Dealing with dynamic changes in time critical decision-making for MOUT simulations

By Shang-Ping Ting* and Suiping Zhou†

Generating realistic behaviors for the non-player characters (also known as bots) is an important task for Military Operations on Urban Terrain (MOUT) simulations. Most of the current bots reasoning models are based on complete rational analysis of all alternatives. However, such models may not be adequate in time critical and uncertain situations. In such situations, due to time constraints and incomplete information, humans rely mainly on experience rather than some structured analysis of the given situation to make decisions. In our previous work, we have developed SNAP, a time critical decision-making framework which aims to imitate human decision-making processes for MOUT simulations. A major limitation that we found about SNAP is its difficulty in dealing with dynamic changes in MOUT simulations. In this paper, we propose to address this problem by incorporating a novel feature of expectations into SNAP. The effectiveness of this new feature is assessed with Twilight City, a virtual environment for MOUT simulations. Experimental results show that our expectation model works well in dealing with dynamic changes in MOUT.

Copyright © 2009 John Wiley & Sons, Ltd.

Received: 21 March 2009; Accepted: 24 March 2009

KEY WORDS: time critical decision-making; MOUT simulations; expectations; dynamic environments

Introduction

If you know your enemies and know yourself, you will fight without danger in battles. If you only know yourself, but not your enemy, you may win or may lose. If you know neither yourself nor your enemy, you will always endanger yourself.

The ancient statement made by Sun Tzu¹ is still valid in modern warfare. To increase the chances of survival, it is crucial for soldiers to achieve clear understanding of their own tactics and the adversarial threats. This understanding is especially important for Military Operations on Urban Terrain (MOUT).²,³ Modern armies may suffer substantial casualties in MOUT warfare. The limitation of urban terrains means that soldiers are expected to function in small teams without the support of air-power or tanks. To survive, soldiers rely on small squad tactics and individual situational awareness. Going into MOUT operations without proper preparation could be very dangerous to the soldiers.⁴,⁵ Soldiers often need to recognize the current situation with incomplete information and to make rapid decisions under time pressure, uncertainty, high stakes, and changing conditions.⁶–⁹

Through proper modeling of various tactical situations, MOUT simulations provide an important and cheaper way for the soldiers to get familiar with various tactical situations in MOUT. In MOUT simulations, the virtual urban environments are populated by various characters. While some of these characters are controlled by human players (i.e., the trainees), most of them are non-player characters (or bots) which are usually represented by AI-driven agents. For an MOUT simulation to be effective, it is important for these bots to demonstrate some human-like tactical behaviors. Although different approaches may be used to this end, we believe that human-like behaviors

*Correspondence to: S.-P. Ting, School of Computer Engineering Nanyang Technological University, Singapore 639798.
E-mail: ting0021@ntu.edu.sg
†Member, IEEE.
should be generated by human-like decision-making processes.

In our previous work,\textsuperscript{10} we have proposed a time-critical decision-making framework called SNAP that aims to imitate human decision-making processes in MOUT situations. As pointed out by Klein in [11], humans rely much more on experiences rather than deliberative rational analysis of possible alternatives to make time-critical decisions. Soldiers fighting in MOUT often need to pick up some key situation cues and retrieve past experiences to make quick decisions. Therefore, in our previous implementation of SNAP, we have introduced two novel features including case-based reasoning (CBR) and thin-slicing. CBR is used to make quick decisions by comparing the current situation with past experience cases. CBR is the process of solving new problems based on the solutions of similar past problems. It has been argued that CBR is not only a powerful method for machine reasoning, but also a pervasive way for humans to make decisions in everyday situations.\textsuperscript{12} Thin-slicing is used to model human’s ability to quickly form up situation awareness under uncertain and complex situations using key cues from partial information. In his book Blink,\textsuperscript{13} Gladwell discussed how humans are able to achieve rapid situation awareness using only a narrow slice of information. This technique is called thin-slicing. Using thin-slicing, humans can focus on the most significant information (or key cues) about a situation, and make judgment quickly.

