Available online at www.sciencedirect.com
ScienceDirect
Expert Systems with Applications 36 (2009) 455–463
Expert Systems with Applications
www.elsevier.com/locate/eswa

Modelling situation awareness for Context-aware Decision Support

Yu-Hong Feng, Teck-Hou Teng, Ah-Hwee Tan *

Intelligent Systems Centre and School of Computer Engineering, Nanyang Technological University, Nanyang Avenue, Singapore 639798, Singapore

Abstract

Situation awareness modelling is popularly used in the command and control domain for situation assessment and decision support. However, situation models in real-world applications are typically complex and not easy to use. This paper presents a Context-aware Decision Support (CaDS) system, which consists of a situation model for shared situation awareness modelling and a group of entity agents, one for each individual user, for focused and customized decision support. By incorporating a rule-based inference engine, the entity agents provide functions including event classification, action recommendation, and proactive decision making. The implementation and the performance of the proposed system are demonstrated through a case study on a simulated command and control application.

Keywords: Situation awareness; Context awareness; Decision support; Intelligent agents

1. Introduction

The concept of situation awareness is well established in the field of human factor studies in complex environments. According to Endsley (1988), situation awareness refers to "*the perception of the elements in the environment within a volume of time and space, the comprehension of their meaning and the projection of their status in the near future*". In the domain of command and control, wherein a designated commander is required to exercise authority and direction over various forces in order to achieve the given goals, a clear picture of the current situation and an accurate projection of the future states are essential for effective decision making. As such, situation awareness modelling has become an important component of command and control decision support systems. Over the years, guidelines and processes have been developed for building situation awareness models from a goal-oriented perspective (Endsley, 2001). However, to the best of our knowledge, none has incorporated the notion of context awareness for providing customized situation awareness.

Context awareness was introduced by Schilit and Theimer (1994) to develop software that adapts according to its locations of use, the collection of nearby people and objects, as well as changes to those objects over time. With technology advancements and the growing interest in mobile and wearable computation devices, context awareness has become one of the major research areas in those fields. The definition of context has also expanded from physical attributes to include device characteristics as well as user-specific factors, such as profiles and preferences (Dey, 2001; Harter, Hopper, Steggles, Ward, & Webster, 1999; Moran & Dourish, 2001).

Whereas situation awareness focuses on the modelling of a user's environment so as to help the user to be "aware of his current situation", context awareness is about exploiting the context of a user and helping the user to have a more effective interaction with the system by actively changing the system's behavior according to the user's current context or situation. In the domain of command and control, individual users require specific sets of situation awareness. Also, the same piece of information may have different meanings and usages for different people in the

* Corresponding author.
E-mail addresses: yhfeng@ntu.edu.sg (Y.-H. Feng), teng0032@ntu.edu.sg (T.-H. Teng), asahtan@ntu.edu.sg (A.-H. Tan).

0957-4174/$ - see front matter © 2007 Elsevier Ltd. All rights reserved.
doi:10.1016/j.eswa.2007.09.061

same environment. Therefore, the system must know the context of the current user and forward focused contextual information to the user. Building a single situation model and presenting the full set of information to all users is not only inefficient but highly confusing. On the other extreme, building separate situational awareness models for individual users is not only inefficient but more importantly, raises the issue of consistency.

In this paper, we present a system known as CaDS (for Context-aware Decision Support) that incorporates a shared situation awareness model but provides individual human operators with customized views and services through a group of entity agents. The entity agents, one for each individual user, communicate with the situation model and extract information of relevance for presentation to their respective users in accordance to the user context. To reduce the cognitive load of the human operators, the entity agents perform event classification and action recommendation in a proactive manner. We have applied CaDS to a simulated command and control domain and evaluated its performance based on several mission scenarios. Our experimental results show that the entity agents are able to perform event classification and action recommendation with a high level of accuracy and thereby reduce the cognitive load of the human operators significantly.

The rest of the paper is organized as follows. Section 2 presents the overall system architecture. Section 3 discusses the various components in the shared situation awareness model. Section 4 presents the design and implementation of the entity agents for context-aware decision support. Section 5 illustrates the various system functionalities through a case study on the command and control application. Section 6 describes our evaluation methodology and reports the experimental results. Section 7 discusses related work. The final section concludes and outlines the future work.

2. System overview

Referring to Fig. 1, the CaDS architecture incorporates a situation model managing the shared situation awareness model and a group of entity agents, supported by an underlying game simulation engine. The game simulation engine GECCO (for Game Environment for Command and Control Operations) GECCO (2004) is a publicly available generic platform for creating and playing real-time multiplayer strategy games.

