NA Actions		0.788	0.802
Followed Pedagogy	Yes	20	14
	No	10	10
LOE Score		361.4	358.1

Table 1: Mean percentage of non-aggressive actions, number of participants following pedagogy, and mean LOE scores

Effect of Pedagogy Because there are no significant differences between the two experiment groups on the variables we are interested in, we combined the data from the two groups for the following analysis. Overall, we found that all the participants overwhelmingly adopted Pedagogy 1, choosing more non-aggressive actions (79%) than aggressive actions (21%). This means that we do not have data to compare scores between participants who followed Pedagogy 1 and those who did not. We will focus on Pedagogy 2 for the remainder of the analysis.

We then conducted an ANOVA test on performance on LOEs between participants who followed Pedagogy 2 and those who did not. Overall, there is a significant difference on performance on LOEs between participants who followed the pedagogy and those who did not. People who followed the intended pedagogy (Clear → Build) performed better on the LOE scores than those who did not ($M_{\textit{NotFollow}} = 330.4$, $M_{\textit{Follow}} = 377.1$, $N = 45$, $p < .001$).

Figure 1a shows that the distribution of LOE scores from participants who did not follow Pedagogy 2 is a lot more spread out compared to the distribution from those who followed the pedagogy. This implies that some participants who did not follow the pedagogy got high LOE scores. This issue is clearly illustrated in Figure 1b where we dichotomize the performance on LOE into High and Low. In Figure 1b, the left column represents the participants who did not follow the pedagogy, and the right column represents the ones who did. The lighter color represents low LOE scores and the darker color represents high LOE scores. The numbers on the graph represents the percentage of participants in that particular case, e.g. followed pedagogy and got a high LOE score.



(a) LOE scores

(b) Dichotomized LOEs scores

Fig. 1: Comparison of performance on (a) LOE scores and (b) dichotomized scores between participants who followed Pedagogy 2 and those who did not.

We can see that a significant percentage of participants achieved high LOE scores despite not following the intended pedagogy (Clear → Build). In fact, this group of participants all followed the Build → Build strategy, which worked just as well as the Clear → Build strategy. This could be problematic in a training simulation because following the Build → Build strategy would have severe consequences in the real world, e.g. early builds will be destroyed if an area was not secured first through Clear/Hold.

Figure 1b also shows a region of participants who followed the pedagogy but received low scores (the lower right). While this is also indicative of an error, our procedure for identifying Clear → Build strategies is subject to false positives, in that strategies that we identified as following Clear → Build may still be violating Pedagogy 2. For example, while a student's Build actions may be restricted to only the second half of the game, they may have been executed in regions that had not been previously cleared. Our purely temporal classification would not detect such an error. On the other hand, a strategy that does not satisfy Clear → Build in our crude classification definitely violates Pedagogy 2, so the upper left region of Figure 1b (and the rewarded Build → Build strategy within) corresponds to clearly undesirable outcomes.

4 Simulation-Based Verification of Pedagogy

Section 3's experimental results demonstrate that the simulation generally encourages the correct behavior, thanks to the rounds of playtesting and model editing that had already occurred. However, the results also identified one pedagogically incorrect strategy (namely, Build → Build) that was also rewarded by the simulation. The encouragement of such a strategy suggests the need for changes to the underlying scenario model to bring the simulation more in line with the intended pedagogy. Unfortunately, it is prohibitively costly to playtest after each such change, making it impossible to use human subjects in a tight iterative refinement cycle. Moreover, the playtesting results from Section 3 represent only 57 possible simulation paths. However, the training scenario

provides the student with over 10^{26} possible simulation paths in trying to capture the complexity of real-world urban stabilization. Thus, even if playtesting were feasible, it could explore only an infinitesimal portion of the possible space.