Our previous experiments show that with CBR and thin-slicing, SNAP is able to generate quick decisions in various tactical situations with incomplete information. However, we have also observed that it is unable to deal with the dynamic changes in the environment. Real MOUT situations are highly dynamic, thus the decision-making process of a soldier is never a one-step process. A solution may become invalid as the situation evolves. In this case, a soldier may re-assess the current situation and attempt to find new solutions. We believe that expectation plays an important role in this regard.

In MOUT warfare, by matching the current situation cues with his/her experience cases, a human soldier may have a quick assessment of the current situation and find a solution to deal with the current situation. In the meantime, the soldier will also form up expectations on the likely future events. As the situation evolves, if the expectations are not fulfilled or violated, the soldier may infer that his/her initial solution may be invalid due to dynamic situational changes. Then the soldier may abort the previous solution and find a new solution. In our previous implementation, SNAP does not provide future projections (i.e., expectation) on the evolving situation. Without a mechanism to assess their solutions during execution, the bots can only know the validity of the solution after the execution of the complete solution. This could result in the bots blindly executing the initial solutions even though they are no longer effective.

Therefore, we propose to handle the dynamic situational changes with expectations. In this paper, we consider expectations as some events that are likely to occur in the near future. Bots continuously observe the ongoing situation after a solution is selected. If the observations are consistent with expectations, then the solution will be enhanced. Observations that do not match expectations may lead to the invalidation of the solution. In this case, the bots re-assess the situation to find new solutions and form up new expectations. The feature of expectations is designed to work in tandem with the existing features of CBR and thin-slicing. Thin-slicing will pick up key cues from partial information and pass these cues into the CBR process. CBR will match the cues with past experiences to retrieve the relevant experience cases. Expectations are formed up along with the past experiences. The bots will use these expectations to assess the validity of the solutions in the face of rapidly changing situations.

Now let us use a simple example to illustrate how these features of the SNAP framework could help a bot to make decisions in time-critical and dynamic MOUT situations. Figure 1 shows a sniper bot aiming at a soldier bot.

The soldier bot needs to make a rapid decision to counter the sniper. Using the CBR process, it will retrieve past experiences that are similar to the current situation and reuse the previous solutions. If needed (i.e., the current situation is new), the proposed solution can be
revised to adapt to the current situation. Once the revised solution had been successfully adapted to the new situation, it will be retained in the memory for future engagement. The matching of the current situation with past experiences is done by picking up key cues from the partial information gathered within the limited time, i.e., by thin-slicing. These cues may include the location of the sniper, whether the sniper is in open area or hiding behind a wall, the weapon being used and strength of the fire, etc.

It is likely that the enemy sniper is attacking the soldier from a building. According to doctrine, a human soldier will attempt to take cover and form up with his squad before performing a counter sniper operation to neutralize the enemy threat. During the counter sniper operation, the soldier is expected to overpower the enemy through speed and superior fire power. If there are more enemies or stronger firepower than the soldier expects, then the initial solution to perform a counter sniper operation may become invalid. Figure 2 shows the simulation time line of a group of soldier bots being attacked by a sniper and their subsequent actions.

The soldier bots select the solution to perform a counter sniper operation at T1. With this in mind, they hold a set of expectations on the enemy behaviour, e.g., the enemies are adopting a hit-and-run tactics and will not be waiting at the sniper location. They do not expect to face heavy enemy fires during the counter sniper operation. At T2, an unexpected threat of Close Combat Fire is observed. This new threat violates the current expectation and therefore invalidates the counter sniper solution. The violation of the expectation implies that there might be an ambush waiting for the counter sniper squad and a new solution will need to be generated.