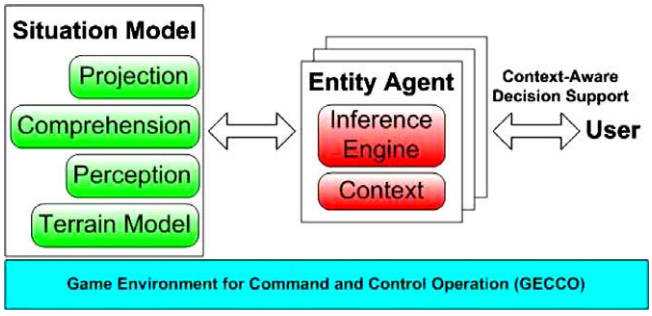

Fig. 1. The CaDS system architecture.

The situation awareness model is based on the Endsley's three layer structure (Endsley, 1995), but with the addition of a terrain model for the command and control domain. Assuming the existence of a sensor network, incoming data are first represented at the Perception layer (level 1) of the situation model. The Comprehension layer (level 2) then interprets the data and provides assessment of the current situation. To support anticipatory decision making, the Projection layer (level 3) predicts future states based on the understanding of the current situation. At present, only some basic projection functions have been implemented for illustration purpose. All three layers of the situation awareness model function concurrently and iteratively.

Each entity in the environment is represented by an entity agent. Each agent has a set of goals and strategies, which form the basis of the entity's behaviors. The goals and strategies are in turn translated into executable structures like missions, plans and actions. Together with the physical attributes such as location and strengths, all these constitute the context based on which an agent provides a customized set of views and services of the shared situation awareness model to its user. Given a user context, an entity agent pro-actively scans the situation awareness model for relevant information in the environment. It then highlights important events to the user according to their significance in the given context. Our current implementation of the entity agents is based on the production rule representation and a logical reasoning mechanism.

3. Situation awareness modeling

We describe the various levels of the situation awareness model, including the terrain model, in the following sections.

3.1. Terrain model

Representing the knowledge of the physical world, terrain information forms the basis of all three layers of situation awareness as well as context-aware decision support. As such, the situation model must be able to monitor the dynamic changes in the terrain, which in turn may result in significant events in the situation awareness model.

The *terrain model* consists of two key elements, namely *nodes* and *links*. A *node* represents a critical point or location on the map with the necessary descriptive attributes. A *link* corresponds to a connection or path between a given pair of *nodes*, with the necessary attributes describing the condition of the connection as well as a *weight* attribute providing distance information. The terrain model further incorporates a shortest path algorithm for identifying the optimal route between any two nodes in the terrain.

To facilitate the process of building and editing the terrain model, an interactive software tool named Terrain

Builder (Fig. 2) has been developed in-house. Using the notions of *nodes* and *links*, a terrain model can be easily constructed from a two-dimensional map by clicking on the appropriate positions on the map.

3.2. SA Level 1: Perception

The key elements in the level 1 of the situation model are entities and events. An entity represents an object in a situation which has attributes such as identities, capabilities, and so on. Virtually everything in the environment can be an entity. As shown in Fig. 3, the *Entity* class is a general description of an object in a situation with the basic and necessary attributes, such as *identity*, *trajectory*, and *childEntities*. The trajectory is a sequence of movement made by the entity.

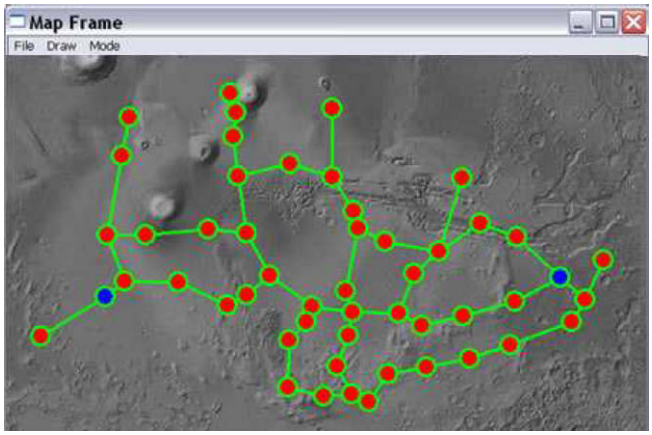

Fig. 2. The Terrain Builder.

