NAVAL POSTGRADUATE SCHOOL

MONTEREY, CALIFORNIA

DISSERTATION

EVENT PREDICTION FOR MODELING MENTAL SIMULATION IN NATURALISTIC DECISION MAKING

by

Dietmar Kunde

December 2005

Dissertation Supervisor: Chris Darken

Approved for public release; distribution is unlimited

THIS PAGE INTENTIONALLY LEFT BLANK

REPORT DOCUMENTATION PAGE			Form Approved OMB No. 0704-0188
Public reporting burden for this collection of information is estimated to average 1 hour per response, including the time for reviewing instruction, searching existing data sources, gathering and maintaining the data needed, and completing and reviewing the collection of information. Send comments regarding this burden estimate or any other aspect of this collection of information, including suggestions for reducing this burden, to Washington headquarters Services, Directorate for Information Operations and Reports, 1215 Jefferson Davis Highway, Suite 1204, Arlington, VA 22202-4302, and to the Office of Management and Budget, Paperwork Reduction Project (0704-0188) Washington DC 20503.			
1. AGENCY USE ONLY ***(Leave blank)***	**2. REPORT DATE** December 2005	**3. REPORT TYPE AND DATES COVERED** Dissertation	
4. TITLE AND SUBTITLE: Event Prediction for Modeling Mental Simulation in Naturalistic Decision Making			**5. FUNDING NUMBERS**
6. AUTHOR(S) Dietmar Kunde			
7. PERFORMING ORGANIZATION NAME(S) AND ADDRESS(ES) Naval Postgraduate School Monterey, CA 93943-5000			**8. PERFORMING ORGANIZATION REPORT NUMBER**
9. SPONSORING / MONITORING AGENCY NAME(S) AND ADDRESS(ES) MOVES Institute Center for the Study of Potential Outcomes Monterey, CA 93943-5000			**10. SPONSORING / MONITORING AGENCY REPORT NUMBER**
11. SUPPLEMENTARY NOTES The views expressed in this thesis are those of the author and do not reflect the official policy or position of the Department of Defense or the U.S. Government.			
12a. DISTRIBUTION / AVAILABILITY STATEMENT Approved for public release; distribution is unlimited.			**12b. DISTRIBUTION CODE**

13. ABSTRACT

Nearly all armies of the Western Hemisphere use modeling and simulation tools as an essential part of performing analysis and training their leaders and war fighters. Tremendous resources have been applied to increase the level of fidelity and detail with which real combat units are represented in computer simulations. Current models digress from Lanchester equations used for modeling the big Cold War scenarios towards modeling of individual soldier capabilities and behavior in the post Cold War environment and increasingly important asymmetric warfare scenarios. Although improvements in computer technology support more and more detailed representations, human decision making is still far from being automated in a realistic way. Many "decisions" within a simulation are based on overly simple models and hardly at all on cognitive processes. One cognitive model in naturalistic decision making is the Recognition Primed Decision Model developed by Klein and Associates. It describes how the actual process humans use to come up with decisions in certain situations is radically different from the traditional model of rational decision making. Mental simulation is an essential part of this model in order to picture possible outcomes in the future for potential courses of actions. This research provides a computational model for mental simulation in a combat simulation environment. It generates the look into the near future with a finite Markov Chain as one instance of several possible predictive models. The results of the model are compared with preliminary human experimental data. The experiments show that the model developed performs in the human range with respect to prediction and decisions. This research shows that entities in a combat simulation environment having the capability of looking ahead into the near future based on statistical data perform more realistically than those that just use the information of the present, not even including the past.

14. SUBJECT TERMS Mental Simulation, Naturalistic Decision Making, Event Prediction, Combat simulation, Human Behavior Representation, Agent-Based Simulation			**15. NUMBER OF PAGES** 145 **16. PRICE CODE**
17. SECURITY CLASSIFICATION OF REPORT Unclassified	**18. SECURITY CLASSIFICATION OF THIS PAGE** Unclassified	**19. SECURITY CLASSIFICATION OF ABSTRACT** Unclassified	**20. LIMITATION OF ABSTRACT** UL

NSN 7540-01-280-5500 Standard Form 298 (Rev. 2-89)
Prescribed by ANSI Std. 239-18

THIS PAGE INTENTIONALLY LEFT BLANK

Approved for public release; distribution is unlimited

EVENT PREDICTION FOR MODELING MENTAL SIMULATION IN NATURALISTIC DECISION MAKING

Dietmar Kunde
Lieutenant Colonel, German Army
M.S. Surveying, German Armed Forces University Munich, 1985
M.S. Operations Research, Naval Postgraduate School, 1997

Submitted in partial fulfillment of the
requirements for the degree of

DOCTOR OF PHILOSOPHY IN MODELING, VIRTUAL ENVIRONMENT AND SIMULATION

from the

NAVAL POSTGRADUATE SCHOOL
December 2005

Author: _____
Dietmar Kunde

Approved by:

_____	_____
Chris Darken, Ph.D.	Rudy Darken, Ph.D.
Professor of Computer Science	Professor of Computer Science
Dissertation Supervisor and	Director, MOVES Institute
Dissertation Committee Chair	
_____	_____
Thomas M. Cioppa, Ph.D.	Thomas Otani, Ph.D.
Naval War College	Professor of Computer Science
_____	
John Hiles	
Professor of MOVES	

Approved by: _____
Dr. Rudy Darken, Chair, MOVES Academic Committee

Approved by: _____
Julie Filizetti, Associate Provost for Academic Affairs

THIS PAGE INTENTIONALLY LEFT BLANK

ABSTRACT

Nearly all armies of the Western Hemisphere use modeling and simulation tools as an essential part of performing analysis and training their leaders and war fighters. Tremendous resources have been applied to increase the level of fidelity and detail with which real combat units are represented in computer simulations. Current models digress from Lanchester equations used for modeling the big Cold War scenarios towards modeling of individual soldier capabilities and behavior in the post Cold War environment and increasingly important asymmetric warfare scenarios. Although improvements in computer technology support more and more detailed representations, human decision making is still far from being automated in a realistic way. Many "decisions" within a simulation are based on overly simple models and hardly at all on cognitive processes. One cognitive model in naturalistic decision making is the Recognition Primed Decision Model developed by Klein and Associates. It describes how the actual process humans use to come up with decisions in certain situations is radically different from the traditional model of rational decision making. Mental simulation is an essential part of this model in order to picture possible outcomes in the future for potential courses of actions. This research provides a computational model for mental simulation in a combat simulation environment. It generates the look into the near future with a finite Markov Chain as one instance of several possible predictive models. The results of the model are compared with preliminary human experimental data. The experiments show that the model developed performs in the human range with respect to prediction and decisions. This research shows that entities in a combat simulation environment having the capability of looking ahead into the near future based on statistical data perform more realistically than those that just use the information of the present, not even including the past.

THIS PAGE INTENTIONALLY LEFT BLANK

TABLE OF CONTENTS

LIST OF FIGURES

LIST OF TABLES

THIS PAGE INTENTIONALLY LEFT BLANK

ACKNOWLEDGMENTS

The single author's name of a dissertation doesn't reflect that also many other people have made direct or indirect contributions to the thesis beyond the ones listed in the References. The space available in this thesis is too short to express my gratitude to all those who have supported and encouraged me during the past three years, but there is a handful I would like to mention here.

First of all, I would like to thank my supervisor Dr. Christian J. Darken for his encouraging way of guiding me through the Doctoral Program, for keeping me focused, for his patience with my drafts from a nonnative English speaker, and for his invaluable comments during the whole work with this dissertation.

I also want to thank my committee members, who always were demanding in a supportive way and made scheduling of milestones an easy issue.

Special thanks belongs also to the development team of Combat XXI without their support the pain would have been much greater.

I also want to thank Lee Rappold and Curtis Blais for reviewing the manuscript and making it "English."

Finally, I would like to thank my wife 'Utemaus' and my children Katharina, Justus, and Rolf who always supported and motivated me. Because of their love and support, failing was not an option.

THIS PAGE INTENTIONALLY LEFT BLANK

I. INTRODUCTION

A. THESIS STATEMENT

This thesis shows that the methodologies of statistical event prediction can be used to effectively model mental simulation for improving human models in combat simulations. "Mental simulation" in this context means the ability of software agents to simulate future events in order to evaluate their own courses of actions in combat simulations and to hypothesize events that might occur given the current and past situation.

B. PROBLEM STATEMENT

Running combat simulation models for training purposes is very time- and personnel-intensive, because the low degree of artificial intelligence possessed by the constructed units in the simulation requires both extensive manual input of initial orders and human monitoring during the simulation run. The capabilities of autonomously acting units are very limited. The range of modeling military decision making goes from sophisticated methodologies, e.g., comparing scored values of possible actions and taking the highest or the lowest value depending on circumstances (Norling et al., 2000), to less sophisticated cases, in which units execute their initial orders according to an internal script – these are mainly "movement orders" – and react to opposing fire, properties of the terrain, or movement data. Their perception of the environment is restricted to that which is directly relevant to the application domain: for example, a simulated tank commander knows only certain knowledge about tank combat, nothing else. By contrast, many human tank commanders have life experiences that may influence their decisions more strongly than domain knowledge (Forsythe, 2000). For many research prototypes of agents the learning capability has been addressed. However, in combat simulation models these issues have not been implemented in a satisfying scale. Therefore, the lack of a learning component of simulated commanders precludes them from making complex decisions at a scale to humans; many decision making aspects of the simulation have to be resolved externally and then put into the system, requiring much skilled assistance.

Increasing the degree of Artificial Intelligence (AI) increases the cost-effectiveness of a simulation system during its use. Fewer staff are needed for scenario

input and system setup. During a run, the greater autonomy of the system leads to longer decision cycles before the units reach unreasonable or unacceptable conclusions. With enhanced AI, an assistant can control or monitor more units, a factor that is especially valuable for the analytical application domain of modeling and simulation, for which there is usually of a paucity of personnel as compared to a training environment. However, there is also a drawback. Decisions made within the system by simulated commanders are not normally as good or as high-quality as decisions made by human commanders, an observation valid not only in regard to the ingenious decisions made by great generals or admirals, but also to conventional and small-scale decisions as well. One of the differences between the performance of artificial commanders and human commanders lies in the ability of humans to mentally simulate a number of potential outcomes of their actions. The following example illustrates this capacity.

A platoon is defending a position with tanks. Enemy tanks are expected to come around a forest corner within firing range. A human platoon commander, seeing one enemy tank, would expect more tanks and therefore would wait longer before opening surprise fire than a simulated commander would. The human commander knows that, if he fires prematurely, he may destroy the lead tank, but the others will be warned and try to outflank him. So he projects, or simulates, forward in time the possible consequences of his actions. Since mental simulation is beyond the present capability of simulated commanders, they may choose a different tactic, leading to a different outcome, unless overridden at particular decision points.

Adding a mental-simulation capability to constructed units will contribute to the enhancement of AI and to overall economy and quality. Why can this be expected? Because, that is how humans think.

During his nearly 20 years of empirical research, Gary Klein investigated the decision making processes of firefighters, pilots, nurses, military leaders, nuclear-power-plant operators, and experts in a range of other domains (Klein, 1999). He developed a model that focuses on human strengths and capabilities that have not been modeled in classical decision theory. In his writing, Klein describes how commanders and leaders

(and experts in general) are often required to make urgent decisions in moments of uncertainty. In this regard, Peter Thunholm also concludes:

> The study of military tactical planning and decision-making has shown that experienced commanders, quite contrary to what is prescribed by traditional military prescriptive planning models, make intuitive decisions based on recognition and mental simulation (Thunholm, 2000).

Susan Hutchins (1996) finds that, in those situations leaders use recognition-based reasoning instead of the classical rational approach. That does not necessarily mean they decide irrationally in the usual sense of the word; rather, they arrive at good decisions by a different path. All three researchers state that they are not discussing a *prescriptive* theory, but a *descriptive* theory, of decision making: that is, a theory of actual human decision making processes.

Recognition-primed decision making (RPD) is an established subfield in the domain of psychology. In the annual conferences since 1998 a large number of applications and advances in the field have been described (NDM, 2005). The attractiveness of the approach and degree of adaptation possible within the military is quite enormous. Klein conducted approximately fifteen studies, funded by the U.S. Army Research Institute, to investigate decision making in a military environment (Klein, 2003). The US Committee on Technology for Future Naval Forces stated in 1997 that the Navy should pursue an approach to joint-model development with a long-term view and an associated emphasis on flexibility. Especially with respect to the technical attributes needed in joint models, decision models should represent the reasoning and behavior of commanders at different levels, naturally reflecting the actions, plans, and adaptations that commanders make in the course of operations (Committee on Technology for Future Naval Forces, 1997). It is thus appropriate that the Naval Studies Board of the National Research Council, Washington D.C., foresaw the advantage of mental simulation, as recommended in 2000:

> The Department of the Navy may need to train commanders in recognizing patterns in typical cases and anomalies encountered in operations to improve their mental simulation skills and enable quicker and better decisions (National Research Council, 2000).

This statement provides a long-term view of the need for modeling mental simulation for future command-decision modeling. Huang (2003) also points out the need for commanders to predict and evaluate future situations. The emergence of huge information resources, especially, requires crucial support of the commander by appropriate C2 systems. In decision-making, the commander pursues information superiority, creates opportunity or risk foresight, and then realizes command superiority. To do so, the commander requires a C2 support system to enhance his situational awareness by presenting him with an explanatory picture and supporting situational assessment, including prediction and evaluation of the future status (Huang, 2003). The development of algorithms for C2 systems is not under review here; but to represent decision-making behavior appropriately it is necessary to represent prediction and evaluation. This research applies the methodologies of event prediction to achieve the capabilities mentioned above. While it is impossible to create a crystal ball that looks into the future with a magic eye and predicts the next event with a probability of 1, there is hope that predictive techniques developed in various domains can be applied to prediction of future events in the simulation, given the observation of the past. The techniques examined in this research have been applied already in reliability analysis, speech recognition and control theory (Aven, 2002; Rabiner, 1989).

In reliability analyses the number of events of type X that occur in a certain period of time are monitored. These events normally represent failures of devices, whether component failures or malfunctioning. The occurrence of a sufficient number of monitored events allows the determination of the parameters from uncertainty distributions, e.g., Poisson distributions. Applying these distributions yields a prediction of the next event with a certain probability (Aven, 2002). For more complicated systems, or where there is a lack of sufficient historical data, we use the Bayesian approach, e.g., Bayesian belief networks. They have a mathematical formalism that allows reasoning during conditions of uncertainty and provides a robust probabilistic framework to evaluate the impact of evidence on uncertain outcomes (Ganesh, 2001).

Hidden Markov models (HMM) are commonly recognized as a state-of-the-art technique in speech recognition. HMMs can be considered as finite-state machines with

known or estimated transition probabilities as well as probabilities for emitting observations from a state. The state of the system is hidden from the observer. Only the observations are "visible." With sophisticated algorithms we can calculate the past transitions that created a certain observation sequence. All these techniques will be detailed in Chapter II. Here we demonstrate that, in many domains, events are predicted that are bases for decisions. In this research the need for a decision will require the system to estimate, that is to provide the probability of the next event. In this research also, prediction of the next event is a crucial part of mental simulation.

Mental simulation in the sense of the Recognition Primed Decision model will contribute to enhanced decision making and make the results of a decision more stable and unsurprising. To illustrate, consider a chess computer. In chess, the number of figures and possible movements is finite—a huge number, but still finite. With current high-end computer power we are still not able to compute the whole search tree in a "reasonable" amount of time. Therefore, the chess computer looks a limited number of plies into the future. A ply in computer chess is a half-move, which is one turn of one of the players: in other words, it is the depth of the search tree the computer looks ahead (Russel & Norvig, 2003). For instance, a player, looking only one ply ahead, cannot avert adversarial moves in time and loses the game. Therefore, it is reasonable to look more than one ply ahead, to decrease the probability of being surprised by a "killer move" of the opponent.

This example shows not only that it is beneficial to mentally simulate, but also that mental simulation is independent of current technology. In a chess game, the player's move assessments are based on relatively simple scoring functions that are applied over and over again. Researchers are hopeful that simple scoring functions will also work for constructed units in the combat-simulation world. The more a system or simulated unit is able to look forward in time, the better its chance of making a sound decision that replicates the reasoning of human commanders.

Mental simulation is still a new field. Though several approaches have been developed (Sokolowski, 2002; Warwick et al., 2001), they all focus on general issues of RPD, not explicitly on mental simulation. By contrast, this research will focus on the mental simulation component of RPD.

C. APPROACH

Since computational approaches to mental simulation are fairly new, few ground-breaking papers have been written. This research will compare various techniques, approaching via the Army simulation model, Combat XXI. This constructive model writes raw data in an externally accessible log-file, and, to apply the different approaches, the data must be preprocessed. In the main part of the analysis, the predicted events will be compared with the actual events.

D. CONTRIBUTIONS

Our representation of human cognition in a decision-making process has not previously been implemented in combat simulation systems. Providing agents the ability to assess possible consequences of their actions can result in proactive, instead of reactive, agents. The main goal here is not to make the agents better than experienced human commanders, but to provide new behavioral aspects that will significantly enhance the agents and to come close in range to human performance. This embellishment could be very valuable for the performance of closed-combat simulation systems, because it would allow us to follow the agents' reasoning process and provide an explanatory component not available before.

1. Contribution Goals

This thesis has five goals:

- it provides the first computational model for mental simulation in a combat simulation environment,
- it provides context and an improved situational awareness to the simulated entities,
- it enables simulated entities to look into the near future and have, therefore, more realistic performance than those that include only knowledge of the present,
- it provides an empirical terrain assessment tool, and

- it provides an explanatory component for the reasoning in terms of losses, time or probabilities. Therefore, the decisions can be explained in a natural human way.

2. Scope

It is beyond the scope of this research to implement and validate all five goals into a current combat simulation model in full detail. However, a partial implementation has been constructed to conduct proof-of-concept experiments.

E. DISSERTATION OVERVIEW

The remainder of this dissertation is organized as follows:

- **Chapter II, Related Work,** describes current research about mental simulation and event prediction in various fields. It describes the goals of developing and implementing mental simulation in the various fields and depicts prediction techniques to show their applicability to this research.
- **Chapter III, Design Considerations for the Model,** describes a detailed view of mental simulation with respect to the specific environment of a combat simulation. It derives the general architecture of the model and generalizes our approach for similar research problems.
- **Chapter IV, Model Implementation,** gives a detailed description of the model used.
- **Chapter V, Experiment and Results,** describes the design of the experiments and their results.
- **Chapter VI, Conclusion and Follow-on Work,** summarizes the contribution made by the thesis and addresses possible future expansions.

THIS PAGE INTENTIONALLY LEFT BLANK

II. RELATED WORK

This chapter provides a survey of current research relevant to this dissertation. It begins with a precise formulation of what mental simulation is and the nature of its relevance to simulated decision-making. The chapter then discusses various prediction techniques that could be used as a predictive component in our model.

A. MENTAL SIMULATION

In this particular context mental simulation is related to decision-making. But, before we define what we mean by mental simulation, it is useful to take a brief look at other interpretations of mental simulation that are related to our topic but not specifically addressed in this research.