The remainder of this paper is organized as follows. The next section discusses the related work. The Section, Expectation Modeling in SNAP, describes the details of our implementation. Experiments and Analysis covers the results and impact of expectations on SNAP. The last section concludes our current work and provides some insights into our future direction.

**Related Work**

As an interesting and challenging problem, how to generate human-like tactical decisions for the bots in time critical and stressful situations has attracted many researchers and developers in the military training and game AI communities. John Laird and his group has done remarkable work in capturing and encoding human variability in human behavior models for military simulations.6,14 Their models mainly focus on the deliberative reasoning processes, which we feel are not
adequate to model human’s decision-making process in time critical situations.

The recognition-primed decision (RPD) model\cite{11} proposed by Gary Klein emphasizes the role of experiences in human’s decision-making process in various time critical situations. The model also suggests that humans make quick decisions in such situations by matching the current situation with past experiences and selecting solutions to similar situations. Similar ideas were also advocated in \cite{10, 15, 16}.

Gladwell observed that humans often perform thin-slicing, that is, they reply on some key cues of a complex and uncertain situation to achieve rapid situation awareness and to make intuitive decisions.\cite{13} He also pointed out that collecting more information may not help to make the intuitive decisions more accurate. Consistent with this idea, based on numerous case studies, Evans emphasized that military commanders must not be overloaded with information during time critical situations.\cite{17} We believe that thin-slicing is an important aspect of human decision-making in time critical and complex situations. Thus, incorporating this aspect into the modeling of human decision-making process is critical for the realism of bot behaviors in MOUT simulations.

To improve the intelligence of his bots, Laird added anticipation to Quakebots.\cite{18} His bot is distinguished by its ability to build its own map as it explores a level, use a wide variety of tactics based on its internal map, and in some cases, anticipate its opponent’s actions. Laird found that instead of trying to encode behaviors for each specific situation, a better idea is to attempt to add a general capability for anticipating an opponent’s actions. From an AI perspective, anticipation is a form of planning; a topic that researchers in AI have studied for 40 years. The power of chess and checkers programs comes directly from their ability to anticipate their opponent’s responses to their own moves. Anticipation for bots in first-person shooters (FPS) has a few twists that differentiate it from the standard AI techniques such as alpha-beta search.

In terms of challenges and end results, our work on modeling expectations is similar to Laird’s efforts to add anticipation to his bots. However, while his goal is to produce intelligent bots to win FPS games, our goals are to simulate the end-result and procedural realism of a human soldier in MOUT warfare.

In terms of how to represent human’s experiences in our behavior modeling framework, we also borrow ideas from some social and cognitive psychology studies. It has been observed that human beings possess categorical scripts in their memory to interpret and predict the world situations, and new information is processed according to how it fits into these scripts called schema.\cite{19} Since these schema are context specific, they are dependent on an individual’s experience with and exposure to a subject area rather than simply some raw intelligence.\cite{20, 21}


### Expectation Modeling in SNAP

In this section, we describe our design of SNAP, a time critical decision-making framework for MOUT simulations. We will first briefly summarize the major components of SNAP, then focus on the expectation modeling in SNAP.

As shown in Figure 3, the SNAP framework consists of five main components: Goal, Observe, Situation Awareness, Experience Repository, and Action. The Goal component defines the goals of the bots. It will determine which information is relevant for situation awareness. The Observe component collects key cues of the current situation and sends the collected information to the Situation Awareness component. The components inside the dash line box forms the CBR process. By matching the observed cues with the experience cases in the Experience Repository, a solution will be proposed for execution by the Action component. The Observe component monitors the situation and update the Action component as the proposed solution is being executed. Depending on its result, the proposed solution may be revised and the updated solution will be retained in the Experience Repository as a new experience case for future use.