The *Event* class is a data structure encapsulating all the relevant information of a physical occurrence in a situation. These informations are decomposed into five elements, i.e. the five W's, namely When, Where, Who, What and Why. An event injection triggers the system to reassess the relevant entities' attributes and their relations with the others. These changes in turn result in a new state of situation awareness and may eventually lead to critical decisions.

3.3. SA Level 2: Comprehension

The comprehension level makes sense out of the data provided by the perception level and integrates them into meaningful pieces of information. Consider a movement scenario as an example, in which a space vehicle is tasked to move from one location to another and to avoid aliens along the way. In this scenario, a list of high level functions is essential for providing situation comprehension for individual entities. An example list of such functions is given below.

- *Distance:* Calculating the distance from the current location to the destination location.
- *Speed required:* Calculating the speed required to reach the destination on time.
- *Fuel required:* Calculating the fuel required to reach the destination based on the current consumption rate.
- *Enemy on the route:* Scanning the planned route for enemy.
- *Enemy around the route:* Scanning the areas around the planned route for potential threats.

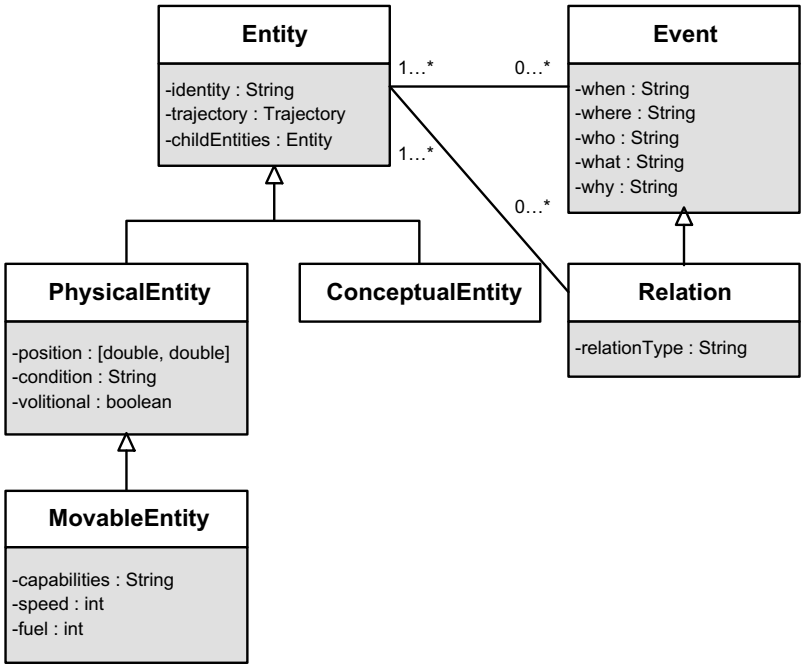

Fig. 3. The schema of the situation model.

- *Enemy behind:* Checking whether there is any enemy coming up from behind.
- *Zone of alert:* Evaluating the conceptual proximity of enemy in terms of zones.

By exploiting the relative location information provided by the enemy detection functions and the proximity information by the zone of alert functions, an entity is in a better position to formulate its strategies and plans. For instance, if an enemy is on the planned route and in zone 3 (meaning far away), an entity might choose to continue on this route as the enemy is still relatively far away and it may eventually go out of its route. However, if the enemy is in zone 1 (meaning near), the user might choose to re-plan his route.

3.4. SA Level 3: Projection

The ability to foresee the future is especially crucial in command and control. Based on the understanding of the current situation, it is possible to predict the future states to a certain extent. For illustration, we have implemented three types of projection functions: *Enemy location projection*, *terrain projection* and *goal status projection*.

Enemy location projection is responsible for predicting the future locations and the corresponding timing of a hostile entity. The system keeps a finite history of the time–space information on the enemy. Statistical inference is then performed over these historical data in order to estimate the entity's expected location at a particular time. Data points far back in the history are either ignored or given a lower priority in estimating the new locations.

Whenever there is a change in terrain data, *terrain projection* is conducted to predict the possible future changes, based on the existing knowledge of terrain changes. For example, if a road is blocked due to a heavy rain and our prior knowledge informs that it usually takes up to three hours for a blocked road to clear, the system would then expect to observe another terrain change to unblock the road within three hours.

As mentioned, an important agenda of an entity is goal attainment. Thus an entity needs to assess the goal status at a regular time interval. Using the present information, such as the planned route, the entity's fuel level and its current speed, the projection layer of the situation awareness model is able to compute the expected time to accomplish the goals. All these projected situations then form a part of the situation awareness.