On the other hand, a student actively trying to succeed in the simulation would never try many of the 10^{26} possible simulation paths. For example, a student would not deliberately choose to devote resources to repair a structure that was already operating at full capacity. Although it is possible that a student might do so in error, the likelihood of such errors is so low that we may safely ignore such a possibility in our verification process. Of greater concern are errors like Build $\rightarrow$ Build that show up in multiple cases even within the relatively small data set of Section 3. Our goal in this section is to use this data acquired as the basis for an automatic method for exploring simulation paths that is sensitive to the likelihood of behaviors by real students.

4.1 Markov Chain Monte Carlo Simulation

Our proposed automatic method generates a plot like Figure 1b by randomly generating paths through the training simulation that give us a final score and that allow us to determine whether they followed the pedagogy. Markov chain Monte Carlo (MCMC) simulation provides such a method, in that we can translate a distribution over student actions into simulation path samples [2, 3]. To apply MCMC to a training simulation, we must first represent the evolving state (both observed by the student and hidden inside the system) as a Markov chain, X_t . In the multiagent system underlying our simulation, the complete state (S from the POMDP) of the UrbanSim scenario is already represented as a set of 1452 features (e.g., a structure's capacity), each a real-valued number from -1 to 1 (e.g., 1 means that the structure is functioning at 100% of capacity). While we wish to capture the evolution of the overall simulation state, the states in the Markov chain must represent the student's decision-making inputs as well. The student sees very little of the 1452 features and is instead informed mainly by the LOE scores (which in this scenario, are derived deterministically from the simulation state). We thus augment the simulation state with the observable LOE scores to capture both the state of the simulation and the factors that influence the student's choice of action. In addition to capturing all of the relevant factors, the Markov chain representation must also capture the transition from the current state to the next as a function of the student's action, but independent of prior state history. However, our survey data identified that students often reacted to *changes* in their score, not just the current value. Therefore, to account for this factor and to preserve the Markovian property, we add the latest change in LOE score to the state as well. In summary, the set of possible states for our Markov chain is defined over the possible simulation states, observable values, and changes in reward values: $X = S \times \Omega \times \Delta R$.

4.2 Sampling Distribution

Given this representation of the current state, we must represent the Markovian state transitions in terms of the distribution over possible student's actions and their effects. The underlying simulation dynamics (T) can generate the effects of actions, the observation function (O) can generate what the student sees of that state, and the scoring

function (R) can generate the changes in rewards. However, all three functions require the student's action choices as input. Therefore, the only new component we need for the dynamics of our Markov chain ($\Pr(X_t|X_{t-1})$) is the students' decision-making. The current state has sufficient information to motivate different students' choices, which we can thus model as a function, $\pi : \Omega \times \Delta R \rightarrow \Pi(A)$, that maps from observation and change in reward to a probability distribution over action choices.

For complex training scenarios, students may have too many possible choices for limited data to generate a meaningful distribution over their decision-making. For example, in the UrbanSim scenario, there are more than 3000 possible actions, so we would require a prohibitively large data set to learn a distribution over the original fine-grained action space, $|A| > 3000$. Instead, we propose clustering the original actions based on their effect on the game scores (e.g., the 6 different LOEs). For a given state, we can sum the cumulative effect of the student's actions on the game score (e.g., the effect of all 11 subordinates' actions on the 6 LOEs).

We can now examine the playtesting data in these terms to compute a frequency count of actions chosen as a function of possible score changes. Table 2 shows the expected rate of different types of actions as a function of changes in one of the score dimensions (labeled *LOE 2*). The probability distribution in this table is based on data collected from 57 participants. Students are roughly half as likely to choose an action to increase LOE 2 if there has been no change in its value, and that actions LOE 1 are more common regardless. Note that the numbers in Table 2 are obviously highly domain-dependent, but the method of acquiring them generalizes quite easily. By clustering the actions according to the scores they immediately increase, one can automatically analyze the logs to compute such frequency counts in a straightforward manner.