### Adding Expectations into the Case-Based Reasoning Process

The CBR process consists of four steps: retrieve, reuse, revise, and retain. Given the goal and observations from the environment, to form up situation awareness, a bot first needs to retrieve past experiences from its Experience Repository. In our implementation, experience cases are represented by \((\text{threat, solution, expectation})\) sets as shown in Figure 4. The threat in an experience case is represented by the precondition cues which act like a pattern (or schema) for the bot to recognize the threat. In SNAP, these precondition cues mainly consist of the levels of some qualitative descriptions about the situation, e.g., whether the enemy is near or far away, whether the strength of a sniper’s fire is high or low, etc. Qualitative measures are used here simply because humans reply more on qualitative rather than quantitative measures to
make sense of a situation. Situation awareness is then formed by matching the evidence cues of the current situation with the precondition cues.

The key cues picked up by the bots are used to form an evidence set consisting of evidence cues. When the evidence cues match the precondition cues of an experience case, the corresponding solution of the case will be reused by the Action component. Each solution is represented by a sequence of actions. Each action is attached with some post-conditions which represent the expected results of the actions. The Observe component monitors the post-conditions while the actions of a solution are executed. If the post-conditions of an action are not met, the next action will not be executed and the

Figure 3. SNAP: time critical decision-making framework.

Figure 4. An experience case.
proposed solution will be considered as unsuccessful. In our current implementation, the revision is done by human experts. These experts will analyze the situation and provide a new solution for the bot. Subsequently, the updated solution will be retained in the Experience Repository as a new experience case for future use.

When a threat is identified, expectations will tell the bots what is likely to happen next. As seen in Figure 4, each experience case contains an expectation frame. This expectation frame consists of a group of events associated with the identified threat. Based on past experiences, the bots expects these events to occur in near future. Each event in an expectation frame is represented by its corresponding cue set (called Event Set in this paper). The event set contains key cues that the bots should monitor in order to confirm their expectations. In our current implementation, these expectations are individual and separate. That is, there is no temporal relations among them. In the future, it may be necessary to introduce temporal relations among expectations to model things like “first, I expect A to occur, then B to occur.”

Now let us use a simple example to further illustrate how expectations work together with other modules of SNAP. Suppose that Ted, a soldier bot, is patrolling a street. He is able to identify a sniper-assault threat by observing a militant bot firing a sniper gun. In this case, spotting the militant bot and the sniper fire are the two precondition cues. Thus, the precondition set for sniper-assault threat is:

\[
\text{Bot(Militant, Snipergun) \land Weapon\_Fire(Snipergun)}
\]

Now suppose that Ted’s current evidence set is:

\[
\text{Bot(A, Militant, Male, Snipergun) \land Weapon\_Fire(Snipergun)}
\]

The evidence set states that Ted had spotted a militant bot and also observed sniper fire. As the evidence cues match the precondition cues of the sniper-assault threat, the experience case for the sniper-assault threat is retrieved from the Experience Repository. The attached solution will be reused by the Action component, i.e., the sequence of actions contained in the solution will be executed.

During the execution of the solution, the expectation frame is monitored. This is done by monitoring the cues from the event sets of the expectation frame. In the Counter Sniper operation, two main events will be monitored. In Event Set A, a soldier bot expects the enemy sniper to attack any soldier bots that moves into the sniping zone. Thus, if a soldier bot is not attacked when moving into the sniping zone, this may imply that the sniper had fled away from his previous location and the Counter Sniper solution will become invalid. In Event Set B, a soldier bot expects to move undetected during his approaching movement. If the soldier bot is detected by the sniper, his approaching movement will be compromised and the enemy sniper may flee from its location too. With such possible events from its expectation, Ted monitors key cues such as “bots not being attacked while in the sniping zone’ and ‘bots being attacked during movement.”