4. Context-aware decision support

The primary role of an entity agent is to provide context-aware decision support in a command and control setting so that decisions can be made in a timely and effective manner. To this end, we have developed entity agents with a range of context-aware capabilities, including event classification, action recommendation and decision mode selection. Referring to Fig. 4, each entity agent communicates with the

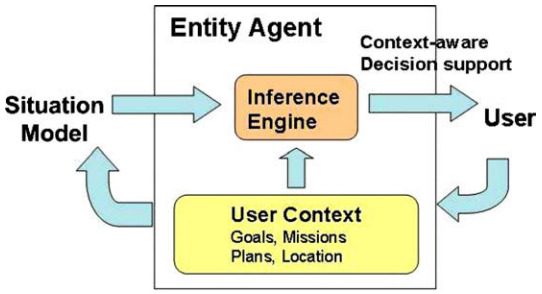

Fig. 4. The entity agent model.

situation model and employs a rule-based inference engine to provide decision support to its user according to the user context model. The user context used by the agents consists of goals, plans and physical attributes, such as location and capabilities. An agent's goal defines the target states of the corresponding entity. An agent's plan contains the planned sequences of actions to achieve these goal states.

4.1. The inference engine

As the activities of an entity should be driven by its goals, its attention must be directed accordingly. Using an iterative process, the attainment of goals must be continuously monitored and assessed. Based on the user context, especially the goals and plans, an entity agent should thus constantly look out for situations in the environment that may affect its goal attainment.

Our current implementation of the entity agents is based on a production rule based engine known as DROOL (Rupp, 2004), which is publicly available and fully open-source. Being one of the established ways to implement situation awareness, production rules provide us with a simple and comprehensible specification for recording the heuristics articulated by the domain experts. Based on the Forgy's Rete algorithm, the rules in Drools can be written in Java, Python and Groovy via the use of XML.

Each rule in Drools consists of a list of parameters, the IF conditions and the THEN consequence. In addition, a salience attribute can be assigned to each rule, as a form of priority in the activation queue. The working memory is stateless and it contains all the knowledge or facts. Facts can be asserted, modified and retracted from the working memory. Rules have to be loaded into the working memory before firing. A rule firing (execution) can modify the content of the working memory and therefore possibly trigger other rules to fire.

4.2. Event classification

To reduce the cognitive load of human operators, information should be presented to a user only if it is of relevance and significance to the user. In our system, a user corresponds to an entity in the simulated world. For example, an explorer commander is represented as an explorer

entity in the simulation. Therefore, the explorer commander should only receive relevant and significant information with respect to his/her context.

Specifically, information should be presented according to their impact on the goals of an entity. Depending on how an event may affect the goal attainment status of the entity, it is presented to the user in one of the three message levels: Events that do not affect goals are posted as *Information*, events that may affect goals are issued as *Warning*, and events that endanger goals are flagged as *Alert*.

The event classification rule module assigns an appropriate message level to each incoming event that would be presented to the user according to the situation assessed. A typical event classification rule for an explorer[1] is as shown below. The *TerrainRouteBlocked* rule checks that if the situation type is *Terrain* and the event will affect the current route of the entity, the message level is set to *Alert*.

```
RULE name = TerrainRouteBlocked
  IF
    SituationType==Terrain
    AffectCurrentPath is TRUE
  THEN
    MsgLevel = Alert
```

4.3. Option generation and evaluation

For effective decision making in a given situation, a human operator needs to gain a good level of situation awareness by assessing his current situation. With the situation awareness, he can then consider the options of actions that can be performed and decide on the best options available. An entity agent facilitates this process by monitoring the current situation and recommending possible courses of actions to its user when the need arises.

The generate choice rule module is responsible for deducing choices or options for a particular situation. Currently, there is a total of five action choices available for any given situation, namely *Resume*, *Increase Speed*, *Decrease Speed*, *Reroute* and *Wait*. The rules determine if a choice is applicable in a given situation and generate a list of appropriate choices from the set of actions. For instance, the following rule illustrates the generation of a choice. Under the situation type of *Terrain*, if the rule detects that the current route is blocked and an alternate route is available, a *Reroute* choice is created and inserted into the choice list.

```
RULE name = SituationTl
  IF
    SituationType==Terrain
    CurrentPathIsBlocked is TRUE
    AlternateRouteAvailable is TRUE
  THEN
    action = Reroute
```

[1] Different entities can be associated with different rules.