The term "mental simulation" is found in system dynamics and software development and is one of four essential measures of program comprehension in software development (Dunsmore, 2000), where it is used as a tool for evaluating pre-built models (with respect to trusting the results from a first simulation run and whether it can be assumed that the modeler has adequately explored the system). In that context, mental simulation forces the modeler to thoroughly understand the reasons behind a model's behavior and helps the developer find problems in computer models (Whelan, 2001). Lebeck (1994) describes mental simulation as similar to asymptotic (e.g., worst-case behavior) analysis of algorithms, which programmers use to study the number of operations executed as a function of input size. There, the program-reference pattern of the underlying cache organization is mentally applied so that the program's cache performance can be predicted. But the term occurs in the development of multi-agent architecture as well. Tambe and Rosenbloom (1996) state the need for multi agent architectures to provide support for flexible and efficient reasoning about other agents' models and enabling mental simulation of their behaviors.

Mental simulation also has a connection to the research area of mental imagery. Mental imagery, often referred to as "seeing in the mind's eye" or "visualization," is a quasi-perceptual experience associated with cognitive functions such as memory, percep-

tion, and thought (Nigel, 2003), but it occurs without the appropriate stimuli for the relevant perception (Finke, 1989). For the interested reader, Finke classified five principles of mental imagery:

1. Implicit stored experience becomes explicit information by mental imaging (implicit encoding).
2. Imagery and perception use similar mechanisms of the cognitive system (perceptual equivalence).
3. Spatial relationships are in images similar to reality (spatial equivalence).
4. The mental rotation of an image is similar to the rotation of real objects (transformational equivalence).
5. The structure of imaged objects is similar to the structure of real objects. Images are coherent and well organized and can be reinterpreted (structural equivalence).

Given the above principles, the recall of mentally stored images is similar to the actual experience of seeing it. The goal of mental imagery is to simulate attributes of an object that have not actually been stored in the memory and cannot just be put together. Kosslyn (1995) made observations about how people answered questions such as "which is bigger: a light bulb or a tennis ball?" People responded by retrieving mental images of the two objects and "looking at them." The similarities between mental simulation and mental imagery lie in the absence of direct stimuli to start the process and precipitate reasoning about what will happen next. In mental imagery, the reasoning uses mental images (Pylyshyn, 2002). Figure 1 shows a well-known example of the mental rotation of 3D-objects in this domain for the purpose of reasoning whether they are identical except for orientation (Shepard & Metzler, 1971).

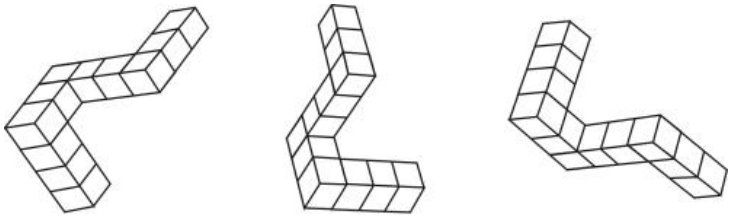


Figure 1. Mental Imagery: To decide whether these objects are identical except for orientation, they are mentally rotated. (adapted from Shepard & Metzler 1971).

Mental simulation in the psychological domain, however, includes areas that are not related specifically to decision-making. Sutton (2000) uses the term "mental simulation" in the context of learning, considering it a tool for making new predictions from old and as the start of a computational theory of knowledge use. Although Sutton uses the term "mental simulation," he focuses on reinforcement learning (which offers possible solutions to the problem of decision-making), but does not apply it to naturalistic decision-making. The simulation (or mental simulation) theory maintains that human beings can use the resources of their own minds to simulate the psychological causes of others' behavior, typically by making decisions within a pretended context (Gordon, 2001). According to the psychologists Davies and Stone (2001), "mental simulation" is the simulation, replication, or re-enacting of aspects of the mental life of another person. Aspects may include, for example, the other person's thinking, decision-making, or emotional responses. So mental simulation seems to offer a methodology for predicting the mental states of other people; and the same methodology can also figure in the practice of attributing mental states on the basis of observed behavior and of explaining behavior in terms of mental states. Mental simulation is also the imitative mental representation of some event or series of events (Taylor and Schneider, 1989). It can be thought of as the cognitive construction of hypothetical scenarios or as a reconstruction of real scenarios. This can include rehearsals of likely future events, fantasizing about less likely future events, realistically re-experiencing past events, or reconstructing past events, mixing in hypothetical elements. Simulation can be used as a heuristic for estimating probabilities or assessing causality (Kahneman and Tversky, 1982). Mental simulation is the process used to serially evaluate actions if course-of-action evaluation is necessary (Dodd et al., 2003).

Dodd brings us back to the present research. As mentioned in Chapter I, mental simulation is an essential part of naturalistic decision-making (NDM). NDM focuses on context and the decision-making process—not just the choice or decision *per se*.

> The study of NDM asks how experienced people, working as individuals or groups in dynamic, uncertain, and often fast-paced environments, identify and assess their situation, make decisions and take actions whose consequences are meaningful to them and to the large organization in which they operate (Zsambok, 1997).

Naturalistic decision-making is mainly explained as "the way people use their experience to make decisions in field settings" (Zsambok, 1997). We consider field settings that refer to real-time situations that can be described as dynamic, time-constrained and uncertain, in which wrong decisions have significant impact on both the individual and the organization (Klein, 1999). The point is that experienced decision makers (among whom we include military commanders) do not employ classical decision theory. The differences from classical decision theory are due mainly to a lack of competing alternatives. In general, experienced decision makers evaluate only a single option but examine different aspects of that option through mental simulation. They make a decision when the option becomes feasible. That does not imply that NDM is always optimal decision-making: important influences are time pressure, high stakes, the contributions of other experienced decision makers, missing or ambiguous information, ill-defined goals, and poorly defined procedures (Klein, 1999). A well-known model within NDM is the Recognition-primed Decision-making Model developed by Klein.

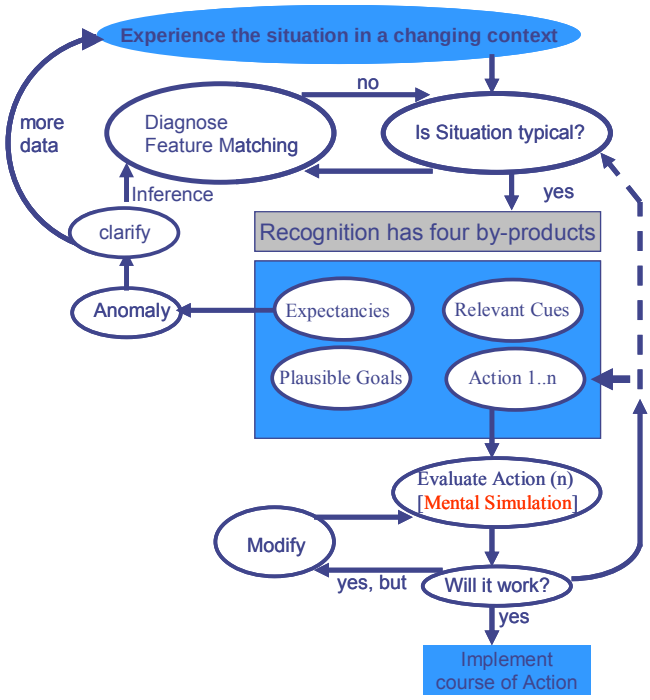


Figure 2. Integrated version of the Recognition-primed Decision-making model. (Graphic from Sources of Power by Gary Klein)

Figure 2 depicts the basic features of the integrated version of the RPD model. It focuses on two processes: first, on how a decision maker sizes up a situation to determine which course of action makes sense; second, on how an experienced decision maker evaluates a course of action.

If the situation is recognized as "typical", decision makers know what type of goals make sense, which cues are relevant and important, what should be expected next, and what are typical ways of responding. Single options for actions to be taken are evaluated by mental simulation. In that context, that means picturing how the course of action will turn out (Klein, 1999).

Our project creates a computational model for mental simulation applied in a combat simulation environment. There are several options for how many steps we can at present predict. The most common method is "iterated prediction": build a single-step predictor and use it recursively. The estimates a model provides are put back into the system as feedback until the desired number of prediction steps is reached. This method of

iterated prediction can be applied to both neural networks (Boné, 2002) and hidden Markov models (Liehr, 1999) (see the next section).

We model mental simulation in the sense of the recognition-primed decision (RPD) developed by Klein, a process of predicting what event can be expected next. Our approach to modeling mental simulation is based on statistical estimation. To construct our model, we monitor events in a scenario, process them, and reach a point at which a prediction could be made.

To date, researchers have made the following attempts to model recognition-primed decision-making.

The work closest to this project was done by Sokolowski (2002). He implemented a model for the recognition-primed decision - making of a Joint Task Force commander in an operational military scenario using a multi agent system approach. With this computational system, Sokolowski could mimic the cognitive process. Figure 3 explicates how the process of deciding according to the RPD model works. The RPD model consists of several components (such as human experience, a recognition process with goals, cues, expectancies, and actions, as well as an action-evaluation and -selection process. To accommodate these components and some engineering issues, the modeler used several agent types. A MainAgent is responsible for system management and the human-computer interface and for establishing and maintaining the experience database. A RecognitionAgent attempts to match the decision request with stored experiences via a table look-up. If there is a match with the experience database, the data is retrieved and made available to other agents.

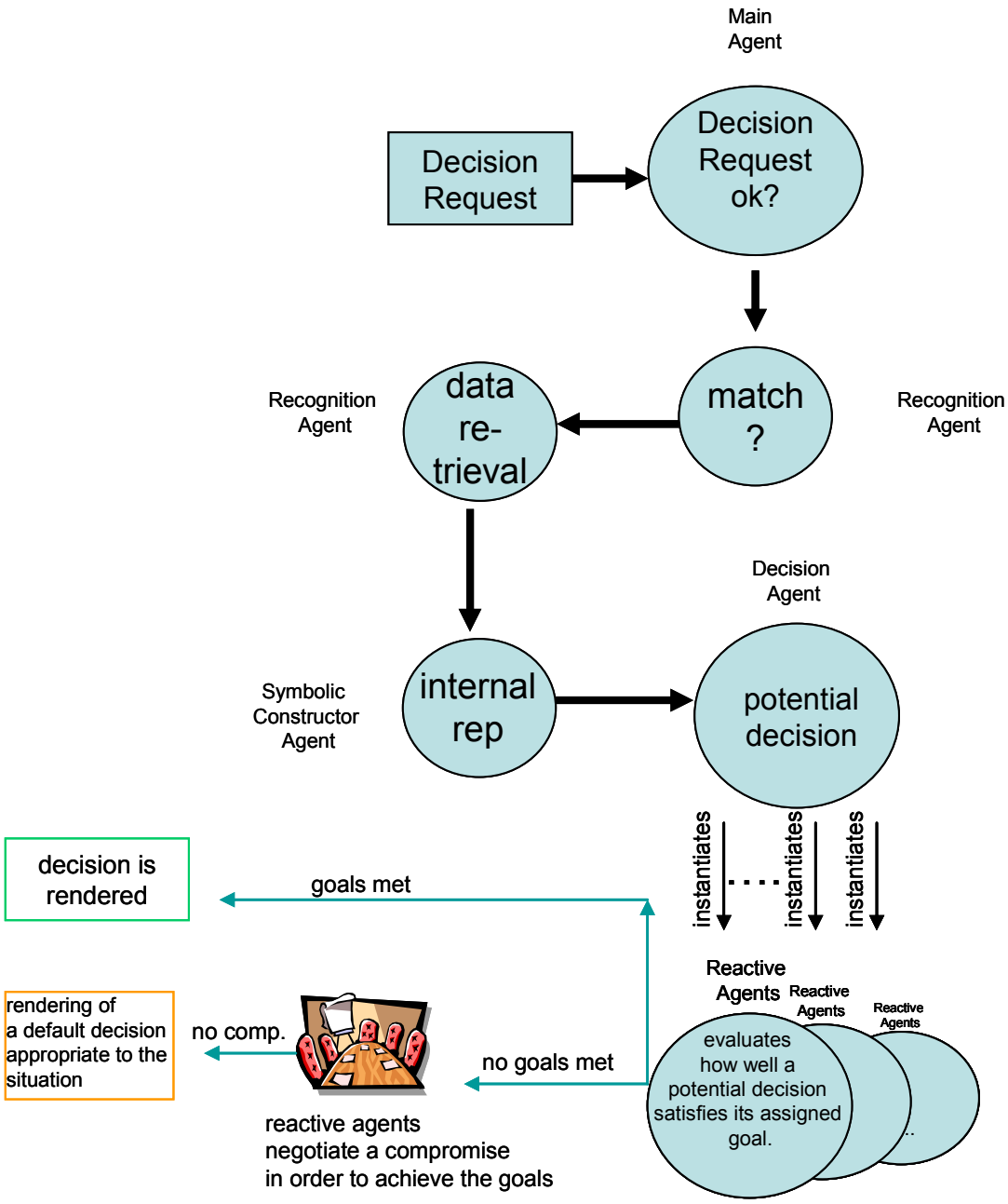


Figure 3. Sokolowski's RPD Model

After retrieving the data, a SymbolicConstructorAgent creates an internal representation of the decision environment. The SymbolicConstructorAgent instantiates a DecisionAgent who looks at the internal representation of the situation and experience. The DecisionAgent surveys the actions available for the current context and chooses the po-

tential decisions that appear most favorable. The second task of the DecisionAgent is to instantiate one ReactiveAgent for each goal associated with the current decision context. A ReactiveAgent evaluates how well assigned goals are satisfied by a certain action. Sokolowski calls this evaluation "mental simulation." To evaluate the degree to which goals are met, he first maps the variables describing the environment into cue values, using fuzzy logic. This yields a categorization of cue values into three categories: satisfactory, marginal, and unsatisfactory. This mapping is essential for quantifying the model's experience. After this, the cue-value category is mapped into a goal-value category. This goal-value category, also obtained by applying fuzzy logic, is an evaluation of a particular action's potential to achieve a goal - a potential based on how well the cues associated with an action favor accomplishment of the goal (Sokolowski, 2003). If all goals are met by all ReactiveAgents, the DecisionAgent renders a decision, based on the action considered. If not all goals are met, a negotiation phase is added. Agent negotiation (Sprinkle et al., 2000) is the method use to resolve the goal-achieving conflict. Sokolowski (2002) also stated that agent negotiation best represents how a human decision maker uses mental simulation to arrive at a compromise among multiple conflicting goals within his mind (Minsky, 1986). If this negotiation is successful, a decision can be rendered; otherwise, the next-best action is evaluated. This goes on until either a satisfactory action is found or all possibilities are exhausted. If the latter, a default decision is rendered. Sokolowski (2002) also stated "The mental-simulation process will most likely need to be enhanced to better replicate the role of mental simulation within RPD."

Warwick et al., (2001) approached their modeling of RPD by encoding the long-term memory (LTM) of decision makers. They modeled LTM in a data structure by storing individual decision-making experiences as a two-dimensional array. When new situations occur, they are compared with experiences stored in the LTM. Computing a "similarity value" yields a measure of comparability in order to recognize a usable experience and the appropriate course of action. Although it seems to show promise as a model of parts of RPD, the mental simulation part has yet to be designed (Warwick, 2002).

B. PREDICTION TECHNIQUES

Prediction in this context is far from a Nostradamus-like predictions about envisioning future events based on intuition or "higher" insight. We focus on *quantitative*

prediction: that is, statistical reasoning over time. A common task is to predict future events given a sequence of observations over a period of time. We also focus on discrete event prediction: that is, we consider discrete event prediction as the modeling of a system as it evolves over time, where the system can change at only a countable number of points in time (Law & Kelton, 1991). Our goal is to estimate the next event based on statistical methods.

Figure 4 shows the world of dynamic systems divided into finite (i.e., discrete) and continuous state spaces. By Continuous Dynamics, we mean that if the update equation for the dynamic system is $x(t+1) = f(x(t))$, f is a continuous function. It is hard to imagine a linear dynamic system with a finite state space. In the discrete case, we only consider nonlinear models like Hidden Markov Models or Dynamic Bayesian Networks. In the nonlinear case of a continuous state space, there exist many Time-Series prediction models, for example, Neural Networks. Box-Jenkins models (ARMA in Figure 4) are commonly applied when the update function is linear (Box and Jenkins, 1994). Kalman Filters can be extended to a part of the nonlinear dynamics with the Extended Kalman Filter.

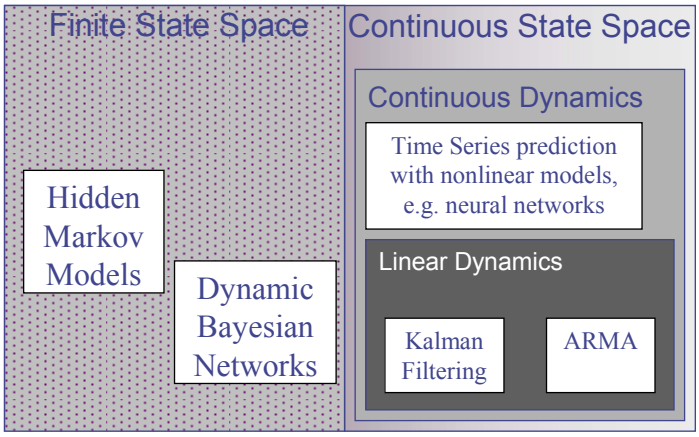


Figure 4. Finite vs. Continuous State Space. The white boxes show models that are applicable within the respective domain.

The question is what kind of state space and what kind of dynamics can we expect in a combat simulation environment? We can expect a finite set of entities, for example, weapons, vehicles, platoons, and other units we have to deal with. In this case, we have discrete quantitative variables. We also expect variables like artillery impact, river cross-

ings, movement, positions, etc., where then we have categorical variables. The extent to which we are dealing with nominal (unordered) or ordinal (ordered) variables will be discussed in the next chapter.

The greatest part of the system will behave nonlinear; however, a linear model might be applicable for some subset of data. The Box-Jenkins approach is basically a combination of an autoregressive (AR) model and a moving average (MA) model. The autoregressive model is a linear regression of the current value of the series against one or more prior values of the series, while the moving average model is a linear regression of the current value of the series against the white noise or random shocks of one or more prior values of the series (NIST/SEMATECH, 2004). While Box-Jenkins models forecast the future values of an observed time-series, we do not consider events as numerical values, even when they are sometimes coded as numbers. We expect many variables to be categorical. An engagement of a tank with anti-tank missiles cannot be added numerically to an observed river crossing of an artillery platoon. Therefore, we disregard linear continuous-valued time-series predictions like the Box-Jenkins Model. However, we will at least consider Kalman Filters, because they are well suited for motion tracking in a multidimensional space.

First, we try to simplify the environment and use a Poisson Process to predict the next event. We are well aware that the "real world" is much more complicated. Therefore, the Poisson Process serves as a strawman and will be replaced later by a more substantial solution. However, it is always possible that for some data category this might be a suitable method. The main focus in this chapter is on models that have the capability of learning through pattern or character recognition. Especially, we look at Kalman Filtering, Neural Networks, Markov chains, Hidden Markov Models, and Dynamic Bayesian Networks.

1. Poisson Process

A Poisson process is an integer-valued non decreasing stochastic process, characterized by its rate function $\lambda(t)$, which describes the expected number of "events" or "arrivals" that occur per unit time. There are homogeneous and nonhomogeneous Poisson processes.