Action	Decrease	Increase	No Change
LOE 1	0.36	0.32	0.38
LOE 2	0.25	0.22	0.12
LOE 3	0.00	0.01	0.01
LOE 4	0.13	0.10	0.10
LOE 5	0.05	0.09	0.08
LOE 6	0.19	0.22	0.25

Table 2: Expected probability of action types given most recent change in LOE 2

4.3 Simulation Paths

Now that we have the abstract strategy, $\hat{\pi}$, for the students' actions, we can compute the dynamics of our Markov chain:

$$\Pr(X_t = \langle s_t, \omega_t, \Delta r_t \rangle \,|X_{t-1}) = \langle s_{t-1}, \omega_{t-1}, \Delta r_{t-1} \rangle$$

$$= \sum_{\hat{a} \in \hat{A}} \hat{\pi}(\omega_{t-1}, \Delta r_{t-1}, \hat{a}) \hat{T}(s_{t-1}, \hat{a}, s_t) O(s_t, \hat{a}, \omega_t) \Pr(\Delta r_t = R(s_t, \hat{a}) - R(\omega_{t-1}))$$

where we assume that the previous reward is extractable from the previous observation, ω_{t-1}. For training simulations where the students do *not* observe their scores along the way, we can simply explicitly encode the score as an additional component of our Markov chain state, X. The final missing piece is the abstract transition probability, $\hat{T}$, over our abstract actions, $\hat{A}$. The underlying simulation provides the fine-grained transition function, T, which we will use to derive its abstract counterpart. In particular,

for each abstract action, $\hat{a}$, we will define its effect as a uniform distribution over its possible corresponding fine-grained actions, a:

$$\hat{T}(s_{t-1}, \hat{a}, s_t) = \sum_{a|C(a)=\hat{a}} T(s_{t-1}, a, s_t) / \left|\{a|C(a) = \hat{a}\}\right|$$

We can now run the simulation engine and substitute actions sampled according to Section 4.2 instead of the student actions. Each such run requires only 4 minutes (as opposed to the hour required by the typical human subject), and we were able to generate 316 paths in 21 hours of computation time. The end result of each path is a run of the exact same form as used in playtesting and, thus, amenable to the evaluation procedure of Section 3. Thus, we determined whether the generated actions satisfied the intended pedagogy, and we extracted the score achieved by those actions. Finally, we generated the graph in Figure 2 (of exactly the same form as Figure 1b) to identify the degree to which the pedagogy is satisfied. Of the paths that violated Pedagogy 2, most received an appropriately low score, but the simulation identified 143 paths where an incorrect strategy received a high score, far exceeding the incorrect paths found among the 57 student paths in Figure 1b. Given that the simulation was able to generate Figure 2 overnight, as opposed to the weeks required to schedule the human subjects for Figure 1b, our automated exploration method has greatly accelerated our ability to verify the simulation underlying our training system.

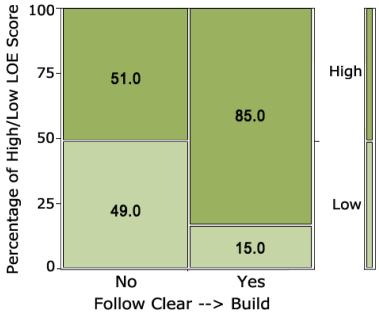


Fig. 2: Results from simulation-based verification

5 Discussion

The methodology presented in this paper provides a mechanism for automatic verification of an agent-based training simulation using limited human user data. The true test of a training simulation is in a thorough pedagogical evaluation of student learning when using the system, and our proposed methodology is in no way a replacement of such an evaluation. Our methodology instead seeks to give the simulation designer feedback during the authoring process. In particular, a graph like Figure 1b identifies paths through the simulation that violate the intended pedagogy, directing the designer to possible modeling errors. Section 4's automatic method for generating such graphs can then give the simulation designer similar feedback for the refined models, without requiring further playtesting. Furthermore, the systematic exploration of a larger space of possible student strategies can give the simulation designer greater confidence in the agent models before proceeding to the overall pedagogical evaluation and deployment.