Given the current situation, the entity agent goes through the complete set of rules, each of which checks for specific conditions and generates a choice when appropriate. The result of this process is a list of action choices ready for ranking.

The rank choice rule module assesses each of the available choices for a given situation with a score between 0 and 1. Generally, the scores are calculated as a function of minimal changes required and payoff in terms of goal attainment. Thus, the option with the minimal change is preferred among those with the same payoff. For example, in the movement scenarios, a *Reroute* action is a high change option, whereas *Resume* is a low change action. Therefore, given the same level of payoff, *Resume* is preferred over *Reroute*.

4.4. Mode selection

To exploit their proactive capabilities, we further allow entity agents to take an action automatically when appropriate. The guiding principle is that when a decision is of high confidence and low significance, the action can be carried out without waiting for an explicit instruction from the human operator.

Currently, there are three decision modes defined in the system. In the *Auto* mode, an entity agent automatically chooses the best option available for the current situation. The *Recommend* mode is often used when the selected option is of high consequence, wherein the agent raises a dialog prompt with a ranked list of the available options for the user. The *Don't Know* mode is used when the system does not recognize the current situation and/or the system cannot identify an appropriate action.

The mode selection rule module evaluates the decision mode for the current situation according to the ranked list of choices. If the agent cannot identify an action choice that can attain the assigned goal, the decision mode is set to *Don't Know*. If the top choice is of high consequence, the *Recommend* mode is used to prompt the user for a final decision. Otherwise, the decision mode is set to *Auto*, under which the agent will carry out the top action choice automatically without the need for user intervention.

The following is a sample rule used for mode selection. This rule says that the decision mode is set to *Auto* if the situation type is *GoalStatus* and the agent has found that it is impossible to attain the goal with the given resource. Consequently, the agent will report to the headquarter automatically that the entity is unable to attain its assigned goal.

```
RULE name = GoalStatusReportToHQ
  IF
    DecisionAssist is TRUE
    SituationType==GoalStatus
    GoalAttainment is FALSE
  THEN
    DecisionMode = Auto
```

5. Case study: Mission on mars

Our implementation of CaDS is based on GECCO (GECCO, 2004), which is capable of supporting various scenarios, including war strategy games, fire fighting and rescue missions. A simple exploration and movement mission is used in our study to demonstrate the various capabilities described.

Given a fictional mission on Mars, a space vehicle *Explorer ER100* is tasked to explore the unknown areas on Mars. With a sensor network deployed on the explored regions, the areas are continually monitored and activities are reported to the situation model. Consider a scenario in which the Explorer ER100 is required to return to its space station before running out of fuel. Along the way, the explorer may encounter aliens which will threaten his survival and thus the attainment of the goal. Another threat that may affect goal attainment is the changes in the terrain situation. For example, an onset of bad weather may prevent it from reaching the destination on time. In our simulated environment, there is another entity called Headquarter that oversees the Explorer's mission and makes critical decisions only when necessary.

During the simulation, two types of events are injected into the virtual environment: terrain events and entity events. A terrain event changes the terrain model, for instance, by blocking a particular node. An entity event leads to changes in a selected entity, either in terms of its attribute values or activities. As the situation changes upon each event injection, the situation model is constantly updated accordingly. In each cycle, the agent re-assesses the situation, gains a new understanding and possibly projects future changes.

As shown in the CaDS graphical interface (Fig. 5), all entities are displayed as image icons overlayed on the terrain map. Each entity is represented and assisted by an entity agent so that the sensed data in the situation awareness model are processed and filtered and only relevant information are presented to the user. For each entity, its goals, missions and plans are displayed accordingly on the main panel, together with other important attributes, such as the fuel level and the health index. Each entity agent communicates with its user through a message panel (in the bottom right corner), in which incoming events are displayed according to their significance, and recommendations are provided to the user as the situation arises. The message panel and the graphical terrain map together enhance the user awareness of the situation in a context-aware manner. The display on the interface automatically switches into the corresponding perspective when a different entity is selected.

Besides classifying and displaying events in the appropriate message levels, the entity agent pro-actively assesses the situation for available action options and ranks them accordingly. If an action is required of which the entity agent has no authority to perform, the agent raises a choice dialog window to the user, with a ranked list of the actions and its recommendations. As shown in Fig. 6, an alien is spotted along the planned route of ER100. The agent detects the threat, assesses the various options, and recom-

Fig. 5. The context-aware graphical interface of CaDS.