The homogeneous Poisson process has a constant arrival rate $\lambda(t) = \lambda$, and the marginal distribution N(a) is Poisson distributed with parameter λa, where a denotes the time interval 0 to a in which arrivals occur. To qualify as a Poisson process the following conditions have to be true (Ross, 1993):

- No event at time t=0;
- The process has independent increments, which means the number of events which occur in disjoint time intervals are independent; and
- The process has stationary increments, meaning that the distribution of the number of events that occur in any interval of time depends only on the length of the interval.

A very simple approach to prediction would be to consider events of a certain type as a Poisson process. Random single-type events occur at a certain rate. These events 1...n are observed and the time they occur is recorded, for example, $T_1, T_2 \ldots T_n$. The objective is to predict T_{n+1}, or better, to give a point estimate when the next event n+1 will occur.

We assume for a first and simplified approach that the events occur according to a Poisson process with rate λ. This assumption implies that the interarrival times can be considered as independent, identically distributed, exponential random variables.

After a certain number of events the parameter of the underlying distribution is estimated via the maximum likelihood method.

The maximum likelihood estimator for λ is: $\hat{\lambda} = \frac{n}{\sum t_i}$ where

n = number of events observed,

t_i = interarrival time between event i and i-1.

An estimate for the next mean interarrival time $\hat{E}(T_{n+1})$ will then be $1/\hat{\lambda}$.

The expected number of arrivals in time interval t is $\hat{\lambda}\,t$.

Furthermore, we can calculate the probability that at time t we have observed n events:

$$P\{N(t) = n\} = e^{-\hat{\lambda}t} \frac{(\hat{\lambda}t)^n}{n!}.$$

Time period	arrival time	interarrival time
t_1	0.3	0.3
t_2	0.8	0.5
t_3	1.2	0.4
t_4	1.5	0.3
t_5	2.2	0.7
t_6	2.6	0.4
t_7	3.1	0.5

Table 1. Example for a Homogeneous Poisson Process

Based on the data in Table 1, using MLE for λ yields: $\hat{\lambda} = \frac{n}{\sum t_i} = \frac{7}{3.1} = 2.258$

Mean time to the event n+1: $E(T_{n+1}) = (n+1)/\hat{\lambda} = 0.44*8 = 3.54$

In a simplified algorithm, this would progress as follows:

1. Monitor the arrival times.
2. Determine the interarrival times.
3. When sufficient arrivals have occurred, estimate lambda.
4. Predict the next event.
5. After event has occurred, update the lambda estimate.

We can expand this model to encompass multiple event types. Now we consider that, at each given arrival of the Poisson process, an independent trial is performed that classifies the event as type 1 with probability p or as type 2 with probability 1-p. Then:

M (t) = number of type 1 events to occur in (0,t), and

N (t) = number of type 2 events to occur in (0,t)

are independent Poisson processes with rate λp and λ(1-p), respectively.

If we have observed twice as many type-1events, we can assume that the probability of an event type 1 is 2/3, given an arrival.

The expected number of type-1arrivals within a certain time interval 0..t will then be $\hat{\lambda}pt$. If we now have k possible type of events i = 1, 2, …., k, then if an event occurs at time y, it will be classified as a type I event with probability p_i, i = 1, 2, … , k where $\sum_{i=1}^{k} p_i = 1$. The expected number of type-i events is calculated by $\hat{\lambda}p_i t$, if the probability is independent of the arrival time.

If, on the other hand, the probability depends on time of arrival y, the expected number of type-i events is calculated with

$$E[N_i(t)] = \lambda \int_0^t p_i(s)ds \text{, where} \sum_{i=1}^{k} p_i(y) = 1.$$

This approach requires that all possible events be "pre-categorized." The event with the shortest expected arrival time is taken as the prediction. In a combat simulation environment, this would be a simplistic approach. It is obvious that a Poisson process with its assumptions can cover only a small portion of the entire complexity.

2. Kalman Filtering

Kalman filtering is a method for recursively estimating an unknown state of a dynamic system, in which the measurements have noise. In other words, it describes a method for updating an estimate of a system's state by processing measurements. The basic Kalman filter developed by R. Kalman in the early 1960s is a linear model. The following equations demonstrate how the state of the system can be estimated and corrected after the measurement has been processed. A discrete-time controlled process is governed by the linear difference equation

$x_k = Ax_{k-1} + w_{k-1}$ with a measurement $z_k = Hx_k + v_k$,

where x_k : the state to estimate at time step k.

A: transition matrix; relates the state at the time step k-1 to the state at the current time step k.

w_{k-1}: movement noise at time step k-1.

H: measurement matrix; relates the state to the measurement z_k.

v_k: measurement noise at time step k.

Assuming that the random variables w and v are independent and normal distributed with mean 0 and variance Q and R, respectively, then the equation will be for the time update of the state $\hat{x}^{-}_{k} = A\hat{x}_{k-1}$, where $^{-}$ means the *a priori* prediction of the state, with the error covariance $P_k^- = AP_{k-1}A^T + Q$. The measurement update with the measurement z_k will yield for the state estimate $\hat{x}_k = \hat{x}^{-}_{k} + K_k(z_k - H\hat{x}^{-}_{k})$. The error covariance is update as well and computed with $P_k = (I - K_k H)P_k^-$, where K is the Kalman gain: $K_k = P_k H^T (HP_k^- H^T + R)^{-1}$.

There is also an extension to the Kalman filter when the measurement is a non linear function of the state variables. The measurement matrix H is then obtained by linearization of the nonlinear function.

Kalman filters are ideally suited for tracking the motion of an object in a multidimensional space.

3. Neural Networks

Neural networks – that is artificial neural networks - model the function of the human brain. They consist of many hundreds of artificial neurons, or nodes, which are simple processing connected by directed links. Each of these artificial units is a simplified model of a real cell in the brain (Figure 5). An artificial neuron sends off ("fires") a new signal when it gets a sufficiently strong input signal from the nodes to which it is connected (Russel & Norvig, 2003).

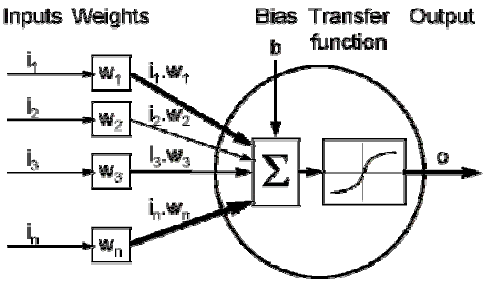


Figure 5. A simple mathematical model for a neuron

The strength of the input signal and the specific firing threshold of the neuron result in the ability to perform different tasks, corresponding to different patterns of node-firing activity. The strength of the connection between neurons is represented by connection weights. They serve as the collective memory of the neural network (Mitchell, 1997). For a unit to produce output, it first has to compute the weighted sum of inputs in_i with the equation

$$in_i = \sum_{j=0}^{n} W_{j,i} a_j \ .$$

By applying an activation function g, the output is derived by

$$a_i = g(in_i) = g\Big(\sum_{j=0}^{n} W_{j,i} a_j \Big).$$

Each unit in a neural network can have different activation functions. Typical activation functions can be either threshold functions, or sigmoid functions, also called logistic functions. The input values for a unit (artificial neuron) can either be continuous between -1 and 1 or discrete with values -1, 0 or 1. For accurate Boolean functions, we need to use a hard-step activation function like sgn(x). For fuzzy versions of Boolean functions, we can use a logistic function like $f(x) = 1/(1+e^{-x})$.

The network has to be trained after setup to yield proper output, usually by using training data consisting of input and associated output. Both the initial output and the error to the desired output are calculated. The error is propagated backward through the network and the weights of the connections are adjusted. This procedure is repeated until the error at the end is in an acceptable range (Mitchell, 1997).

The advantage of neural networks is their ability to take incomplete or noisy data while producing an output similar to one obtained from perfect input data. In this manner, a neural network can provide satisfactory decisions, based on the uncertain and highly dynamic conditions that exist in a complex war scenario (Sokolowski, 2003). Neural networks are well suited for recognizing underlying patterns in data (Mitchell, 1997). However, a significant amount of training data would be required to properly prepare a network to recognize situations over the entire domain of joint warfare. In an attempt to model RPD with neural networks, twelve military experts and twelve scenarios were used

to define the goals and plans to learn from (Liang, 2001). Another disadvantage of neural networks is the difficulty of interpreting results.

In addition to their application for robot control and handwriting recognition, a major application area for neural networks is the reliability domain. Reliability theory is a body of ideas, mathematical models, and methods directed at predicting, estimating, understanding, and optimizing the lifespan distribution of systems and their components (adapted from Barlow et al., 1965). In reliability theory a prediction capability is very important for estimating when a critical component will fail. But reliability theory is not limited to "hardware components"; it can be applied also to software. In critical applications, it is often important to be able to make accurate predictions about the mean-time-to-failure (MTTF) of software, because a failure can be considered a lack of stability. Philips Research at India-Bangalore (Patra, 2003) applied a neural network approach to MTTF-prediction. With the neural network, the parameters of the formal model were estimated and the neural network itself learned the process to predict outcomes. Using a feed-forward network and back-error propagation yielded successful predictions that outperformed parametric models, such as the nonhomogeneous Poisson process (Patra, 2003).

4. Markov Chains

A Markov chain is a finite state machine with probabilities for each transition, i.e. the probability that the next state is s_i, given that the current state is s_j (NIST, 2005). The state space can be either continuous or discrete. If the state space is continuous, the random process may take on any value over a continuous interval or set of intervals. If it is discrete, there is a finite or countable number of states. Many publications refer to a discrete-state random process as a "chain." A Markov chain is indexed by time: the time indicates the state change. In a discrete-time Markov chain, the state changes are preordained to occur only at the integer points 0, 1, …, n, which map to the time points t_0, t_1, … , t_n. In a continuous-time Markov chain state changes may occur anywhere in time. Therefore, it is possible to have a discrete-state space and a continuous time Markov chain. That corresponding process is referred as a continuous-time Markov chain (Bose,

2001). It has the Markovian property that the conditional distribution of the future X(t+s) given the present X(s) and the past X(u), $0 \leq u \leq s$, depends only on the present and not on the past.

5. Hidden Markov Models (HMM)

In a hidden Markov model, or HMM, the sequence of observations $\{Y_T\}$ is modeled with the assumption that each observation depends on a discrete hidden state. Furthermore, we assume that the sequences of the hidden states satisfy the Markov assumption. The Markov assumption means that the transition to the next state depends only on the current state and not on previous ones. We want to consider a sequence of random variables that are not completely independent, but the value of each variable depends on the previous elements in the sequence. We have a simple Markov-process model when we can observe the states the system is in. For example, we have two urns with black and red balls. The first urn has twice as many red balls as black. The second urn has an equal number of each color. Both urns may have the same number of total balls. The system is said to be "in state one" when the balls are drawn from the first urn and in state two when the balls are drawn from the second urn. Hence, we can assign each event, or color of the drawn ball, to the state n, or urn 1 or 2. We can also assign a probability that a certain color ball was drawn from a certain urn.

If we do not see the urn, that is, the state and are only told the event, a ball with color x has been drawn, then we have a hidden Markov model. The event "black ball" can happen now in either state one or two. An HMM is considered a statistical model for any system that can be represented as a succession of transitions between a finite set of discrete states, where the next event depends only on the previous event. An HMM is a probabilistic function of a Markov process, whereby we do not see the state sequence the model is going through but we know, or estimate, the probabilistic function of the state sequence. Therefore, HMMs can be described as stochastic finite-state automata with an underlying stochastic process that is *not* observable, that is hidden, but that can be observed through another set of stochastic processes that produce the sequence of observed symbols (Rabiner & Juang, 1986).

These observations 1..n yield an observation sequence O. Furthermore, we assume that the observations occur due to transitions between internal hidden states, in con-

junction with the random emission of a sequence element, for example, observation symbol v. Markov models are, in their original form, not very flexible in the prediction of dwell time in a state, although researchers have attempted to model the dwell time in a state. Ferguson (1980), extended an HMM to include a probability distribution for the duration of each state and therefore allowed a sequence of observations to be made while the system remained in a single state. The following introduction shows a basic case of the HMM; the modeling of the duration will be explained in Chapter III.

Hidden Markov models are characterized by the following qualities:

1. N is the number of states in the model; states are denoted as S1, S2, … S_N, where the individual state at time t is q_t.
2. M is the number of distinct observation symbols per state j. The individual observation symbols are denoted $V = \{v_1, v_2, \ldots v_M\}$.
3. There is a matrix A for the state transition probability distribution; a_{ij} is the transition probability from state i to state j where $a_{ij} = P\{q_{t+1} = S_j | q_t = S_i\}$ and $1 \le i, j \le N$.
4. There is a probability distribution matrix $B = \{b_j(k)\}$ for the possible observations, where $b_j(k) = P\{v_k \text{ at } t | q_t = S_j\}$, $1 \le j \le N$ and $1 \le k \le M$.
5. The initial state distribution is $\pi_i = P\{q_1 = S_i\}$, where $1 \le i \le N$.

N and M are specified by B implicitly. Therefore, a hidden Markov Model can be described as a triple $\lambda = (A, B, \pi)$.

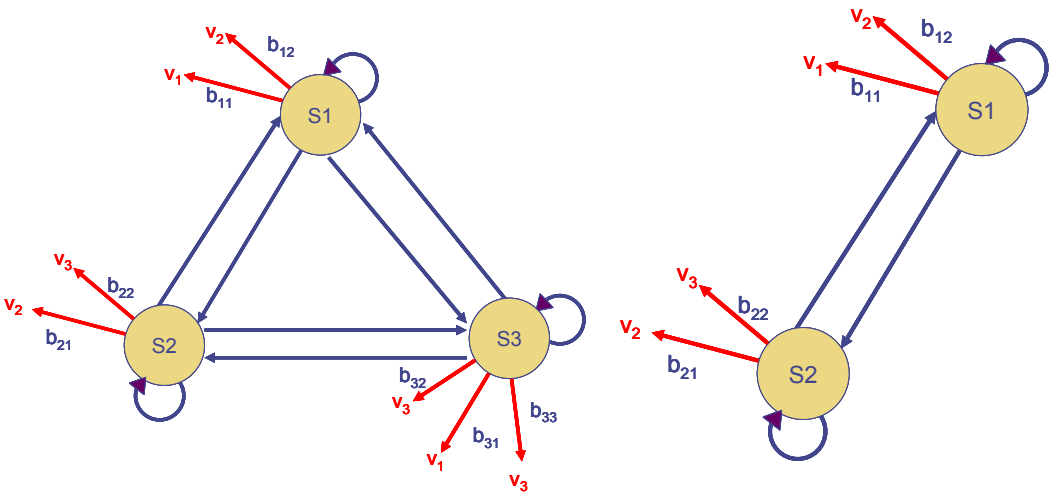


Figure 6. Example of HHM for the Urn Problem

The transition diagrams in Figure 6 show possible models of the simple urn problem mentioned above. It is given that we are only told the event and do not know what the underlying model is, and there may exist more than one model.

The HMM theory distinguishes now between three problems known as:

1. Learning
2. Decoding
3. Evaluation

We talk about "learning" of an HMM when we try to determine the model parameters λ. We know a coarse structure of the model that allows us to know the number of states; we do not know the transition probabilities between the states and the occurrence probabilities of the observations per state. In the urn problem above, we know there are two urns from which balls are drawn. The Baum-Welch algorithm was developed to solve the learning problem.

The term "decoding" denotes a process to find the most likely sequence of state transitions that lead to the observed and known sequence (Viterbi-Algorithm).

If we have a known model, we have a complete transition matrix and probabilities for the observations. What we are interested in is determining the probability that a certain sequence will occur: we call this process "Evaluation." Another contributing factor is that we are also interested in the probability that a certain model created this sequence.

This is especially interesting when several competing models exist, as in Figure 6, except we do not know whether there are three urns or only two.

Computational aspects:

1. Baum - Welch Algorithm (BW)

To use BW we must have an observation sequence O and a coarse structure of the model. We want to find the values of the model parameters lambda that best explain what we have observed. We will use maximum-likelihood estimation, that is we want to find the values that maximize P(O|lambda), which can be written: $\arg\max_{\lambda} P(O_{training} \mid \lambda)$. There is no analytical method to choose lambda to maximize P(O|lambda). However, the BW algorithm, also known as the forward-backward algorithm, maximizes P(O|lambda) by applying iterative hill-climbing algorithms. The first step is to use an initial model that can be either pre selected according to rules or chosen by random. The observed sequence is then run through the initial model, which yields an expectation of each model parameter. After updating the model parameter, the observed sequence is run again through the model. Baum proved that the property $P(O|\bar{\lambda}) \geq P(O|\lambda)$ holds, where λ is the initial model and $\bar{\lambda}$ is the model after a learning step (Manning & Schütze, 1999). The iteration is done when there is no longer any significant improvement. This solution does not guarantee finding a global optimum, but in practice the re-estimation is usually effective (Manning & Schütze, 1999).

2. Viterbi Algorithm

The evaluation of a model can be solved exactly with the forward procedure. However, we are more interested in finding the most likely state sequence associated with the observation sequence Q. The Viterbi algorithm provides a computationally efficient way of analyzing observations of HMMs to recapture the most likely underlying state sequence. It exploits recursion to reduce the computational load and uses the context of the entire sequence to make judgments.

The Viterbi algorithm computes the most likely complete path by maximizing $\arg\max_{X} P(X \mid O, \lambda)$, where X is the vector of the most probable visited states. It is

sufficient for a fixed O to maximize $\arg\max_X P(X, O \mid \lambda)$ (Manning & Schütze, 1999). A common representation of all the possible transition sequences that can be obtained is a trellis. A trellis is a layered graph whose vertices represent possible observations in the corresponding state (Figure 7). The vertices of the trellis can be embedded in a two-dimensional matrix with the vertices in each layer assigned to elements in the corresponding column of the matrix (Bouchaffra et al., 1996).

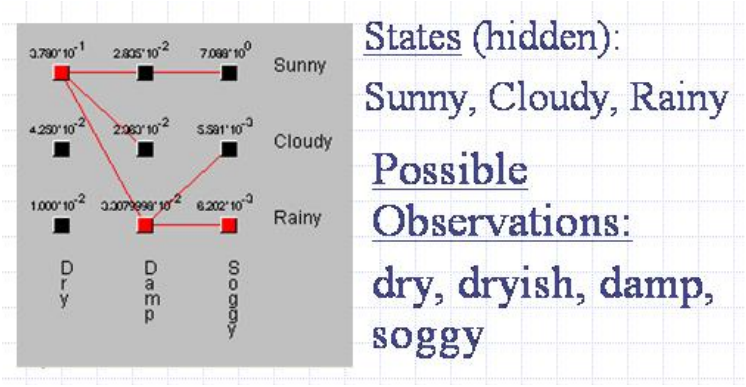


Figure 7. Example of a trellis in a Viterbi Decoder (Image taken from University of Leeds, 2004)

The variable $\delta_j(t)$ stores, for each point in the trellis, the probability of the most probable path that leads to that node $\delta_j(t) = \max_{X_1 \cdots X_{t-1}} P(X_1 \cdots X_{t-1}, o_1 \cdots o_{t-1}, X_t = j \mid \lambda)$.