Solution Validation with Expectations

The Observe component monitors the evolving situation and forms the observation set. The observation set is made up of a set of key cues picked up by the bot. These key cues are used for matching with the cues in the event sets that form up the expectations. The successful matching of the cues means that the expected events had occurred and the expectation is therefore confirmed. For example, suppose that Ted is currently approaching the sniper in the Counter Sniper operation. The current expectation of Ted contains Event Set A and Event Set B as we discussed before. The Event Set B of Ted can be expressed as:

\[
\neg \text{Weapon\_Fire(All)} \land \neg \text{Vehicle(Militant)} \land \neg \text{Bots(Militant)}
\]

This event set means that Ted does not expect to be attacked by any type of weapon fire, enemy vehicles and militants during his movement. The Observe component monitors the virtual environment for key cues and forms observation sets. Suppose that the observation set of Ted about the current situation is:

\[
\text{Bot(A, Militant, Rifle) \land Weapon\_Fire(Rifle) \land BotGroup(Civilian, 50), \ldots}
\]

It means that Ted had been engaged by “bot A who is a male militant with a rifle, rifle fire, and a group of civilians are around, and . . .” This set of observation cues is used to compare with Ted’s Event Set B. The comparison reveals that the observations Bot(A, Militant, Male, Rifle) and Weapon\_Fire(Rifle) violate Event Set B cues of \neg Bots(Militant) and \neg Weapon\_Fire(All) respectively. This tells Ted that its expectation is not fulfilled and the current solution is likely to be invalid. This is consistent with the fact that the movement route of Ted had been compromised and the enemy may attempt to ambush Ted before he reaches the sniper’s location.
Dynamic Expansion of Evidence Sets

When a solution is deemed invalid, the observation cues that violate the event set cues will be combined with the current evidence set to form an expanded evidence set. This allows the bots to adapt to new situation rapidly. The Situation Awareness component will then retrieve new experience cases with the updated evidence set. The newly retrieved experience case will be sent to the Action component for processing. Consider the case when Ted had identified a sniper-assault threat and is currently performing the Counter Sniper solution. The current evidence set is

\[
[\text{Bot}(A, \text{Militant}, \text{Male}, \text{Snipergun}) \\
\wedge \text{Weapon\_Fire}(\text{Snipergun})] 
\]

New observation sets will continue to be monitored by the Observe component. As discussed before, suppose that the new observation set is:

\[
[\text{Bot}(B, \text{Militant}, \text{Rifle}, \text{Near}) \wedge \text{Weapon\_Fire}(\text{Rifle}) \\
\wedge \text{BotGroup}(\text{Civilian}, 50), \ldots] 
\]

The two new observations Bot(B, Militant, Male, Rifle) and Weapon\_Fire(Rifle) violate Event Set B of Ted’s expectation frame. These new observations are added to Ted’s existing evidence set. The dynamically expanded evidence set is thus changed to:

\[
[\text{Bot}(A, \text{Militant}, \text{Snipergun}, \text{Far}) \\
\wedge \text{Weapon\_Fire}(\text{Snipergun}) \\
\wedge \text{Bot}(B, \text{Militant}, \text{Rifle}, \text{Near}) \wedge \text{Weapon\_Fire}(\text{Rifle})] 
\]

The expectation violation will make Ted terminate the execution of the current solution and to re-assess the evolving situation. Using the new evidence set, Ted will identify a Close Combat Fire threat. This new threat is of immediate concern and takes precedence over the previous solution. Thus, the Close Combat Fire experience case is retrieved and Ted will perform the corresponding solution to this situation.

Experiments and Analysis

In this section, we first give a brief overview of Twilight City, a virtual environment built for MOUT simulations. Subsequently, we shall present our experiment results and analysis.

Overview of Twilight City

Twilight City aims to provide a high fidelity simulation platform for MOUT simulations. It is built on top of the Unreal Tournament (UT) engine with various modifications. Twilight City simulates urban warfare in an area of approximately 20 km by 20 km. There are more than thirty buildings in the virtual environment. Figure 5 shows some screen shots of Twilight City.