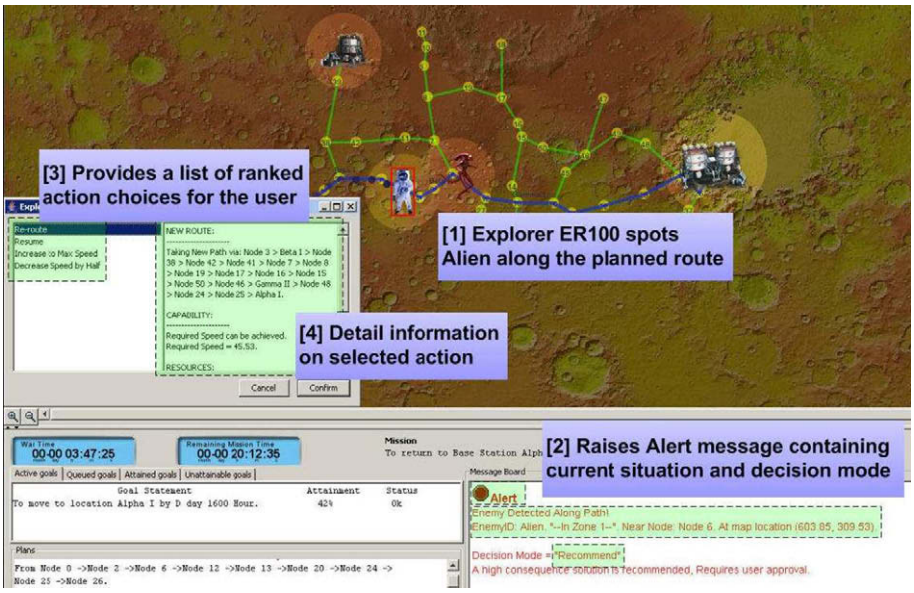

Fig. 6. An illustration of context-based decision support.

mends *Reroute* as the best solution. As *Reroute* is a major decision, the action is posted as an recommendation to the user together with a detailed description of the analysis and reasoning process.

6. Performance evaluation

In this section, we present an experimental study in which a human operator is asked to gauge the accuracy of the entity agent acting for Explorer ER100 in terms of event classification and action recommendation. The human operator is to determine, given a specific situation, if the system classifies an incoming event or recommends an action in an appropriate manner. In addition, we estimate the effectiveness of the ER100 agent in terms of the reduction in the cognitive load of its commander.

During the course of simulations, events can be generated based on either implicit or explicit triggers. Implicit triggers are generated from the natural evolution of the simulation (e.g., the programmed responses of an entity towards the others), whereas explicit triggers are the results of external event injection. We use the following two key parameters in generating the test scenarios: (i) **Initial position**: The initial position of ER100 determines its route selection and therefore influences the subsequent evolution of the simulation; and (ii) **Terrain change**: Any variation in the terrain data, for example the blocking of a road and the clearing of a blocked road, may induce the entity into making the necessary adjustment to its current plan.

Each distinct initial position of ER100 marks a unique scenario setting. In each scenario, multiple situations are created and decisions are made as the scenario plays out. A total of 73 situations based on five scenarios have been collected. The goal of ER100 remains unchanged throughout all the scenarios.

6.1. Event classification

A primary role of entity agents is to classify incoming events into three levels of significance, namely *Information*, *Warning* and *Alert*. We define two performance measures for this function, namely the accuracy of event classification and the reduction in the cognitive load for the human operators.

Based on the test situations, a human operator is asked to gauge the accuracy of the system in terms of event classification. When the message level assigned by CaDS to an event is different from that of the operator, the specific event classification is deemed as inappropriate. As shown in Table 1, the entity agent achieves a high accuracy of 100.0% in classifying events. This high level of performance is achievable as the task of classifying events is relatively straightforward and our rules are sufficiently rich to cover most types of events.