The corresponding variable $\psi_j(t)$ then records the node of the incoming arc that led to this most probable path. Using dynamic programming, we calculate the most probable path through the whole trellis as follows:

1. Initialization: $\delta_j(t) = \pi_j$, where $1 \le j \le N$
2. Induction: $\delta_j(t+1) = \max_{1 \le i \le N} \delta_i(t) a_{ij} b_{ij}$, where $1 \le j \le N$

   the back trace is then stored: $\Psi_j(t+1) = \arg\max_{1 \le i \le N} \delta_i(t) a_{ij} b_{ijo_t}$, where $1 \le j \le N$
3. Termination and readout of the path are then done by backtracking.

$$\hat{X}_{T+1} = \arg\max_{1 \le i \le N} \delta_i(T+1)$$

$$\hat{X}_t = \Psi_{\hat{X}_{t+1}}(t+1)$$

$$P(\hat{X}_t) = \max_{1 \le i \le N} \delta_i(T+1)$$

With $P(\hat{X}_t)$ we know the probability of the best path so far producing the next observation at the next time step. This reflects the transition probability and allows us to predict what the next probable observation would be.

6. Dynamic Bayesian Networks

A dynamic Bayesian network (DBN) belongs to the group of "graphical models." A graphical model is a graph that represents certain properties about sets of random variables. The nodes in the graph correspond to random variables; the edges encode a set of conditional independence properties (Bilmes & Zweig, 2002). DBNs are directed graphical models of stochastic processes and they allow the modeling of discrete-time processes as they evolve over time. DBNs are an extension of Bayesian networks that unfold over time or gets time-sliced. The term "dynamic, " as in "dynamic Bayesian network," means that a dynamic system is modeled; it does not mean that the graph structure changes over time.

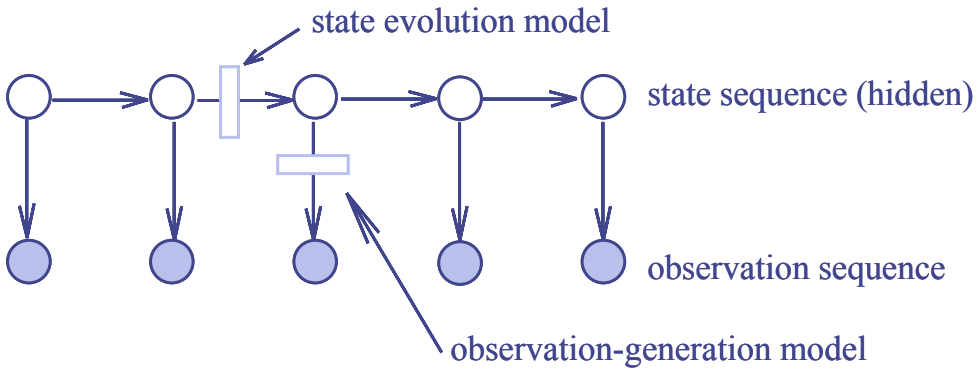


Figure 8. A Dynamic Bayesian Network

To represent causality in an approximate way, the graphs have directed edges, which means that a variable i connected to a variable j has a causal influence on it DBNs assume that the state variables are Markovian and stationary (Deviren, 2001). "Markovian" in this context means that the set of state variables in interval k depends only on the set of state variables in the interval k-1. This time dependence determines the parents

of the current node (Figure 8) and with the stationary property, the network structure is time-invariant, which means the network structure does not change over time. DBNs generalize hidden Markov models by representing the hidden (and observed) state in terms of state variables, which can have complex interdependencies. The graphical structure provides an easy way to specify these conditional independencies, and hence, to provide a compact parameterization of the model (Murphy, 2002). Figure 9 depicts a simple example of an observation sequence in a DBN.

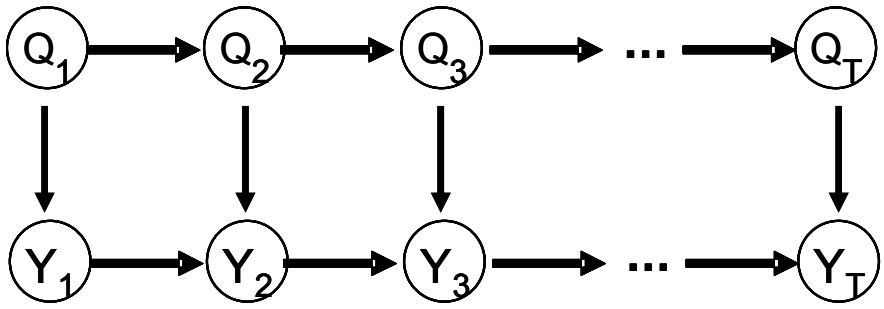


Figure 9. Observation Sequence Y_1 to Y_T in a DBN

We can calculate the probability of this particular observation sequence by $P(Y_1,Y_2,\ldots Y_T)=P(Y_1)P(Y_2|Y_1)\ldots P(Y_T|Y_{T-1})$. Given that we know the model and have observed Y_T, we can predict the value of Y_{T+1}. The temporal order of the graph is important, because it specifies the direction of causality (Gharamani, 1997).

7. Various other Approaches

Event prediction problems address issues similar as time-series prediction problems. An event sequence is a sequence of time-stamped observations that are described by a fixed set of features.

AT&T Labs has been especially interested in predicting failures of telecommunication equipment based on logs of alarm messages (Weiss & Hirsh, 1998a). In that domain, they are interested in predicting a specific event within a window of time, a so-called rare event, from the time-stamped observations. Since statistical methods require numerical features, classical time-series prediction techniques are not applicable (Brockwell & Davis, 1996; Weiss, 1999). Weiss uses a genetic algorithm that searches directly for predictive patterns in the data. In his case, the event-prediction problem was formu-

lated and solved as a machine-learning problem. The situation we are interested in, however, is different. We are interested in *what* the next event will be, not *when* a specific event will occur.

System management is another domain in which event prediction in temporal sequences plays an important role. The ability to predict rare events that are harmful in a production network can be helpful in automatically detecting real-time problems (Domeniconi et al., 2002). For example, a computer network is assumed to be under continuous monitoring. The monitoring process produces event sequences in which each observation has a fixed set of categorical and numerical features. For Domeniconi, the event consisted of four components, one of which addressed the severity of the failure. The severity was ranked in five steps: harmless, warning, minor, critical, and fatal. Prediction focused on events in which the severity was either critical or fatal, much like the telecommunication case explained earlier. However, instead of a genetic algorithm approach, Domeniconi formulated the problem as a classification problem. Applying the means of singular-value decomposition — a powerful set of techniques dealing with sets of equations or matrices that are either singular or numerically very close to singular — provided numerical answers to the prediction problem.

For content providers and consumers pre-fetching web pages has considerable value. The prediction of the user's next request for a desired content improves downloading time. Many techniques have been considered for this purpose. Davison (2002) implemented machine-learning techniques in order to predict the next user action on the Web.

8. Predictive Control Theory

The prediction techniques presented so far are methods for modeling dynamic systems. This brief section demonstrates that predicting possible next events and estimating how a system will behave in the near future, are not merely academic questions. In process industries, it is crucial to predict system dynamics. In chemical-processing industries, especially, model-based predictive control is currently the most popular advanced-control theory. And in other industries as well — power plants, petroleum refineries, food processing, automotive, and aerospace — predictive control can be found. A common characteristic is that an explicitly formulated process model is used to predict and opti-

mize future behavior (Hovd, 2004). The main term used is “model predictive control” (MPC), one of the most popular control techniques in industry. Figure 10 shows a schematic diagram of a model-based predictive controller. The basic principles are that the internal model, here the plant model, is known, and predicts the future output of the system. The reference signal is predicted for a finite number of steps into the future and, depending on this prediction, the control for “adjusting” the system is calculated, returning as feedback into the system (plant).

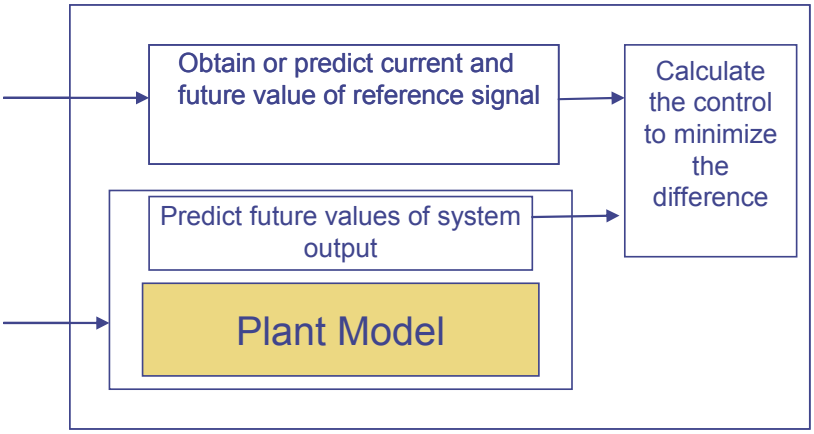


Figure 10. Example of a Model based predictive controller (adapted from Ordys)

This technique is not applicable to linear systems alone. Recently, due to more and more constraints, such as environmental and safety considerations, and to process-immanent non linearities, the attention to nonlinear-model predictive control has increased (Findeisen & Allgoewer, 2002). An overview can be found in Qin (1996) and Hovd (2004).

THIS PAGE INTENTIONALLY LEFT BLANK

III. THE MENTAL SIMULATION MODEL (ARCHITECTURE)

A. INTRODUCTION

This chapter begins with an overview of the major requirements that bound the architecture of the model. It covers in detail the requirements from the psychological theory of naturalistic decision-making as the outer framework for mental simulation. We consider it important to develop the model in accordance with the current state of knowledge of naturalistic decision-making. We also discuss the framework in which the test bed, Combat XXI, fits. In addition, we describe the main application domains for combat simulation models and their current critical deficiencies. Some of the deficiencies can be resolved by mental simulation as an essential part of naturalistic decision-making. The developed architecture is explained in detail. The chapter ends with an overview of how this work might be applied to domains other than the ones considered in this work.

B. MENTAL SIMULATION

1. Uses of Mental Simulation, in Detail

A general overview of mental simulation has already been outlined in Chapter II. In the following sections mental simulation will be described in detail and relevance as it is used in NDM.

Mental simulation is an essential part of RPD. In the recognitional decision-making it is used in

(a) diagnosing to form situation awareness,

(b) generating expectancies to help verify situation awareness, and

(c) evaluating a course of action.

These map well with Endsley's definition of situation awareness: "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" (Endsley, 1995). He specifies three hierarchical phases or levels: perception, comprehension, and projection into the future.

Perception, as a phase or level of situation awareness, identifies the key elements that define a decision-making situation. In our context, that means the different formations of tanks and the corresponding number of tanks. Comprehension, as a phase or level of situation awareness, means understanding the current decision-making situation in tactical terms, such as: attacking in direction x, at speed y, supported by z. The third phase or level of situation awareness, projection into the future, means anticipating the predicted or expected evolution of the current situation. In our context, this could mean, for example, foreseeing a necessity for reinforcements or adjustments in the resource allocations.

How might now a predictive model be used to support these three levels of awareness? The answer lies in this concept: mental simulation enables us to explain observed events, actions as described by Klein being one type of observable event.

(a) Diagnosing to form situation awareness

If we understand the inherent events, we can diagnose a decision-making situation: we have a picture in which things fit together. If we do not understand the events, we cannot explain the situation: our situational awareness is insufficient. In mental simulation, according to Klein, we match features of the events to the perceived situation. The following example may clarify what we mean here. Assume a combat situation in which a tank is in a defensive position. Although, in reality, the tank's defensive position could indicate any number of platoon situations, for the sake of clarification, we choose just two. In one situation, the tank's position means the platoon is defending itself against an enemy's main attack. In the other situation, it is not. For each situation there is a corresponding predictive model, or data structure, that maintains the platoon's knowledge of past experiences and the parameters of the current situation. Although the simulated platoon may not be aware what kind of situation it's in, it can make certain sequential observations. To form a diagnosis of the situation, we could run the observation sequence in different models, where the best match would give a reasonable estimate of the platoon's situation. If we used an HMM as a predictive model, we would run the Viterbi algorithm. The results would show the state sequence that is most probable to explain the observation sequence. That could then be mapped to a corresponding model. Therefore, in sum,

we argue that a predictive model can be used to diagnose situation awareness. Chapter IV will show the results of our implementation.

(b) Generating expectancies, to help verify situation awareness

A predictive model can also support the generation of expectancies, to help verify situation awareness. Taking our example and using comprehension, Endsley's second level of situation awareness, we reach the following conclusions. We know that a certain formation has x number of tanks and is moving in a particular direction at a certain speed. Predicting the next event, that is, seeing the number of tanks that we expected to see, would help confirm our picture of the situation as compared to the actual event. However, if we suddenly see twice as many tanks as expected, and we have no idea how this happened, it would cause us to reconsider our picture of the situation. So, we generate an expectancy, and then see whether or not that prediction confirms our assumption about situation awareness.

(c) Evaluating a course of action

This aspect of mental simulation, or situation awareness, is the most intuitive of the three. If you mention mental simulation to a layman that is what they will immediately think of. Chapter IV will demonstrate how the possible actions of our simulated platoon can be projected into and evaluated in the future.

Situation awareness is critical for making intelligent decisions. Without it there is no context for adapting one's behavior to accommodate the current and future state of the world (Klein, 2000). According to Endsley, situation awareness is more than just knowledge of numerous pieces of data. It also requires having an advanced understanding of a situation and some projection into the future, based on the user's goals (Endsley, 1995). Understanding a situation requires the mental integration of many pieces of information. Mental, or intelligence, simulation is the means to achieve that in virtual worlds. This requires knowing both that the information exists and how it interrelates with other pieces of information in the situational context. People usually know when something is occur

ring in which they are involved, or that a particular piece of information that they need exists. But they do not always understand how these relate in the overall larger context (Albers, 1999).

2. Key Points of Mental Simulation

Klein identified the following key points of mental simulation through many years of empirical research in the decision-making field (Klein, 1999):

- "Mental simulation lets us explain how events have moved from the past into the present." In this work we do not try to explain how events moved into the present, but we use the events in the past and present in order to predict events into the future.
- "Mental simulation lets us project how the present will move into the future." This is the key point we model in this research.
- "Constructing a mental simulation involves forming an action sequence in which one state of affairs is transformed into another."
- "Because of memory limitations, people usually construct mental simulations using around three variables around six transitions." The number of variables we use in this research matches this usual behavior.
- "It takes a fair amount of experience to construct a useful mental simulation." This is considered and applied in our research.
- "Mental simulations can run into trouble when the situation becomes too complicated or when time pressure, noise, or other factors interfere." This point is not addressed in this work.
- "Mental simulation can be misleading when a person argues away evidence that challenges the interpretation." In this research we do not consider this point.
- "There are methods for improving mental simulation, such as using crystal ball and premortem strategies and decision scenarios." This point is beyond the scope of this thesis.

3. Application of Klein's Model

Klein has developed a generic conceptual model of mental simulation. He begins with two types of need: First, to explain the past, and second, to project the future. Figure 11 depicts a generic model of mental simulation developed by Klein. The parameters for the mental simulation process depend on the type of need. To explain the past the initial state is considered; to project into the future, the terminal state is the relevant one. Klein derived empirically that people usually use three mental states and about six transitions. This result will be exploited later. In our research we start with mental simulation from the current state and evaluate possible actions to arrive at a terminal state. According to Klein the projection into the future has two purposes: The one purpose is to predict what is about to happen and to take the appropriate measure in order to be prepared. The other purpose is to observe a potential sequence of actions and to determine whether there exist flaws that lead to rejection of this action sequence.

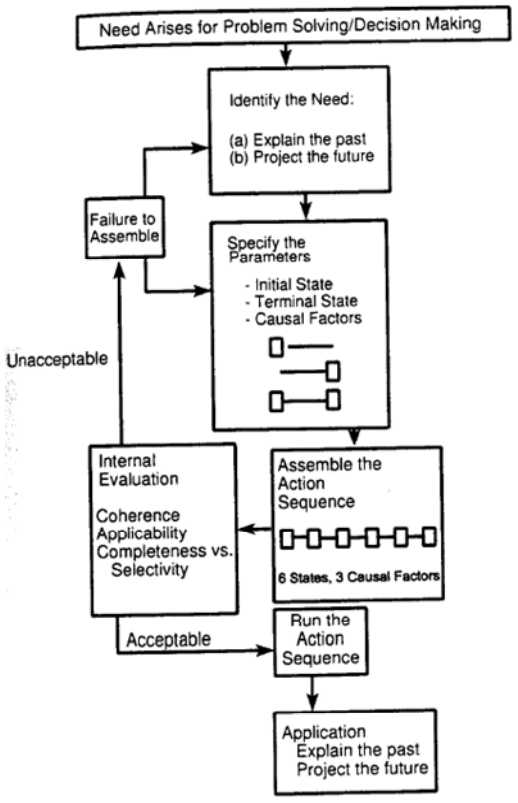


Figure 11. A generic model for mental simulation adapted from Klein, 1999.

Having determined an action sequence, we can evaluate it internally for coherence, applicability, and completeness. If the action sequence passes our internal evaluation, the action sequence is run to generate a model of explanation or projection, whichever is needed. If the action sequence fails, we must start over and reconsider the parameters. As Klein points out, mental simulation is not always successful, but when it works it is impressive.

4. Mental Simulation for Projection into the Future

As mentioned above, there are two types of needs for mental simulation: to explain the past and to project the future. For details about the need to explain the past, the reader is referred to the literature. Only a graphical model is displayed, in Figure 12.

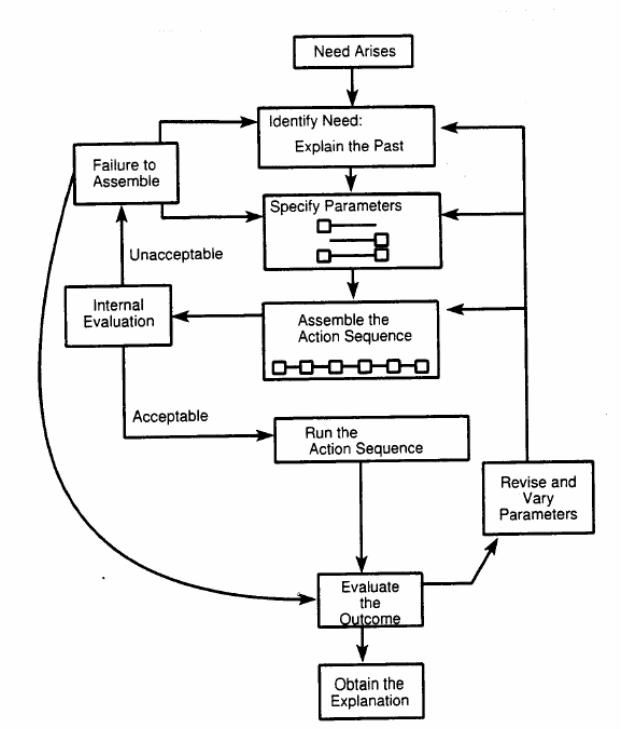


Figure 12. Using mental simulation to explain the past, adapted from Klein, 1999.

The main focus in this research is projection into the future. Figure 13 displays the details of the model graphically. The left side of Figure 13 shows the conceptual model using mental simulation to project into the future, adapted from Klein. This is an extension to Figure 11. In the application of projecting the future the outcome of the ac-

tion sequence is evaluated. The identification of problem areas with respect to plausibility, consistency, or pitfalls during the run and review of the action sequence can require a micro-simulation of the problem areas identified. The problem areas will also be evaluated. The evaluation has three possible results. Firstly, the action sequence looks feasible and the implementation of the action selected starts. Secondly, the course of action is under no circumstances feasible and is rejected. Thirdly, the action sequence still might work, but it has to be modified an re-evaluated.