Going beyond the reactive strategies of our MCMC approach to modeling the student's behavior, there is the potential to use PsychSim's POMDP-based behavior-generation mechanism to provide more sophisticated models of student moves. In the post-questionnaire, we collected information about how students ranked the various LOEs

in priority and how they thought different actions affected those LOEs. The former gives us insight into how the students' subjective reward function deviated from the "rational" student's. The latter gives us insight into how the students' model of the simulation dynamics deviated from the correct transition probability function, T . Thus, we can potentially learn PsychSim models for different students and use these models to generate more deliberative strategies than the reactive strategies of our MCMC approach.

Finally, our verification methodology can be a key component to facilitating the overall authoring process for training simulations. This paper presents a novel method for automatically finding simulation paths that are inconsistent with intended pedagogy. Given the output of our method, we can then use existing algorithms [10] to help automate the modification of the simulation to bring it more in line with that intended pedagogy. Thus, the methodology and algorithms presented in this paper represent a critical step toward greatly reducing the burden of authoring agent-based training simulations while simultaneously improving their pedagogical fidelity.

References

1. R. Calder, J. Smith, A. Courtemanche, J. Mar, and A. Ceranowicz. ModSAF behavior simulation and control. In *Proceedings of the Conference on Computer-Generated Forces and Behavioral Representation*, pages 347–356, 1993.
2. W. Gilks, S. Richardson, and D. Spiegelhalter, editors. *Markov chain Monte Carlo in practice*. Chapman and Hall, London, 1996.
3. W. Hastings. Monte Carlo sampling methods using Markov chains and their applications. *Biometrika*, 57(1):97–109, 1970.
4. L. P. Kaelbling, M. L. Littman, and A. R. Cassandra. Planning and acting in partially observable stochastic domains. *Artificial Intelligence*, 101:99–134, 1998.
5. J. M. Kim, J. Randall W. Hill, P. J. Durlach, H. C. Lane, E. Forbell, M. Core, S. Marsella, D. Pynadath, and J. Hart. BiLAT: A game-based environment for practicing negotiation in a cultural context. *IJAIED*, 19(3):289–308, 2009.
6. J. Klatt, S. Marsella, and N. Krämer. Negotiations in the context of AIDS prevention: an agent-based model using theory of mind. In *IVA*, 2011.
7. S. C. Marsella, D. V. Pynadath, and S. J. Read. PsychSim: Agent-based modeling of social interactions and influence. In *ICCM*, pages 243–248, 2004.
8. R. McAlinden, A. Gordon, H. C. Lane, and D. Pynadath. UrbanSim: A game-based simulation for counterinsurgency and stability-focused operations. In *AIED Workshop on Intelligent Educational Games*, 2009.
9. L. C. Miller, S. Marsella, T. Dey, P. R. Appleby, J. L. Christensen, J. Klatt, and S. J. Read. Socially optimized learning in virtual environments (SOLVE). In *ICIDS*, 2011.
10. D. V. Pynadath and S. C. Marsella. Fitting and compilation of multiagent models through piecewise linear functions. In *AAMAS*, pages 1197–1204, 2004.
11. D. V. Pynadath and S. C. Marsella. PsychSim: Modeling theory of mind with decision-theoretic agents. In *IJCAI*, pages 1181–1186, 2005.
12. J. Rickel and W. L. Johnson. Integrating pedagogical capabilities in a virtual environment agent. In *Agents*, pages 30–38. ACM Press, 1997.
13. M. Si, S. C. Marsella, and D. V. Pynadath. THESPIAN: An architecture for interactive pedagogical drama. In *AIED*, pages 595–602, 2005.
14. M. Tambe, W. L. Johnson, R. M. Jones, F. Koss, J. E. Laird, P. S. Rosenbloom, and K. Schwamb. intelligent agents for interactive simulation environments. *AI Magazine*, 16:15–39, 1995.