Context-based classification of events contributes to the reduction of cognitive load on the human operators. Instead of having to attend to each and every incoming events, a user now only needs to pay attention to events classified as *Warning* and *Alert*. To evaluate the system's performance in quantitative terms, we define the *cognitive load reduction* (CLR) index for an event type c formally

Table 1
The prediction accuracy in event classification

	Number of predictions	Number of correct	Prediction accuracy (%)
Information	30	30	100.0
Warning	34	34	100.0
Alert	9	9	100.0
Total	73	73	100.0

Table 2
The CLR indices for warning and alert events

Total number of events	Number of warnings	Number of alerts
73	35	8
CLRI	52.1%	89.0%

Table 3
The prediction accuracy in action recommendation

Action choice	Number of predictions	Number of matches	Prediction accuracy (%)
Increase speed	2	1	50.0
Decrease speed	0	0	NA
Resume	20	15	75.0
Reroute	23	23	100.0
Wait	0	0	NA
Total	45	39	86.7

as $R_c = 1 - (N_c/\mathcal{N})$, where N_c is the number of event c requiring attention and N is the total number of incoming events. As shown in Table 2, the entity agent has been effective in reducing the cognitive load of the ER100 commander by 52.1% for *Warning* and 89.0% for *Alert*.

6.2. Action recommendation

In this set of the experiments, we compare the recommendations made by the entity agent against the preferred action choices of the human operator. As observed in Table 3, among the actions chosen, the entity agent has a prediction accuracy of 50.0%, 75.0% and 100.0% for *Increase Speed*, *Resume* and *Reroute*, respectively. Fig. 7 shows an instance of the incorrect action recommendations. The resume action was suggested to the human operator when in fact the reroute option would be the more appropriate one. As the scenario evolved, the Explorer ER100 was observed to sustain attack by the alien.

We note that not all the action choices are used during the experiments. This indicates that the specified rules have not been able to cover the space of the available actions adequately, especially on the aspect of controlling speed. Potentially, more rules can be added to cover this deficiency. However, the level of system complexity increases as the decision space gets more extensive with the increased numbers of rules and situation parameters. In particular, as different initial conditions may give rise to a variety of situations as the scenario evolves, the task of addressing every single possible situation is a great challenge.

7. Related work

Intelligent agents have been used for situation awareness modelling in a number of prior work. Urlings, Tweedale, Sioutis, and Ichalkaranje (2003) employ BDI (Beliefs–Desires–Intentions) agents to enhance situation awareness of human–machine team in a military environment. Based on the JACK agent development environment and the Unreal Tournament (UT) game engine, they present a multi-agent framework. Three types of agents are described, including a commander agent for supporting the human commander, a communication agent for communication between the commander agent and the human commander, and troop agents for exploring the environment and for executing the orders. Although their commander agent is similar to our entity agents, Urling et al. are focusing on situation awareness in single agent and they

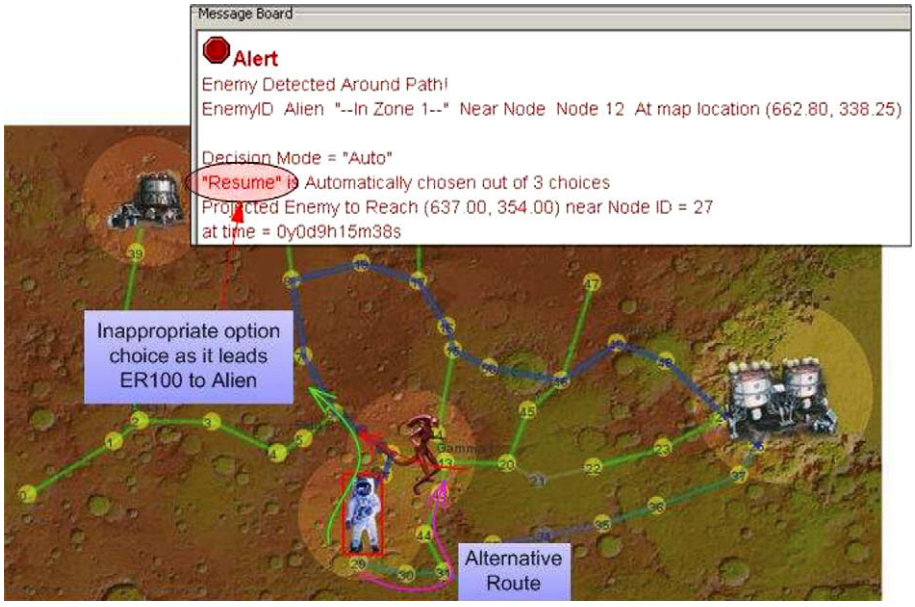

Fig. 7. An incorrect prediction by CaDS.

do not employ the concept of shared situation awareness. In addition, although a high level description of the commander agent is provided, there is no concrete account on the specific decision support functions implemented and their evaluation.