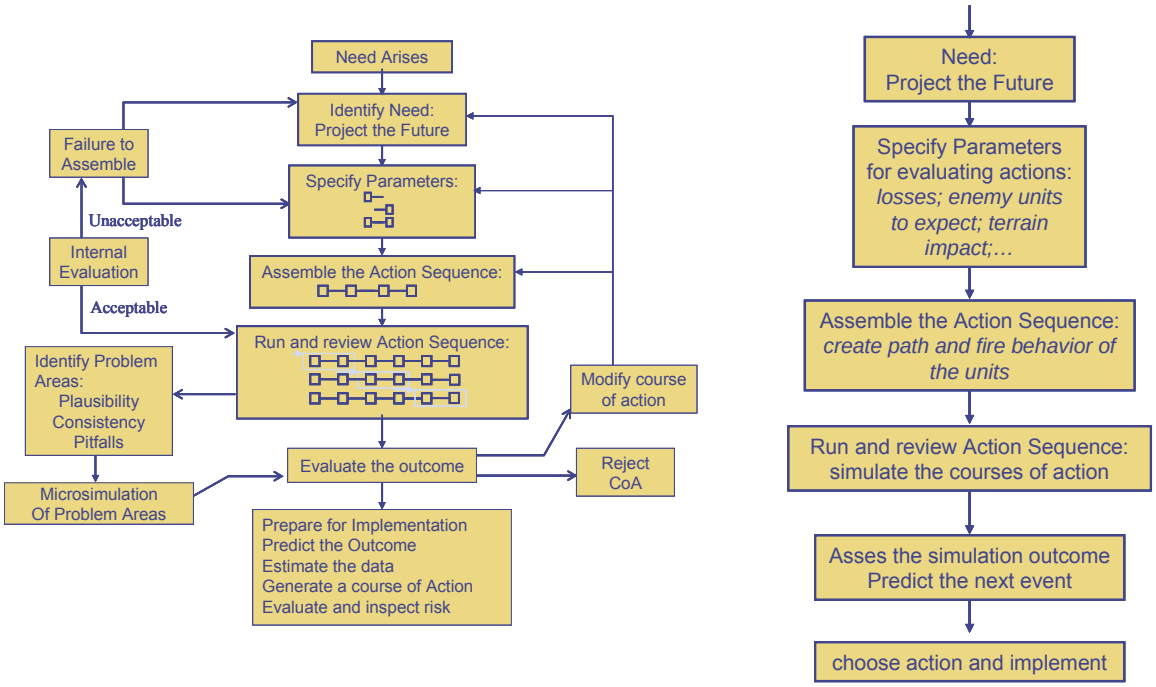


Figure 13. Left: Using mental simulation to project into the future, adapted from Klein, 1999. Right: The Adaptation of Klein's model in our research.

In our application of Klein's model, we already determined the need for projecting into the future. We specify the parameters losses of our own and the opposing forces, the expectation of what to see next, and how the prediction is evaluated with respect to the environment in which it takes place, in our case, terrain. We also assemble an action sequence, which contains possible paths of outflanking or certain firing behavior. The simulation of the possible action is assessed. If the first action is promising then we choose it. In case of ambiguity we add other parameters into consideration. Details will follow in the next chapter.

C. COMBAT MODELING AND COMBAT SIMULATION MODELS

Computer-based military simulations have been used since World War II (Hausrath, 1971). Today, nearly all armies of the Western Hemisphere use modeling and simulation (M&S) as essential tools for analysis and for training their leaders and war fighters. The U.S. DoD Defense Modeling and Simulation Office (DMSO) characterizes M&S tools according to their class (i.e., live, virtual, or constructive), the functional area they support, and the level of detail or fidelity the simulation system contains (DMSO, 2005). Combat simulation models are most commonly classified as constructive simulations. They are analytical models, ranging from detailed engineering models to highly aggregated theater/campaign simulations, in which the performance and/or behavior of components, entities, systems, or collections of systems are represented as a function of time and environmental stimuli (DMSO, 2005). "Constructive" in this sense, means that the playing units and the environment are constructed or synthetic (NATO, 1998). Constructive simulation systems may run slower than real time, at real time, or faster than real time, depending on the particular use or function of the simulation. In contrast, "virtual" simulations involve real weapon systems and operators in synthetic environments, that allow the operators to interface with real equipment and to train in realistic three-dimensional battle spaces. Virtual simulations, in general run in real time in order to evaluate the operators' or the systems' responses to actions. "Live" simulations use real hardware/equipment and troops within a real or realistic environment. What is simulated is mainly the weapons effects. Constructive simulations are now being used more for after-action reviews or for forces that do not actually participate in a given exercise, i.e., adjacent units.

In addition to the modeling and simulation classifications, models are characterized by their scope and level of detail. M&S resources are categorized as engineering, engagement, mission, or theater/campaign resources, as shown in the M&S hierarchy illustrated in Figure 14.

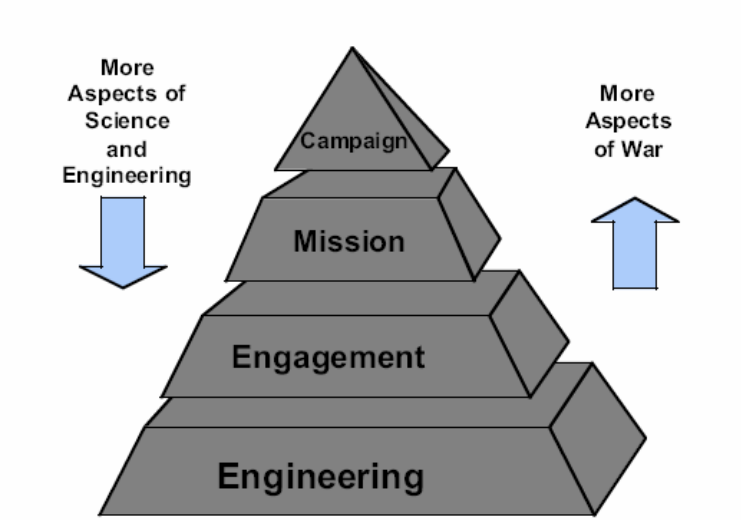


Figure 14. Model and simulation hierarchy. Adapted from Modeling and Simulation Information Analysis Center (MSIAC, 2005).

> Models at the bottom of the hierarchy are more detailed but involve few components and/or systems, whereas models higher in the hierarchy include more players, more aspects of warfare, and simulate longer durations, but the players become more aggregated and the physics is represented more implicitly (DMSO, 2005).

However, there are additional models in the M&S tool box that do not fit the above characterization, for example, environmental representations, threat models, and logistics models. They are not further considered here. This research focuses on constructive simulation models.

The U.S. military uses constructive simulation in the following application domains: advanced concepts and requirements; military operations; research, development, and acquisition; and training (DoD, 1995). The North Atlantic Treaty Organization (NATO) defines the following application domains: defense planning, training, exercises, support to operations, research, technology development and armaments acquisition. However, NATO also notes that individual nations may use different taxonomies to classify M&S application areas (NATO, 1998). The following considerations focus on two main areas of combat simulations. The one is the use in training including exercises; and the other major use is in analysis. The level to which these domains benefit from combat simulation models varies and is driven mainly by the amount of AI required and the

amount available for a specific use. Andrew Ilachinski (2004) discusses another categorization: automation. He divides the combat-simulation-model world according to the types of forces used: semi automated forces and automated forces. In the semi automated category, human agents make many of the tactical decisions, to ensure that the automated units' behavior conforms to expectations and is realistic. Training and exercise applications, especially, use semi automated models. In contrast, the automated-forces category does not involve any use of human agents; instead, it is the simulation model itself that specifies the automated entities' actions. Automated-forces models are referred to as "closed" simulations.

Simulated, or constructed military entities mimic the behavior of real-world units mainly in terms of their physical actions, including troop movement, target detection, target selection, and engagements on a weapon's firing level. Higher-level command functions are either scripted or modeled with common AI-techniques such as case-based reasoning or expert systems (Ilachinsky, 2004). However, even in cutting-edge models, behavior in general is still not at a satisfactory level. In this respect, the following statement from the NATO Modeling and Simulation Master Plan (1995) is still largely valid.

> Constructive simulations are better at representing systems than representing human behavior. For example, a simulation may accurately represent the direct fire effects from weapon systems engaged in a simulated force-on-force event but cannot represent the decision process of the operational commander employing that force.

Some of the U.S.-Department of Defense (1995) requirements concerning human behavior modeling assert that

> the representation of humans in models and simulations is extremely limited, particularly in the representation of opposing forces and their doctrine and tactics. In view of the limited theoretical underpinnings in this area, this issue will require extensive research before human behavior can be modeled authoritatively.

The Modeling and Simulation Resource Repository (MSRR) provides an excellent overview of current simulation models. The MSRR is a DoD-wide system of modeling and simulation databases that allows a user to discover, access, and obtain M&S resources that support military operations, training, and acquisition (MSRR, 2005). Inara

Kuck (2003) summarizes the capabilities and limitations of nearly all modeling and simulation programs that are sponsored by the DoD. We give only a brief description of the model, Combined Arms and Support Task Force Evaluation Model (CASTFOREM), because this is currently the U.S. Army's highest-resolution, combined-arms combat simulation model. CASTFOREM is used for the evaluation of weapon systems and tactics in brigade - and - below combined arms conflicts. It uses closed-form mathematical expressions, probability distributions, and an embedded expert system (Ilachinsky, 2004; MSRR, 2005). It will be replaced by the Combined Arms Analysis Tool for the XXIst Century (COMBAT XXI), which will be explained in detail in the next section.

D. COMBAT XXI AS TEST BED

1. General Description

Combined Arms Analysis Tool for the 21st Century (COMBAT XXI) is a high-resolution, closed-form, stochastic, analytical combat simulation. COMBAT XXI is being developed by the TRADOC Analysis Center – White Sands Missile Range (TRAC-WSMR) and the Marine Corps Combat Development Command (MCCDC). COMBAT XXI will be used for the analysis of land and amphibious warfare in the Research, Development and Acquisition (RDA) and Advanced Concepts and Requirements (ACR) Modeling and Simulation (M&S) domains.

COMBAT XXI is a replacement for Combined Arms and Task Force Evaluation Model (CASTFOREM). The COMBAT XXI model capitalizes on many of the "time tested" algorithms of CASTFOREM, while providing enhanced capabilities. Many underlying algorithms in the COMBAT XXI model, such as acquisition and engagement algorithms, are derived from CASTFOREM. Enhanced capabilities include the capability to easily compose new combat platforms and tactical units, enhanced scenario building tools with a graphical user interface, and modular architecture for separation of physical modeling and behaviors.

COMBAT XXI is intended to support analytical needs in the ACR domain, including force design, operational requirements, mission area analysis and war fighting experiments. COMBAT XXI is also designed to support force-on-force analysis related to the RDA domain that includes weapons system development, and test and evaluation. COMBAT XXI represents joint combined-arms operations (including ground warfare, aviation operations and amphibious operations) on a tactical level.

One Semi-Automated Forces (OneSAF) and COMBAT XXI are related in a manner similar to Janus and CASTFOREM. Janus is used in areas where

human-in-the-loop war gaming is used to refine scenarios and define the range of major decisions OneSAF is being developed as a human-in-the-loop simulation (with resolution from individual platforms to battalion level units) to support all M&S domains. COMBAT XXI will produce replication data sets (generated by numerous stochastic runs of various scenarios) for combat analysis at the brigade and below level for ACR and RDA M&S domains. COMBAT XXI models tactical (brigade and below) scenarios (CXXI-User's Guide, 2005).

2. Behavior Representation

The degree to which a combat simulation model represents the real world situation depends not only on the resolution of the modeled forces but also on the behavior of the units when performing tactical tasks or missions. Behavior and decision-making go hand in hand. The behavior is normally the result of a certain decision made in relevant time to the action. It is not necessarily always important how the simulated units came up with the decision, it is more important to see the impact of the decision as appropriate and tactically correct behavior. In other words, we are not going to model how the brain works, but rather utilize the factors and parameters humans consider in their decision-making process and then exploit the strength of a computer in order to come up with a decision.

Behavior in CXXI can be distinguished in physical algorithms, primitive behavior, such as movement, search and engagement, and tactical behavior, such as bounding over-watch, close-air support, etc. Basic tactical behavior and decision-making is represented in decision-making modules. Additional, situation and scenario dependent behavior can be authored by assigning behavioral rules to entities. These behavioral rules have access to required data during the run and make use of the entity decision-making capabilities. Figure 15 displays the basic structure of a rule.

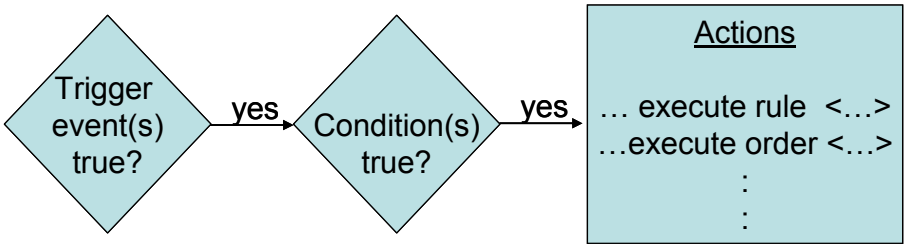


Figure 15. Basic structure for rules in Combat XXI

Trigger events start the entire process of considering a rule to be evaluated for execution. This can be the event that a certain control measure has been met, that certain modules are initialized that are effects on entities, or that…. The next step depends on the type and amount of conditions that are met. In order to actually execute an action all conditions have to be met and the rules and the orders under the actions have to exist. Further options include the selection of an echelon that defines what sub-units will receive a particular rule. It is also possible to repeat the execution of rule. However, for all further executions the entire set of trigger events and conditions required have to be met again.

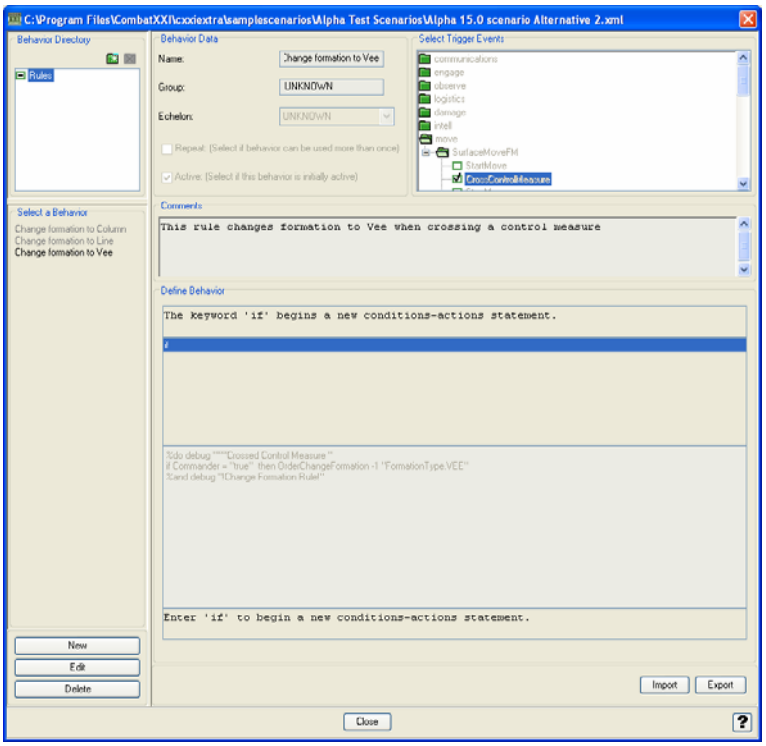


Figure 16. The rule editor template in CXXI for creating behavioral rules that can be assigned to entities.

Figure 16 displays the rule template editor for creating, modifying or deletion of user defined rules. Figure 17 shows an example of a rule body in Combat XXI. The Rule Body defines what rule executes when the trigger event occurs. Rules can be composed. This means that the user can build more complex rules by using a set of simple rules. This feature is exploited in the use of Combat XXI in order to create behavior that is not initially available in the decision modules. However, the features created in this research

are not within the current rule capabilities and according to the development team it is not likely that the infrastructure will provide these features in the near future.

EXAMPLE RULE	
Rule Body	if Commander = "true"
	and AtObjective = "false"
	and DistanceToObjective <= 1500
	and HasCurrentObjective = "true"
	then OrderOccupyObjective

Figure 17. Example Rule to illustrate how a rule looks. Adapted from the Combat XXI User's Guide.

3. Scenario Output

The output of the scenario is written to a set of log-files. Each default log file contains data regarding the type of the log file. A movement log contains data lines for simulation time, entity name, coordinates, speed, azimuth and pitch. An acquisition log contains basically who saw whom where, with what sensor, and with what accuracy. The model developed uses acquisition, movement, and engagement log files as input. The output will be discussed in detail in chapter IV. The model is event driven. That means that only output is provided when events have happened. If a sensor does not detect a target then this is not an event and, therefore, no output is logged.

4. Run Manager

Other features of CXXI utilized in this research is the Run Manager. This tool allows the user to execute multiple replications of a scenario run. The functionalities cover the number of replications, the duration of a single scenario run, the type of random number generator, which data loggers to create and given a network, what computers to use.

5. Summary

Aggregated, Combat XXI provides a capability to run a ground-force-based scenario repeatably with full detection and engagement functionality where the units follow

scripted actions. Rules allow the user to invoke special behavior depending on trigger events and conditions met. The output is written in text-files.

E. GENERAL MODEL ARCHITECTURE

The most general application of the model developed is depicted in Figure 18. The entire system consists of four components: the environment, which covers mainly the simulation system, the situational awareness component, the mental simulator, which predicts and assesses, and the decision component which evaluates the influencing factors and actually renders the decision. Figure 19 gives a more detailed view of the components of Figure 18.

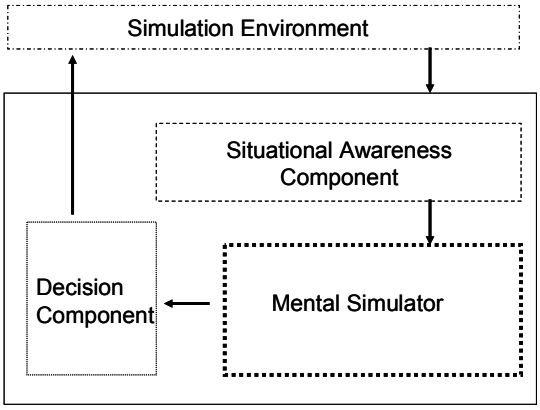


Figure 18. The components used in the model developed

1. Simulation Environment Component

The simulation environment is the driving component. It contains the simulation system that can run on the same computer or can be networked. For the general use it does not matter, as long as the output of the system contains the required data for the other components. That sounds totally obvious, however this is not always the case and the simulation system might need to be adjusted. One occurrence of that case could be related to the way detections are handled in a combat simulation system. The target acquisition algorithms yield detections of entities, but in contrast to a human observer on the battlefield they normally do not provide the event when a spotted unit goes out of sight. This can only be deduced when in the next observation-sweep the specific entity does not show up on the "detection list" any more. But then it is still unknown at what

specific time and at what specific location this occurred. Another consideration could be the case in which aggregated units are used. The attrition of aggregated units is normally computed by Lanchester Equations. The target detection and acquisition does not provide information about individual tanks. There exist combat simulation models where the resolution is not on the entity level, like in Vector in Commander (VIC, 2005). That does not exclude aggregated models from being used in this research. However, the decision-making process will not be more detailed than the model's resolution level.