Experiment Results

To evaluate the performance of SNAP, particularly its effectiveness in dealing with dynamic changes in time critical tactical situations, we have conducted some experiments. In this section, we summarize the major results of these experiments. These experiments were conducted on a computer with Intel T2500@2 GHz processor and 2GHz RAM. The testing scenarios are
performed within *Twilight City*. They include engaging a set of 20 soldier bots with various threats such as *Sniper Assaults*, *Close Combat Fires*, *Bomb Assaults*, and *Air Strikes*. The objectives of the experiments are to test the behaviors of the soldier bots under different situations and the impact of expectations on the soldier’s behaviors. In particular, the behaviors of soldier bots during the *Sniper Assault* threat are compared. Three types of soldier bots were used to this end. They are S(A) bots which have expectations, S(B) bots which have no expectations, and UT bots which are equipped with the default tactics in UT. The S(A) bots and S(B) bots are also equipped with similar experiences to deal with various threats, such as *Sniper Assaults*, *Close Combat Fires*, etc.

In the testing scenario, the bots were first engaged by enemy sniper in *Sniper Assault* and subsequently hunted by enemy bots in *Close Combat Fire*. The average results from 10 runs are shown in Figure 6.

The results in Figure 6 shows how each type of soldier bots reacted in different tactical situations. For example, the data in the first column show that for the 20 soldier bots of S(A) type, under *Sniper Assault* threat, 2.1 of them just stood around, 10.6 of them initiated a *Counter Sniper* operation, none of them initiated a *Close Combat Fire* operation, 4.0 of them returned fire immediately, and 3.3 of them took cover first. In the *Close Combat Fire* threat, the militant bots chase and attack the soldier bots. For experienced human soldiers, when the threat has changed from *Sniper Assault* to *Close Combat Fire*, the natural reaction is to adapt their operation immediately. In this situation, continuing with the *Counter Sniper* operation may be useless since the movement of the soldiers are exposed. It is also more important to deal with the immediate threat of *Close Combat Fire*. It can be observed from the experimental results that the 9.1 of S(A) bots changed their operation into *Close Combat Fire* upon being contacted while only 1.0 of the S(B) bots changed their operation. This is because the S(A) bots are equipped with expectations that allow them to adapt their operations to deal with the dynamic changes. S(B) bots had already committed themselves to the *Sniper Assault* operation and will continue the operation even when attacked by enemies. They have no experience on how to deal with the threat of *Close Combat Fire*. 17.6 of the UT bots simply returned fire as they are designed to retaliate upon being attacked. Compared with the S(A) bots, the behaviors of S(B) bots are unrealistic in this situation. The S(A) bots, on the other hand, behaved more like real soldiers. They were able to retrieve the *Close Combat Fire* experience when the expectation violation invalidates the original solution.

Another experiment was conducted to compare the mortality rate between S(A), S(B), and default UT bots during a 20 minutes simulation. The simulation exposes the bots to *Sniper Assault*, *Close Combat Fire*, *Ambush* and *Bomb* threats in rapid succession. The average results of 10 simulation runs are shown in Figure 7.

As it can be observed, the mortality rate of the S(A) and S(B) bots are much lower than the UT bots. The number of UT bots that remain alive started decreasing in a faster rate than the SNAP bots (i.e., S(A) bots and S(B) bots) after 2 minutes. In 14 minutes, UT bots are totally destroyed by the threats, whereas an average

<table>
<thead>
<tr>
<th></th>
<th>Upon Sniper Assault</th>
<th>Upon Close Combat Fire</th>
</tr>
</thead>
<tbody>
<tr>
<td>S (A)</td>
<td>2.1</td>
<td>1.3</td>
</tr>
<tr>
<td>S (B)</td>
<td>3.1</td>
<td>0.5</td>
</tr>
<tr>
<td>UT</td>
<td>15.1</td>
<td>0.3</td>
</tr>
<tr>
<td>Stand Around</td>
<td></td>
<td></td>
</tr>
<tr>
<td>Counter Sniper</td>
<td>10.6</td>
<td>3.1</td>
</tr>
<tr>
<td>Close Combat Fire</td>
<td>0</td>
<td>9.1</td>
</tr>
<tr>
<td>Take Cover</td>
<td>3.3</td>
<td>1.1</td>
</tr>
<tr>
<td>Return Fire</td>
<td>4.0</td>
<td>2.9</td>
</tr>
</tbody>
</table>