So and Sonenberg (2004) incorporate the notion of situation awareness into a computational model for proactive agent behavior. Their model is based on a rule-based knowledge representation and a forwarding reasoning mechanism. While they develop a meta-control strategy for directing an agent's attention in sensing and deliberation, their focus again is on single agent and the approach is to build a situation awareness model for each agent.

More recently, Thangarajah, Padgham, and Sardina (2006) also adopt the notion of 'situation' for decision making in intelligent agents. Instead of using the Endsley's three-level hierarchy, they treat situations as a conceptual entity which arguably allows a richer semantics specification. For modelling the agents' reasoning process, Thangarajah et al. extend a BDI language called CAN for rule-based specification of the agents' behavior. Although they see the need of organizing situation objects according to their relevance to individual agents for efficiency reason, their system again does not have the notion of shared situation awareness and context-aware decision support.

8. Conclusions

We have shown how context-awareness can be exploited in situation assessment and how context-aware decision support can be used to reduce the cognitive load of human operators in the command and control domain. Compared with traditional situation awareness models, our system provides customized views and services out of a shared situation awareness so as to support focused and effective situation awareness based on context. More importantly, we illustrate that a context-based inference engine can be used to support a myriad of proactive decision support functions that facilitate commanders to operate in complex and time critical operations. Whereas Endsley has also advocated a goal-oriented approach to designing situation models, we provide a computational model as well as its implementation for exploiting goal-based contextual information to achieve user-specific situation awareness.

While the current decision support engine based on production rules has served the purpose of knowledge acquisition and the proof of concepts, the task of defining the necessary heuristics based on a bounded definition of the command and control application and responding to each and every new development can be tedious. Instead of using rule-based specification, machine learning capabilities can be incorporated to aid in expanding the scope of the decision-making module. In fact, the task of incorporating learning for situation awareness and context-based decision support has been identified as our next research target. Ultimately, we are moving towards a cognitive decision system that will support both direct rule-based knowledge specification as well as learning based knowledge acquisition based on human-agent interaction.

Although our experimentation thus far is based on a relatively simple scenario of explorer movement and alien avoidance, the framework theoretically can be applied to the general domain of command and control, including battlefield modelling and tactical warfare planning. These will form part of our future work.

Acknowledgements

The reported work is supported in part by a research grant from the Intelligent Systems Centre. The authors thank Jun Jin and Chun-Yip Lam for contributing to the development of the Mission On Mars simulator.

References

Dey, A. K. (2001). Understanding and using context. *Personal and Ubiquitos Computing, 5*, 4–7.

Endsley, M. R. (1988). Design and evaluation for situation awareness enhancement. In *Proceedings of the human factors society 32nd annual meeting*, Santa Monica, CA (Vol. 1, pp. 97–101).

Endsley, M. R. (1995). Toward a theory of situational awareness in dynamic systems. *Human Factors, 37*, 32–64.

Endsley, M. R. (2001). Designing for situation awareness in complex systems. In *Proceedings of the second international workshop on symbiosis of humans, artifacts and environment*, Kyoto, Japan.

Gecco (2004). Game environment for command and control operations. Available from:<http://www.csc.kth.se/tcs/gecco/>.

Harter, A., Hopper, A., Steggles, P., Ward, A., & Webster, P. (1999). The anatomy of a context-aware application. In *Proceedings of the 5th annual ACM/IEEE international conference on mobile computing and networking*, Seattle, WA (pp. 59–68).

Moran, T. P., & Dourish, P. (2001). Introduction to this special issue on context-aware computing. *Human Computer Interaction, 16*, 87–95.

Rupp, N. A. (2004). An introduction to the drools project. Available from: <http://www.theserverside.com/tt/articles/article.tss?l=Drools/>.

Schilit, B., & Theimer, M. (1994). Disseminating active map information to mobile hosts. *IEEE Network, 8*, 22–32.

So, R., & Sonenberg, L. (2004). Situation awareness in intelligent agents: Foundations for a theory of proactive agent behavior. In *Proceedings of IEEE/WIC/ACM international conference on intelligent agent technology* (pp. 86–92).

Thangarajah, J., Padgham, L., & Sardina, S. (2006). Modelling situations in intelligent agents. In *Proceedings of fifth international joint conference on autonomous agents and multiagent systems*, Hakodate, Japan (pp. 1049–1051).

Urlings, P., Tweedale, J., Sioutis, C., & Ichalkaranje, N. (2003). Intelligent agents and situation awareness. In *Proceedings of the 7th international conference on knowledge-based intelligent information and engineering systems*, LNCS 2774 (pp. 723–733).