2. Situational Awareness Component

The situational awareness component represents the internal model of the external world. The external world in this context is the current situation in the combat simulation environment. Many mind theorists have used the term ”internal model of the world” with respect to intelligent adaptive behavior. This expression has not been used uniformly. Rich Sutton and Andrew Barto give an excellent summary in (Sutton and Barto, 1981). They state:

> For some, an internal model is a general knowledge store capable of answering any sort of question about the world. For others, an internal model is much more limited in that it can answer only a single question: “What should be done next?” In the first case another part of the mind can ask the internal model many questions before taking action, whereas in the second the internal model generates a recommendation for action only in response to the immediate situation.

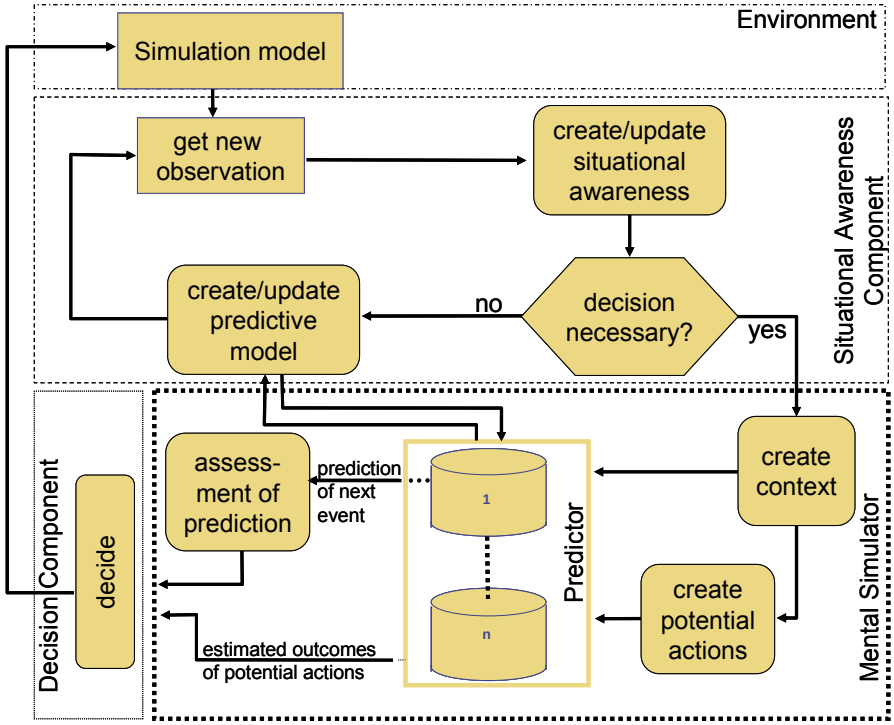


Figure 19. The general architecture of the model implemented.

We follow more the second case mentioned. However, we also consider this internal model in terms of situational awareness of a commander, in the sense of Endsley's level 1, discussed earlier in Chapter B 1. Therefore, the situational awareness component takes the output of the simulation and builds up its own internal perception of the world. It creates estimates about the enemy formations, speed and directions. In case the mental simulator creates its predictive model parallel to the situation development the percepts/observations are also send to the mental simulator. In case there is a predictive model pre-loaded an update might only occur. In this abstract view the situational awareness component is not limited to ground combat situations alone. It is applicable to all cases, where a more sophisticated awareness is required than in the simulation system available. This might include appropriate knowledge in a 3D-environment about the value, benefit, or meaning that "people" seen in the VR have due to their spatial relationship. This might mean to know that I can watch a certain portion of a building and others see a different portion, but overall I know what portion of the building in total can be surveyed. The situational awareness component reads the output file from the simulation

model until a decision is required. A decision might be any action the mental simulator can be of assistance. In our case this might be the decision to fire the weapon or to hold the fire, even given the resources.

3. Mental Simulator Component

The mental simulator component is activated when a decision is required. It creates the appropriate context for the decision which is the basis for developing potential actions. The predictor module in the mental simulator has two possible inputs. One is the context of the decision, i.e., what is the current situation and based on that what can be expected next. The other is the set of potential actions which might be simulated in time within the predictor module or might be a table look-up in a representative data-base. The prediction of the next event is assessed with respect to the context of the decision and in our special case with respect to the terrain where it is supposed to happen. Both, the assessment of the prediction and the estimated outcomes of the potential actions are input for the decision component. The central part of the mental simulator is the predictor module. It contains the predictive model(s) and a capability to either store previous simulations and then look up the results or to create a simulation on the fly and evaluate potential actions.

4. Decision Component

The decision component processes the assessed prediction of the next event most likely to happen and the results of the simulated courses of potential actions. It will also incorporate the terrain impact on the prediction. When the decision has been rendered, the result will fed back to the environment.

F. GENERALIZATION OF THE MODEL

In current simulation systems, even among those currently under development, decisions are based on mechanical behavior, similar to stimulus-response-theory. It is like a shooting gallery: a duck pops up and gets shot at. Entities do not anticipate future events. However, we can add new information into the simulation system with sensors, or equivalent methods, in combination with a predictive model. By doing this, we can cope with counterfactual events. Counterfactuals are events that have not happened yet, but may happen, on a probabilistic basis. This is equivalent to a human imagination what might happen - used in human decision making. This anticipation is generally called

imagination - Klein calls it mental simulation. Using a probabilistic approach, in our case this is a Markov Chain, but could be some other method. A Markov Chain can provide a probability for a transition into the next state and therefore, give an estimate about a future event. This enables having a computational method for coping with imagination. This approach is not totally random; it is governed by line of sight and by experience in the past. We consider mental simulation as anticipation of counterfactual events in a way that allows them to influence behavior. Our implementation demonstrates how to open up a simulation and use probabilistic approaches to imitate human decision making that is based on concepts, counterfactuals and imagination.

The approach to mental simulation may be extended to a wide range of simulations and models. An example shows how the methodology developed here might be applied to a different problem.

This example relates the following numbers to the corresponding scheme in Figure 20. Imagine, for instance, that a need (❶) arises for improving the behavior of simulated humans in a virtual world, a need that might arise in a simple shooting trainer or in a more complex environment with multiple immersed players and avatars who must move appropriately in a threatening situation. The need might be based on requirements for more sophisticated behavior of avatars, because there inadequate behavior can distract trainees. Of course, similar circumstances also occur in constructive simulations. When instances of inappropriate behavior occur in a simulated environment, we ask: What has not been considered yet and why? Which leads to a second question: What decisions do humans make that simulated entities are not designed to make? Considering those questions is the first step when building a mental simulator for decision-making situations (❶). In such simulations, the entities must recognize decision-making situations. Our first task then is to determine what relevant information is needed. In our research, both questions are relevant and they are:

What event has to be predicted? (❷)

What decision-influencing factors should be made available? (❸)

Granted, providing the availability of knowledge of influencing factors might be more complicated than a simple table-search. It might mean hard computational effort. For that, we turn to the field of computer science, which uses the term "instrumenting" for program testing. "Instrumenting a program" (❻) means augmenting it with program code that measures specific aspects of the program (Pedersen, 1999). The necessary additional information might already be generated somewhere in the code, although it is not yet displayed. That is an easy case: as in a debugger, the information just has to be retrieved. The harder case would require adding code that generates additional information, which may require several runs of variations of the current scenario in a parallel, or separate simulator, or a feasibility consideration of alternatives that were not needed thus far.

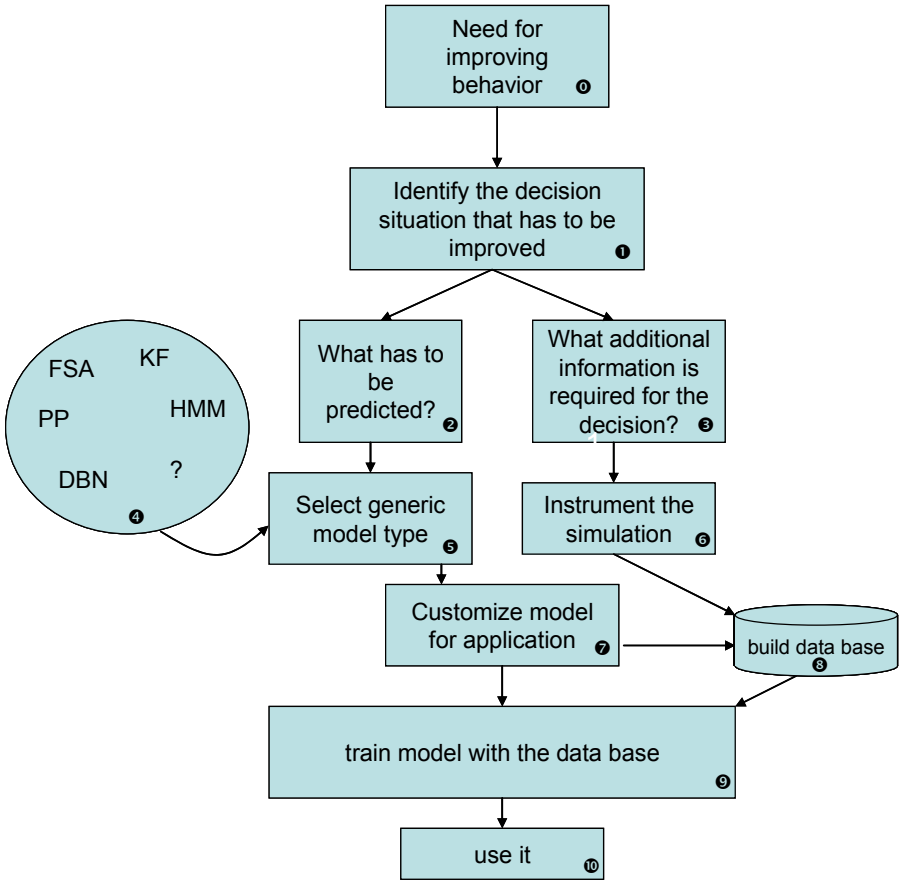


Figure 20. A general sequence of the modeling and improving process.

When appropriate and feasible, a database can be built that provides fast, easy answers in certain decision-making situations (❽). Consider this situation, for example: a

simulated brother-in-arms in an urban-terrain environment sees a foe disappearing around a corner. With that awareness, his behavior in following and moving around the corner—the speed, what body part goes first, his use of a mirror to peek around the corner before he exposes himself to potential enemy fire—is markedly different than if he were simply walking his dog. Such situations can be parameterized and pre-simulated so that the avatar realizes that, in x out of y cases, the results are z. This information can then trigger more realistic behavior, for which this research may provide some guidance.

The event to be predicted determines to some extent which predictive model is best to use (❺). Chapter II gives an overview of some statistical predictive models (❹). The purpose of a predictive model is to provide an estimate of what is possible. However, statistical models can provide probabilities of events based on prior observation of those events, or they can predict novel events that are composed of previously observed events. Both cases, distribution of observed events and distribution of unseen events that are composed of observed events, can be useful in different context. A predictive model has to be customized and implemented for a given purpose (❼). For a case in which a finite-state machine is used, it is important to determine the states and to clarify what the states mean. If the states are chosen poorly, then the state space might explode and be computationally hard to handle. To give qualitatively sufficient predictions, the Markov Chain must be based on a representative data set. For instance, if you have ten states and only five arcs, there may be a prediction gap. For a decision-making situation requiring information about the motion of a target in a multidimensional space, a Kalman filtering might be used. If something else is needed, another model may be a better fit. The final step before applying and using a model is to "train" it (❾). In this context, "training a model" means estimating its parameters to maximize the probability of a set of strings being generated by the model (Pedersen, 1999).

To make this discussion more concrete we will guide the reader through an additional example which will show how the methodology developed in this research has a broader impact. When watching movies with bad and good guys there occur chasing scenes. When a character, being chased, disappears behind a corner, there is more than one option for the pursuer when they reach the corner. In some cases the pursuer goes

around the corner at full speed. In other cases caution is used and the pursuer carefully peeks around the corner in order to avoid being ambushed after the turn. If we were to simulate this chase in a training simulation, it would be important for us to give the pursuer believable behavior. Forsythe (2004) discusses this issue when talking about the use of simulation technology for Law Enforcement. He states:

> Many current simulations, as well as computer games, incorporate human entities and allow participants to interact with those entities. It might seem that the ability for trainees to gain experience in a law enforcement role already exists. Many people are concerned that the synthetic humans used to populate most current simulations do not provide a sufficient level of behavioral realism. For many years, within the simulation and computer-gaming industry, researchers have placed a heavy emphasis on accurately modeling the characteristics of equipment and providing a high degree of realism in computer graphics, sound, and other sensory experiences. Substantially less emphasis has been placed on the behavioral realism of simulated humans. In many cases, synthetic humans have been provided simplistic and predictable behavioral routines that are highly susceptible to gaming (i.e., once the behavioral routine is recognized, players exploit this knowledge of the underlying software to their advantage).

This is similar to the domain of constructive simulation. He also states the main effort that has to be pursued:

> The key development in simulation technology that benefits the law enforcement profession involves the ability to interact in a natural manner with highly realistic and diverse simulated humans. These capabilities are not yet available.

Deriving from the above we state that there is room for enhancing human behavior in virtual environments for Law Enforcement. Models for law enforcement personnel training must provide an ability to interact in a natural way with diverse and highly realistic simulated humans (Forsythe, 2004). The synthetic entities must process cues and interpret them through a humanlike decision-making process that is consistent with human reasoning. Such capabilities are not yet available to a satisfactory degree. However, if the simulations are provided with the ability to create their own awareness of the situations encountered, the "reasoning" behavior of the synthetic entities will be much improved. For instance: If the entities have knowledge, say, about policemen's location, they can anticipate the danger and will behave differently when rounding a corner or entering a

building. The entities will become more situation-dependent and act more realistically, which will enhance the training simulation. We will now reconsider the chasing example in a comprehensive way and explain it in reference to Figure 20.

A simulated brother-in-arms in an urban-terrain environment pursues a foe disappearing around a corner. He is always racing around the corner. This is not always situational depending appropriate behavior. So, there exist a need for improving behavior (1). The next step (2) would be to identify, what has to be improved. In the particular case it would be the behavior when going around the corner. In order to determine what additional information is required, it might be that additional cues or knowledge elements are incorporated (3). This could be to record how long they have been running versus a mean value adults can run at high speed without taking breathing time. It is also possible to categorize the environment in terms of favoring the one or the other behavior. The instrumenting of the simulation (6) would cover making the required information available. This might be relatively easy or hard. This cannot be assessed in this general discussion. However, these types of situations can be parameterized and pre-simulated so that the avatar realizes that, in x out of y cases, the results are z. This knowledge can be stored in a data base (9).

So far it should be known that an essential part of improving behavior is by adding expectations to the simulation. For this purpose we included a predictive model to mimic human imagination. Since we do not know the exact parameters and data available of this particular simulation under discussion, we cannot point out immediately a specific predictive model. However, in the case with the bad guy running around the corner we have to predict a discrete event. That would rule out for example a Kalman Filter. We would use data like:

- How long was he running? The longer the hunt is the more likely might be the need for a breathing time and the more likely will it be that he waits behind the corner in order to fire.
- What can be said about the possibility of finding cover behind the corner? There might be cues available, to what extent cover could be available.

- How dense is the venue populated with people? If this happens on a crowded side walk there might be an assessment possible about the likelihood that he keeps running or not.
- How large is the lead of the bad guy?

The answer in a specific application could then lead to a selection of a particular predictive model or a combination of prediction techniques (4)(5). Once the model is known in detail then the prediction component gets customized (7). Applying pattern matching can lead to a 'key – behavior' representation. The key does not have to be a single condition. In fact the more conditions are used the more precise the behavior should be. When the key matches the current situation, like exhaustion is true, high probability of cover behind the corner, and no other people around, etc, then the anticipated behavior could be 'bad guy will try to ambush' and then the appropriate behavior can be retrieved from a data base. Training the model would be to adjust the parameters used such that the results from test runs are consistent with the model.

The result of this research is neither a crystal-ball-like capability to project into the future nor a "plug and play" component for all types of simulations or decision-making support tools. It is a framework for a computational model of mental simulation in a simulated combat environment. However, there is a degree of usefulness for a series of similar problems and simulation applications that involve uncertainty and time considerations.

"Uncertainty" in this context means relying on assumptions about, or estimates of, behavior and the size or type of the decision-influencing factors. The range in complexity can go from simply assessing the trend of a certain stock commodity to a life-and-death judgment whether or not an old wooden bridge in the jungle or Hindu Kush can carry someone's weight when crossing. As outlined in Chapter II, there are many ways to assess or estimate decision-making parameters, but a model of mental simulation considers only the parameters and the weights. It does not care about their derivation. One important family of decision-making may be based on probabilities that were deliberately fabrication by a devious opponent. Therefore, the model is relatively generic in its parameter

estimation and the respective models used. With respect to time considerations, this research focuses on decision-making situations in which the decision does not require taking action immediately. The system may suggest an immediate need for action, but not in all cases. The system can cover the gamut from making decisions about the right time to trade stock shares to determining the right time for an ambush.

Our research can be used to receive guidance, gain experience, and recognize pitfalls when modeling a mental simulation computationally, not merely conceptually. The main features of the architecture described above are three-fold. A decision must be made with a certain timeframe; the parameters used are:

- How did I perform last time, when I was in a similar situation?
- What can I expect to happen in the near future?
- What do I know/ assess when the expectation comes true?

Combining these parameters in a mental simulation model can make the behavior more human-like.

To illustrate, what we mean, consider this situation in baseball. In a pitcher-versus-batter "duel" situation, the batter has certain expectations about what the next pitch will be. Overall, he knows that there can be four "balls," three "strikes," and, at most, six possible pitches and strikes excluding foul balls. If there are four balls, the batter will walk to first base, and it is the next batter's turn. If there are three strikes, the batter is "out." When the batter has had three balls and no strikes, he can be reasonably confident that the next pitch will be an attempted strike. Otherwise, a fourth ball will result in a "walk." Conversely, when the batter has had two strikes and no balls, the pitcher has greater freedom in his pitch selection. He can choose to try to strike the batter out or to pitch balls, in the hope that the batter will swing on a bad pitch. But what about other situation in which the batter has had less balls or strikes? The batter estimates what the pitcher will do. He can vary the speed, "break," and location of the ball, but his choice is limited. The batter has an array of expectations. If he has observed the pitcher's behavior from batter to batter, or knows his history from videos of former games, he may have a mental model of what the next pitch is likely to be, given his own particular situation.

Within the mechanics of the game (balls and strikes, …) there is also mental simulation included. In case of modeling this, we could use the techniques explained in this dissertation to provide that human behavior based on mental simulation. And as Forsythe pointed out, this extra realism could make all the difference in the world in a training simulation.

IV. MODEL IMPLEMENTATION AND RESULTS

A. INTRODUCTION

This chapter describes the implementation of the model, based on the general architecture described in Chapter III, and discusses in detail the four components: simulation environment, situational awareness, mental simulator, and decision component. It also explains the treatment of the terrain assessment. We then introduce the experiments, which compare the model's predictions and firing behavior to those of human agents. The chapter concludes with the results of the experiments.