S(A) – Bots with Expectations

S(B) – Bots without Expectations

UT – Unreal Tournament bots

Figure 6. Impact of expectations on bot behaviors.
number of 13.1 S(A) bots were alive. After the simulation terminated at 20 minutes, an average number of 8.7 S(A) bots survived. The results show that SNAP bots are able to handle time critical threats better than the UT bots. Between the SNAP bots, the S(A) bots performed significantly better than the S(B) bots. We also observed two major steep declines for S(B) bots at around 8 and 14 minutes. These timings coincide with the activation of new threats and information. The majority of the S(B) bots were destroyed during the activation of new threats as they could not adapt fast to the new threat. The superior performance of the S(A) bots and the steep decline in number of S(B) bots upon activation of new threats are consistent with our expectation that the mortality rate of the bots decreases when SNAP is equipped with expectations to deal with dynamic situational changes.

**Conclusion and Future Work**

Time critical decision-making models are important to generate realistic behaviors for the intelligent agents in virtual training systems and computer games. Formal decision rules based on complete rational analysis of all alternatives do not produce realistic behaviors as humans may not have sufficient time to rationalize their decisions in time critical situations and with incomplete information. Instead, past experiences may have a dominant role in determining how a human will behave in such situations. The SNAP decision-making framework that we discussed in this paper aims to imitate how humans make decision in time critical tactical situations for MOUT simulations. It uses the CBR cycle to enable the bots to make decisions in such situations with past experience cases.

It also uses the thin-slicing technique for rapid situation recognition with partial information about the current situation. To deal with the dynamic changes in MOUT simulations, we incorporated expectations into SNAP. The expectations serve to

1. Validate the initial solution.
2. Provide dynamic situation assessment.
3. Enable bots to adapt their behaviour to changing situations.
4. Allow bots to re-use past cues during experience matching.

Our experimental results demonstrate that the expectation model helps to deal with time critical and dynamic situations effectively.

We will continue to work on the proposed time critical decision-making framework for more complex situations. More experience cases will be investigated and added into the Experience Repository of the bots. The modeling of the temporal relation among expectations will also be investigated.

**ACKNOWLEDGEMENTS**

This work is supported in part by the Singapore National Research Foundation under Grant NRF2007IDM-IDM002-052.

**References**


Authors’ biographies:

Shang-Ping Ting is currently a trader in Nyenburgh. Previously, he worked as an engineer in the Modelling and Simulation Department at Defence Science & Technology Agency. He received his B.Eng. and M.Eng. in Computer Engineering from Nanyang Technological University (Singapore). At the moment, he is reading his Ph.D. in Nanyang Technology University. His research interests include: time critical decision making, situation awareness and human behavior representation in modeling and simulation.

Suiping Zhou is currently an Assistant Professor in the School of Computer Engineering at Nanyang Technological University (Singapore). Previously, he worked as an engineer in Beijing Simulation Center, China Aerospace Corporation, and then joined Weizmann Institute of Science (Israel) as a Post-doctoral fellow. He received his B.Eng., M.Eng. and Ph.D. in Electrical Engineering from Beijing University of Aeronautics and Astronautics (P.R. China). He is a member of IEEE and his current research interests include: large-scale distributed interactive applications (e.g., MMOGs), parallel/distributed systems, and human behaviour representation in modeling and simulation. He has published over 60 peer reviewed articles in these areas. He is currently an associate editor of the International Journal of Computer Games Technology. He has served as technical program committee member of many international conferences and workshops in computer games and virtual environments.