B. SPECIFIC IMPLEMENTATION OF THE GENERAL ARCHITECTURE

To enhance the cognitive capability of certain entities, the intelligent software agent, used in combat simulation models, we built a decision-making model, using Combat XXI (see Chapter III). The agents are enabled to

use statistical estimates to predict next events,

be sensitive to decision-making contexts,

have an improved situational awareness,

determine potential actions, and

provide an explanatory component for the reasoning.

Figure 21 shows the application of mental simulation in a simulated combat environment in between the situational awareness and the making of that decision. At left, when resources are available, the entity fires; unless told otherwise, it always fires. At right, the entity considers the context, predicts the next event, and fires accordingly.

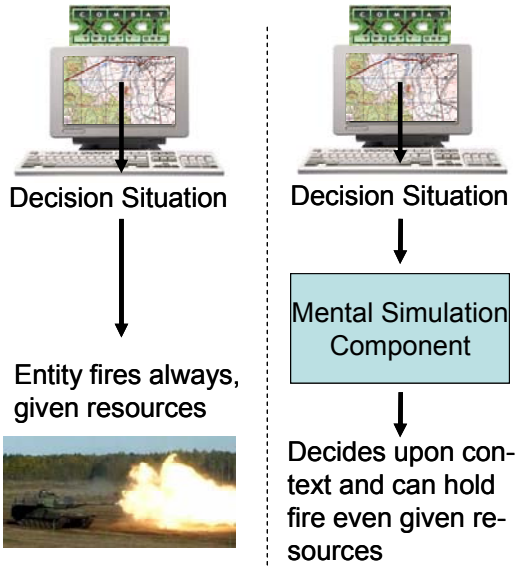


Figure 21. The role of Mental Simulation in the current work.
On the left side: Decision-making Situation Given the resources, the agent always fires.
On the right side: Decision-making Situation Given the situational context, the agent can decide not to fire, even though given resources.

1. Components

The following explanation of the four components refer to Figure 18.

a. Environment/ Combat XXI

It was a conscious decision to couple to an existing simulation system. We wanted to use one of the latest systems in order to have the highest likelihood that the model developed could be applied in a real system. With this approach, we only use the information that is provided by the combat simulation system in the format that it is provided, and generate new information based on this. Thus, we are able to reduce the actual adaptation effort that would be required to incorporate our work inside a combat simulation model like Combat XXI. In order to avoid dealing with version changes of the combat simulation model which is under active development, we chose to construct our model externally and not embed it directly into the combat simulation model.

Here we discuss those features of Combat XXI that are relevant to our implementation. Combat XXI has a set of default loggers, shown in Figure 22.

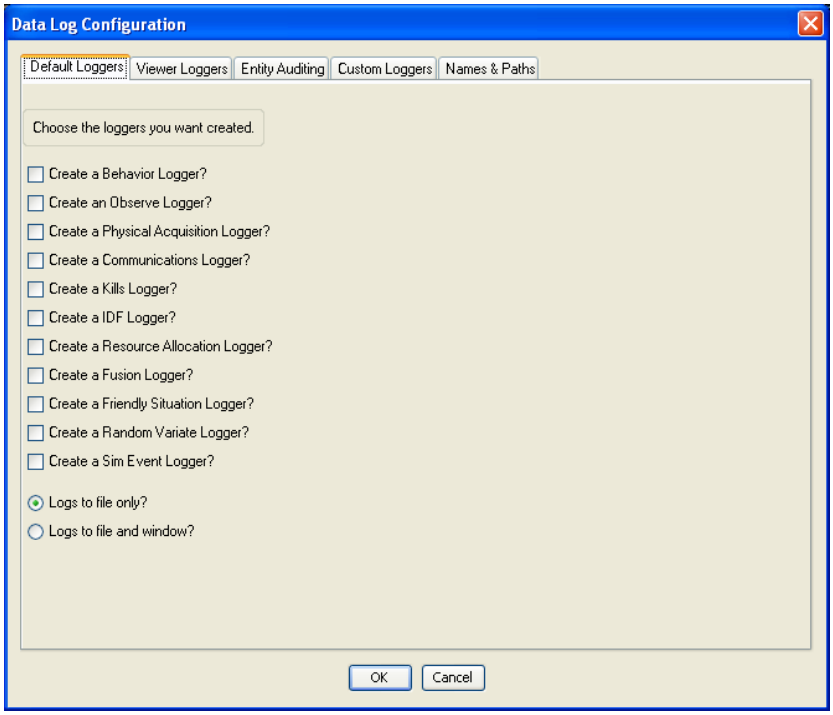


Figure 22. The Combat XXI data-log configuration window

In addition to the default loggers, customized log files can also be created. However, all the information required for our model is covered by the default log-file settings. The following standard log-files yield the input for the model as implemented: KILL-Logger, PHYSICAL_ACQUISITION-Logger, and MOVEMENT-Logger. The BEHAVIOR_RULE-Logger was used for verification purposes only. The standard KILL-Logger captures data on fire, detonation, and damage events, which go together, because events are cross-referenced and logged with the key "event_id." Table 2 gives an overview of the data contained.

Logger	Description	Data provided	Input for model
PYSICAL_ACQUISITION		simulation time	x
		observer_id	x
		observer_easting	x
		observer_northing	x
		observer_msl	
		observer_agl	
		observer_speed	
		sensor_name	
		range	
		detection_level	x
		target_id	x
		target_easting	x
		target_northing	x
		target_msl	
		target_agl	
		target_speed	
KILL: Fire events		eventType	x
		simulation time	x
		entityID	x
		munitionEventID	
		munitionName	
		range	
		targetID	x
		x	x
		y	x
		z	
KILL: Detonation events		eventType	x
		simTime	x
		entityID	x
		munitionEventID	
		munitionName	
		x	
		y	
		z	
KILL: Damage events		eventType	x
		simTime	x
		entityID	x
		munitionEventID	
		damageType	x

Table 2. The log-files required to create the situational awareness and situational context for decision-making events

The PHYSICAL_ACQUISITION-Logger holds the detections that occur on the blue side and the red side. This log-file has no aggregated observations, which means that each data line covers only one observer and one target plus associated data, as shown in Table 2. To determine whether an observer detects multiple tanks in one observation, the data file must be processed separately, outside the simulation model. If two data lines have the same simulation time, it is inferred that this is a multiple observation at a given time. The simulation engine in Combat XXI ensures that no other logger, such as DAMAGE or FIRE or DETONATION, uses the same simulation time. Therefore, no mixing of data from differing event types, such as, for example, a detection event and a fire event, can occur. It might not be obvious, but the PHYSICAL_ACQUISITION-Logger provides only information when a target has been detected. There are no events when a target is going out of sight. This is in the following referred to as missing "undetections."

The MOVEMENT-Logger contains data of the path each entity takes. If an entity does not change its position, no event is logged. This data is not required for running the model. However, the logger's data provides a smooth display of the unit movements and prevents large jumps from observation to observation. This eases the optical assessment. The movement data is not gathered by any type of sensor used in the scenario: it is taken from ground-truth. Figure 23 gives an example of the logged data used in our model. The first column depicts the simulation time; the second column the observer/shooter; the third column indicates the log-file from which the data is taken. The remaining columns are either the coordinates or the type of damage invoked. The units in Combat XXI are tagged with id-numbers.

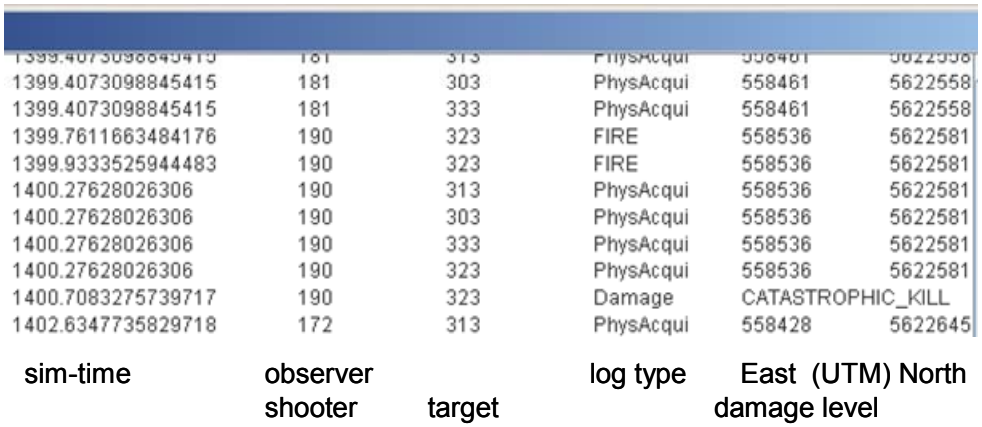


Figure 23. An example of the tuned output of Combat XXI.

b. Situational Awareness

Situational awareness is a critical component in a decision-making environment. The better the awareness the more accurately all the parameters that influence a decision-making situation can be assessed. Good situational awareness is a prerequisite for making a "good" and successful decision. The situational awareness component comprises the entity commander's growing knowledge. It is comparable to a human commander's cognitive picture of a battlefield situation. There is no data retrieval from ground-truth. The Combat XXI output files yield data about detections and engagements on a battlefield, plus associated data like time, location, shooter, targets, etc., all of which are ordered chronologically. In our implementation, there are sensors that are entities,

type “infantryman,” “scattered” over the battlefield, that detect red entities. The sensors are stationary and have no operational impact, which means they do not engage and they do not get engaged. The red movement is also not influenced by the sensors. The sensors yield an operational picture for a platoon commander that would actually be given him via various enemy situation reports.

With respect to its knowledge about enemy formations, the model starts from scratch. In other words, the model has no presumptions about the enemy’s behavior or formation. This receptive status allows the model to be flexible, since the commander cannot count on meeting with strict formations, such as those once aligned according to the old Warsaw Pact rules. The model can in principle be supplemented with a “knowledge or experience database,” which will enable it to be feasible, that is, operable even when few observations have occurred so far. The formations that are detected and categorized carry information as to their size, type, direction, and speed. Currently, the modeled formations are homogeneous, which means a forward artillery observer accompanying a combat unit is a distinct formation, even if they operate together. In the current model, the size of a formation is taken to be the sum number of distinct entities per formation that have been detected. Possible enemy objectives with respect to terrain, such as seizing key terrain, are not yet represented.

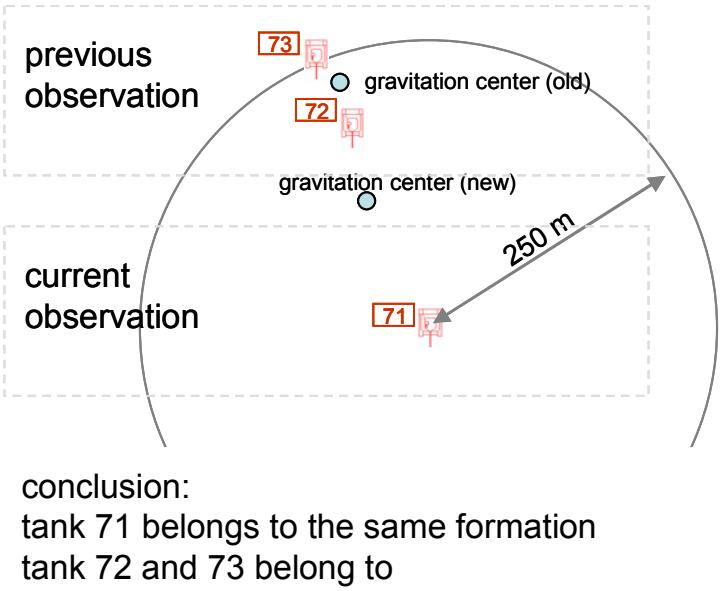


Figure 24. Assigning new observations to tank formations

Figure 24 shows how newly observed tanks are assigned to formations. Each formation "carries" a gravity center, the geometric center of the entities' coordinates. The distance of the newly observed unit(s) to this gravity center determines primarily whether a new tank could be part of that formation. If the distance is less than a threshold value, then it is a candidate for the formation. If the tank is being observed for the fist time, it is assigned either to the closest formation or, when the distance value is beyond the threshold, to a new formation that is then set up. In the current implementation, the threshold is set at 250 meters. The Field Manual for a U.S. tank platoon states

> Formations are not intended to be rigid, with vehicles remaining a specific distance apart at every moment. The position of each tank in the formation depends on the terrain and the ability of the wingman driver to maintain situational awareness in relation to the lead tank. At the same time, individual tanks should always occupy the same relative position within a formation. This will ensure that the members of each crew know who is beside them, understand when and where to move, and are aware of when and where they will be expected to observe and direct fires. Weapons orientation for all tanks should be adjusted to ensure optimum security based on the position of the platoon in the company formation (Field Manual 17-15, 1996).

Although there is no doctrinal number for the distance between tanks, since that always depends on the mission, situation, time of day, etc., according to our experiences and talks with military experts, a distance of 250 meters seems reasonable.

In addition to the formation membership information described above, situation awareness comprises the following: a commander's knowledge of how many formations are in front, how many distinct tanks are assigned in total to particular formations, and the estimated speed and direction of the formation. Since we avoid access to ground-truth, the speed and direction are known only as estimates. If he sees a "known" tank, he also knows the formation to which it belongs and where that formation's remaining tanks were reported last. Figure 25 displays an example of an agent's information about the tanks he sees.

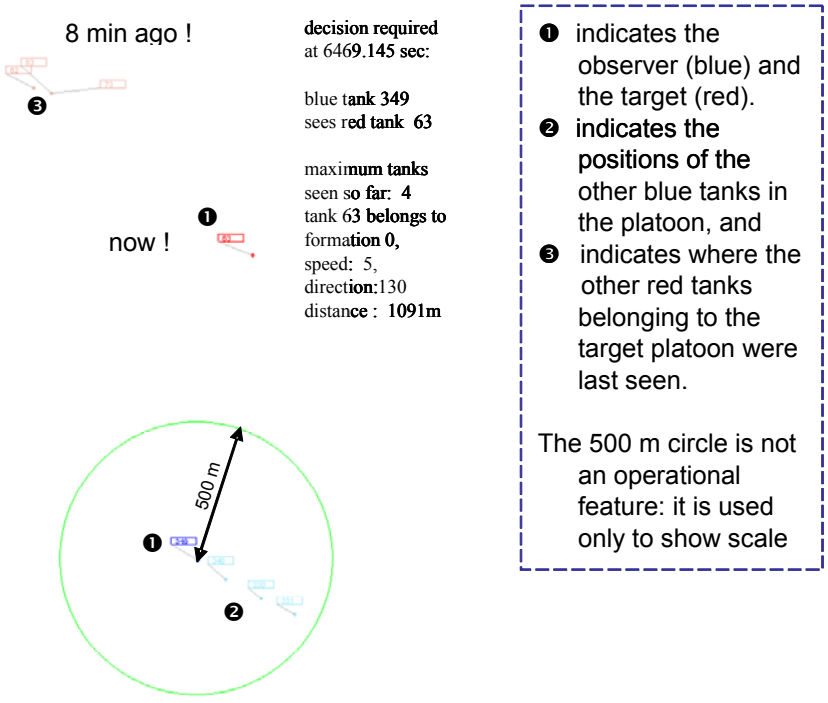


Figure 25. The context provided when a decision situation is invoked.

The situational awareness component also updates or creates the predictive model, whose input, independent of the model type used, consists of observations of spotted enemy tanks. The model yields an option to create either a predictive model for each formation or one model for all incoming formations. For now the number of tanks

seen in one observation represents the state of the predictive model. In section IV.B.2 we discuss the terrain assessment and incorporate it also into the state, then a state of the predictive model will be two-dimensional. In the next section, the states are covered in detail.

The need for a decision occurs when a blue tank sees an enemy only. At that point, the situational awareness component yields context data to the mental simulator component.

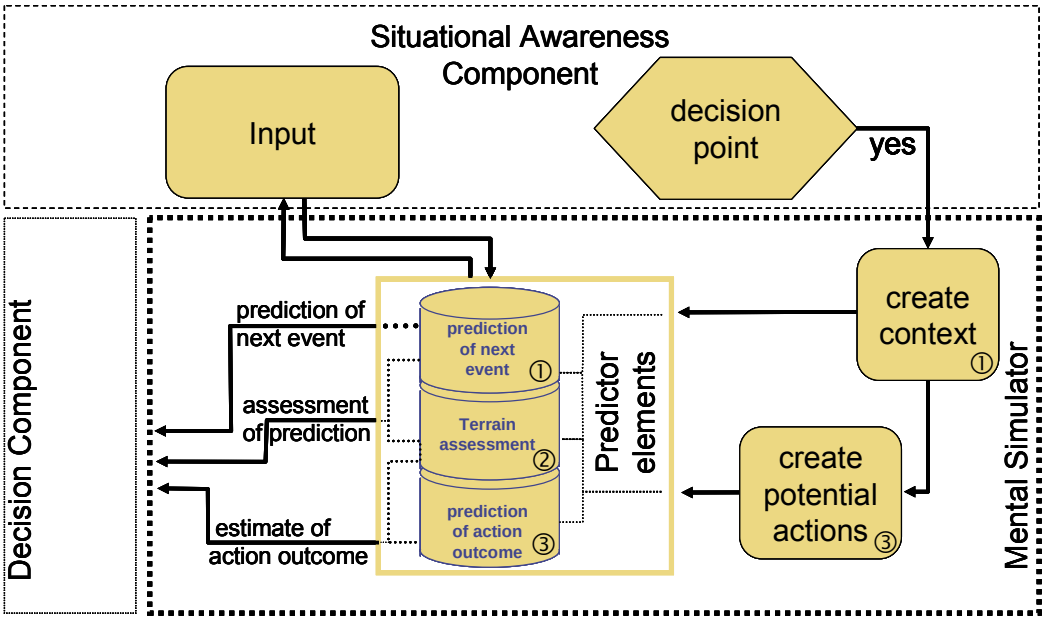
c. Mental Simulator

The mental simulator, the most central component of the architecture, makes the difference between our simulation system and all other combat simulation systems. A detailed view of the mental simulator is depicted in Figure 26. The circled numbers in the figure depict the three assigned tasks of this component:

1. to retrieve a context from the situational awareness component, and to estimate the next probable observation and the average (typically we use the median) time when this event will occur;
2. to predict the terrain quality in the near future, and
3. to create potential actions and estimate their outcomes.

task 1: retrieve context

The context binds the variables of the maximum number of tanks per formation and determines whether, for the upcoming decisions, several formations are currently observed. In the case of an observation of tanks from multiple formations, the one that is the greatest threat is selected to engage first. In the current implementation, the threat is proportional to the distance, which is part of the provided context. If the distance becomes less than 800 m, the blue tank always fires, because the risk is too high that the red tank will fire first. The current value 800 was selected on a personal-judgment basis: it seemed reasonable given the scenario and the three-dimensional perspective of the battle space.



Note: ①, ②, ③ reference the tasks of the mental simulator

Figure 26. The Mental Simulator in detail.

There is also a need to assess whether the red tanks can detect the blue tanks. This assessment exploits a given artifact: that the tanks do not look around a full 360 degrees. The decision component later on decides on the threat evaluation done in the mental simulator.

task 1: predict the next observation

The system, that is, the tank platoon, is in state "i" when in the current observation, "i" tanks are observed. In other words, a "state" is defined as the number of entities detected at an observation time "i." Each agent tracks the observations according to the state diagram in Figure 27. The current state is colored yellow. (When we introduce the terrain features, we will make this state machine two-dimensional). The expected median time for a transition is also determined in parallel to the expected next event. However, this time can be used to estimate only when we can expect the next event, not how long a certain observed tank will be visible, because Combat XXI yields only detection, and no undetection.

The prediction of the next event(s) is currently accomplished by using a Markov Chain. This stochastic state machine assigns probabilities to state transitions

from state "i" to state "j." The probabilities reflect the frequency of state transitions in the observations that were analyzed prior to the current observation and that were normalized so that all the probabilities of emitting arcs in a particular state add up to 1. Although there are other possible models for this data, considering the current status of the combat model used here, a finite-state machine was best suited for the available data.

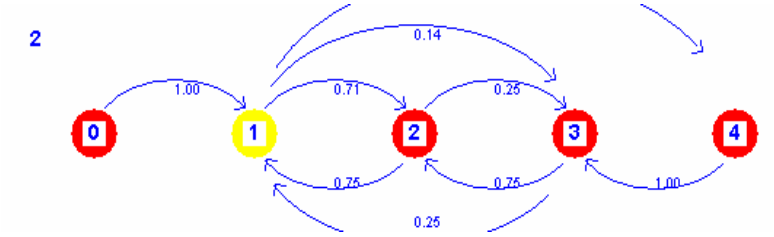


Figure 27. Example of the state machines for a defending platoon that is currently in state "1." A state indicates how many entities are seen in a current observation. The arcs are labeled with the transition probability to the next state. The median dwell times are also stored but are not shown here.

State "1" means that the agent currently sees one entity: he will stay in this state until he makes another observation. If he now sees two tanks, then he moves into state "2," and the transition probabilities and mean dwell times (duration he stays in a state) are updated. Since the combat model does not currently provide data as to when an entity goes out of sight, the state "0" never reoccurs. If the model made available 'going out of sight events', the mean or median dwell times would be more realistic. The second section of the chapter will show our attempt to incorporate such undetections and to estimate when an observed unit ultimately goes "out of sight." Although, initially, the model was trained by having various sensors along the main approaches, we eventually used comparable scenarios and then initialized the state machine with probability and dwell-time values.

An easy prediction criterion for the next transition could be to choose the arc with the highest probability. However, using this approach would not exercise transitions to states of lower likelihood. This would also be a very simple model of mental simulation that has the flavor of RPD. The degree to which humans take less likely out-

comes into account when mentally simulating is, we believe, still a research question. In order to ensure that events with a lower probability will also sometimes be predicted the author uses a Monte Carlo simulation for sampling the values from the probability distribution as estimates. With Monte Carlo Simulation all kinds of questions can be addressed. This method is widely used when an analytically computation is very hard, even though the mathematical model is completely determined (Axtell, 2000). The mathematical and statistical literature refers to one class of problems of this type as "boundary crossing" (Giraudo, Sacerdote, and Zucca, 2001). Many simulation runs can be conducted and then the frequency of occurrence, in the case of boundary crossing when the curve hit the line, can be taken as an estimate. Considering the current decision context there are mainly two questions that seem promising for the model. The first is related to the estimated time to expect a transition and the second addresses the multiple state sequences. The precise questions are:

- What is the most likely state sequence with the next x observations?
- When will a state (or set of states) of interest next be entered?

These mathematical formulations correspond to the human questions "What will happen next?" and "How long until (some anticipated event) occurs?"

task 2: predict the terrain quality in the near future

Predictor element 2 in Figure 26 accomplishes this task. This task is discussed in a separate section (IV.B.2) in detail.

task 3: create and evaluate potential actions

The mental simulator is also tasked to create potential actions. The current implementation is coded to create two potential actions: 1) to fire immediately when a target pops up on the battlefield, and 2) to hold fire.

In case 1) the risk of outflanking arises and the likelihood the likelihood of not seeing all tanks increases. In case 2) when all or most of the red tanks are visible, the likelihood that they will try to outflank the blue tanks is relatively small. In a real-world situation, they would most probably move in ways to avoid cross-movements relative to

the enemy, and try to engage as fast as possible. In Combat XXI, these two actions were simulated with the Run Manager, and the output was determined with respect to blue losses, red losses, and the "starting state," which is the number of red tanks seen in the first observation. This generally varied between one and three tanks. The loss values are stored in a database. The predictor element 3 in Figure 26 would retrieve this information, in the following referred to as loss chart.

Besides to fire immediately or to hold fire indefinitely, there exist a possibility of waiting a certain amount of time. However, this option has not been implemented in the current work due to Combat XXI issues. Conceptually, the length of time the tanks wait before firing would depend on various parameters of the targets, such as direction, speed, and distance. We could assume that a tank platoon has a certain "depth" on the battlefield. A unit's depth is the distance between the first tank and the last, as mapped in a line showing their direction. According to the field manual, a depth of 100 meters for a platoon in an attack seems reasonable. If the first tank seen has a speed of 15 m/s, then, in seven seconds, the last tank will generally appear. Thus, one potential action would be to wait seven seconds before firing. Figure 28 shows how a unit's depth is defined.

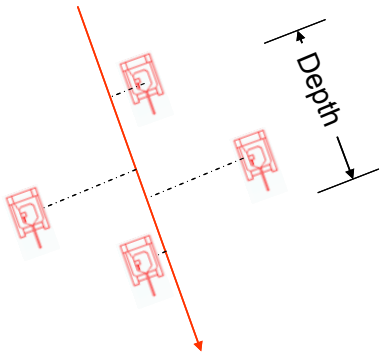


Figure 28. Depth of a unit.

d. Decision

In the current implementation, the decision component requires three inputs:

- a prediction of the next event likely to occur,

- an assessment of the prediction with respect to expected terrain influence, and
- an assessment of possible actions.

In other words, the decision component takes the predicted number of tanks to see in the next observation, retrieves a median time for this event to occur, and estimates the expected location of the tanks to be seen. The new location estimate is calculated geometrically based on the estimated speed and direction. For this estimated location there exists a terrain cell attribute that indicates how likely an observation will occur in this location. The terrain cell attribute is defined in terms of the number of detections in a preliminary run. The terrain attribute will be used to assess the prediction. The assessment of possible actions is retrieved from the database. In preliminary runs in similar scenarios, the dependence of red and blue losses on whether the platoon fired immediately or delayed the firing and also on the number of tanks seen in the initial observation was determined.

In our basic initial example of an agent, a tank platoon commander, the tank decides to fire according to the decision tree in Figure 29. Once a tank is in view, this decision tree gets activated because the need for a decision occurs. The decision component proceeds downwards through the tree until it hits a node that says “fire” or “hold fire.” At each node a condition is checked and based on the outcome of this condition the respective path is chosen.

The top node evaluates the threat level of the tanks observed to the blue (friendly) tank platoon where leader’s decision process is being modeled. Determination of threat level is based on range in the current implementation. Other potential factors could include the heading of the tank or whether the enemy gun points towards the blue position. Our handling of threat assumes that the blue tanks are in a turret-down or hull-down position, in which the probability of detection is relatively small. The threat level might also be influenced by the mission, not only by the risk of being shot at. A good example would be a mission of suppression of enemy reconnaissance. Even if the enemy tank does not detect the blue position it can still be a severe threat, because of the capability of reporting reconnaissance results that might endanger blue’s own operation. The heading of the enemy tank’s gun cannot currently be retrieved in Combat XXI. There-

fore, it is not modeled. However, clearly determination of a threat can be based on various parameters. If an immediate threat is assessed, that is the enemy comes within a certain threshold distance, then the tank immediately starts the engagement process to prevent being shot themselves (path ⑨ in Figure 29). Otherwise a hierarchical approach in the decision tree is further pursued.

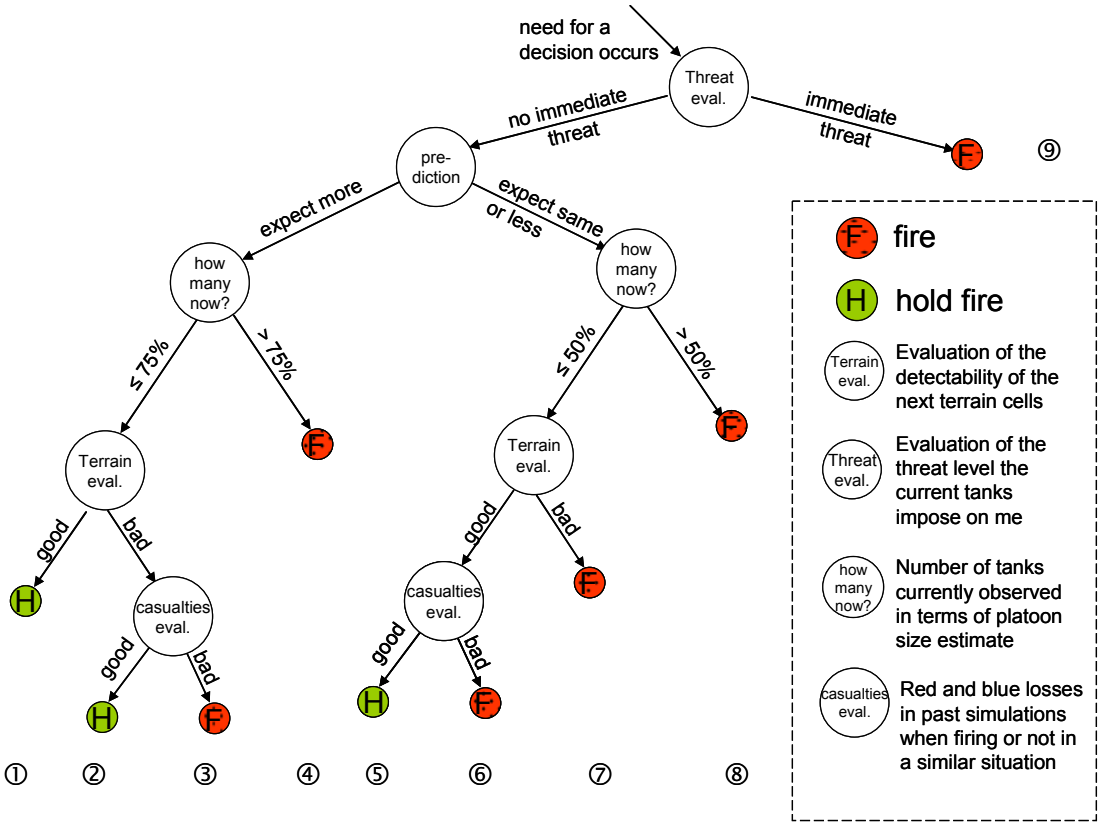


Figure 29. The decision tree for rendering a decision. The numbers 1 to 9 below the decision serve as references to the decision tree branches in the text.

In the next two layers of the tree, the prediction of how many tanks will most likely be seen in the next observations is used. If the prediction will be at most the same number of tanks as currently seen and this value exceeds 50% of the estimated platoon size then the engagement process is also initiated. This rule captures the case where currently three or four out of four tanks possible are observed and it is unlikely to see more in the next observation. Therefore, firing at the ones observed would be a reasonable thing to do. For the case that currently only two tanks out of four are observed, the

terrain is additionally assessed, and in case of a good detectability of the future terrain cell the casualty evaluation is conducted, otherwise the engagement starts. If the casualty evaluation is promising then fire is on hold, otherwise also the engagement process starts immediately (cases ⑤ - ⑧ in Figure 29).

If the prediction indicates a higher number of tanks than currently observed, then the expected percentage of the estimated current platoon size the tank commander will see is determined. This captures the situation when, for example, a platoon has five tanks and four tanks are seen and the system actually starts firing at them (case ④ in Figure 29).

If the number of enemy tanks currently observed is less than the maximum number possible, for example two in our example, then the terrain evaluation triggers the engagement process. If good detectability is anticipated, the model holds fire (case ① in Figure 29). If poor detectability is anticipated, the model assesses the casualties evaluation from preliminary runs. If the casualty evaluation indicates fewer losses when waiting to fire then the model holds fire, otherwise it fires (cases ② and ③ in Figure 29).

These factors enable the simulated platoon commander to make better decisions. The decision tree and the conditions were discussed with officers from the armor branch of several countries represented at the Naval Postgraduate School. In existing models, inappropriate immediate firing remains unpunished because the attacker also behaves inappropriately, ignoring the first shot or even a resulting kill and continuing to follow the scripted path.

The decision component also creates the explanatory component of the system. This means it provides a text string from which the user can see why decisions made by the model turned out the way they did; making the rationale transparent to the user. There are no anonymous numbers that lead to a decision. All numbers used have a meaning in terms of losses, time or probabilities. Therefore, the decisions can be explained in a natural human way.

2. Terrain

This section describes how we incorporated terrain into the mental simulation process. In a real combat environment, a commander observing a tank can continue to look at the tank as long as the same line of sight also continues. In the simulation envi-

ronment, there are events at a particular point in time that determine that certain detections have been made. But, in general, there is not information available as to how long the observed entities will be visible. Although the system developers accepted that there is a need also for undetection information, no such implementation has yet been accomplished. Therefore, we had to work around that lack to get information as to when a tank would probably go out of sight.

a. ACQUIRE Algorithm

The U.S. Army's current standard algorithm for target acquisition is the AQUIRE model. The ACQUIRE algorithm is a common search-and-target-acquisition algorithm used in many army force-on-force models (Cioppa et al., 2003). The ACQUIRE algorithm predicts target acquisition performance for imaging systems that operate in the visible, near-infrared, and infrared spectral bands. Therefore, it covers all sensors that occur in our currently implemented scenarios. According to the user's guide, the ACQUIRE algorithm

> predicts the expected proportion of an ensemble of trained military observers who can discriminate a target of a given size and temperature difference with the background, under specified atmospheric conditions (ACQUIRE Range Performance Model for Target Acquisition Systems, 1995).

The ACQUIRE algorithm was developed for retinal image sizes that are generally smaller than the fovea, which means they are more than 200 meters away (ACQUIRE Range Performance Model for Target Acquisition Systems, 1995), which, in our case, makes the algorithm applicable for tank detections beyond 200 meters.

The ACQUIRE algorithm uses four categories of input parameters to determine the level of acquisition: target characteristics, environmental effects, sensor characteristics, and task description inputs. The scenario-independent data that is required for running the algorithm is stored in an unclassified data base that was provided to the Combat XXI developers and used "as is." In our context, the main task for ACQUIRE is the prediction of the performance of: target spot detection, target discrimination, and time-dependent target detection. Target spot detection means the target is viewed against a uniform background. Target discrimination is used to determine the level at which a

target is detected. The levels of detection are currently categorized according to an increasing order of detail into detections, classification, recognition, and identification. In our experiment we ran the model at different levels of target detection. The objective of the time-dependent target detection is to determine the probability of detection as a function of the amount of time allocated to the task. This is exploited in the terrain attribute determination with respect to the time the simulation runs. The ACQUIRE algorithm uses a Field of View (FoV) and a Field of Regard (FoR) nomenclature. Field of View is the horizontal and vertical angle that the sensor looks at, plus a scaling factor that is not of further interest to our simulation. The ACQUIRE algorithm is applied independently for each FoV. Before a FoV can be revisited, the entire Field of Regard must have been scanned: thus, the bigger the number of FoVs per FoR, the longer it is before any one FoV can be revisited. Figure 30 depicts the relationship between Field of View and Field of Regard in the ACQUIRE algorithm.

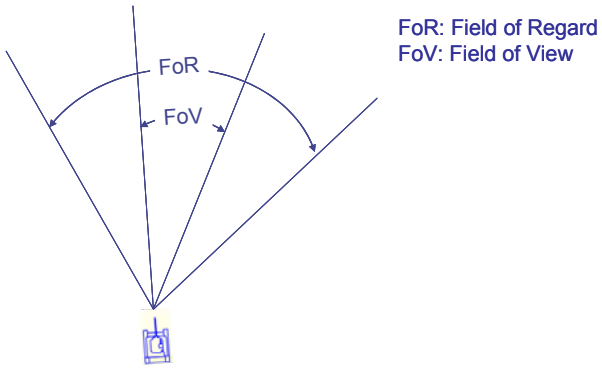


Figure 30. The relationship between Field of View and Field of Regard

The calculated probability of detection is compared to a random draw to determine whether a detection of a particular target has occurred. Therefore, the ACQUIRE algorithm is stochastic. If a FoV is revisited, the result can be different than that of the first visit, even if no entity has moved.

b. Terrain Attributes

A terrain attribute is an index that determines whether a particular terrain cell can be categorized as having either a "good" or a "bad" rate of detectability. Our terrain of interest, that is, the site where we expect decisions to occur, is divided into 100 x

100 m cells. In each cell, approximately four to six tanks were randomly distributed. No entity (i.e., tank) was moving, but the target acquisition algorithm was made active. Then the simulation is turned on and the detections, which occur over time, are recorded. The graph in Figure 31 shows how the detections in a particular repetition occurred over time. To determine the cell attribute, we conducted 50 runs of the combat simulation model. Figure 31 shows the detection rate. In our scenario we had scan times per FoV that were normal distributed over a mean of 3.5 seconds.

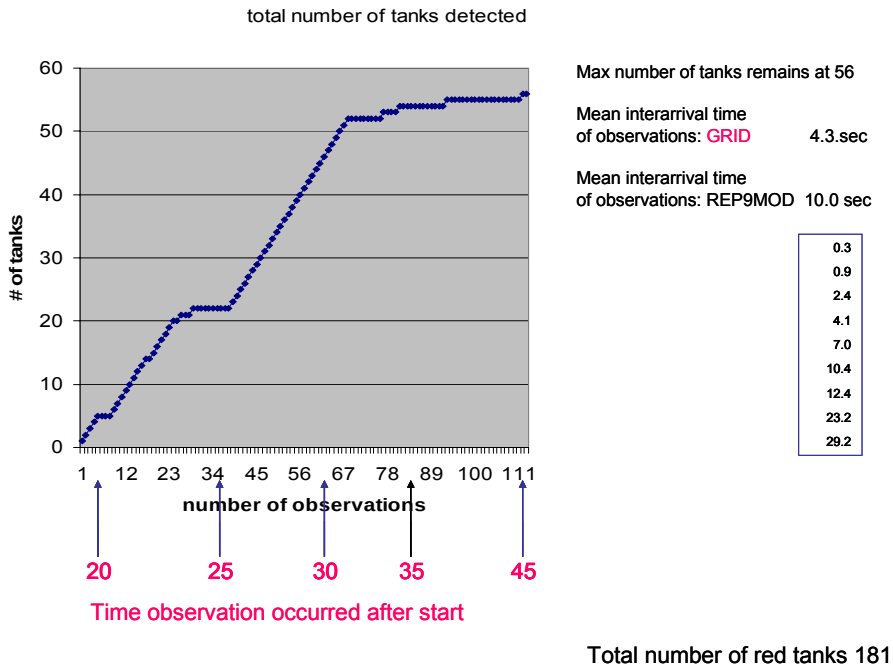


Figure 31. Total detections over time.

Figure 32 depicts one example of the terrain assessment: the cell with the coordinates (59200, 23100), which contains six tanks. Their numbers are listed at the right. The ACQUIRE algorithm detected only one tank. A cell was attributed as "good," in terms of its detectability, when more than 50 percent of its tanks were detected. In this case, the cell was attributed as "bad."

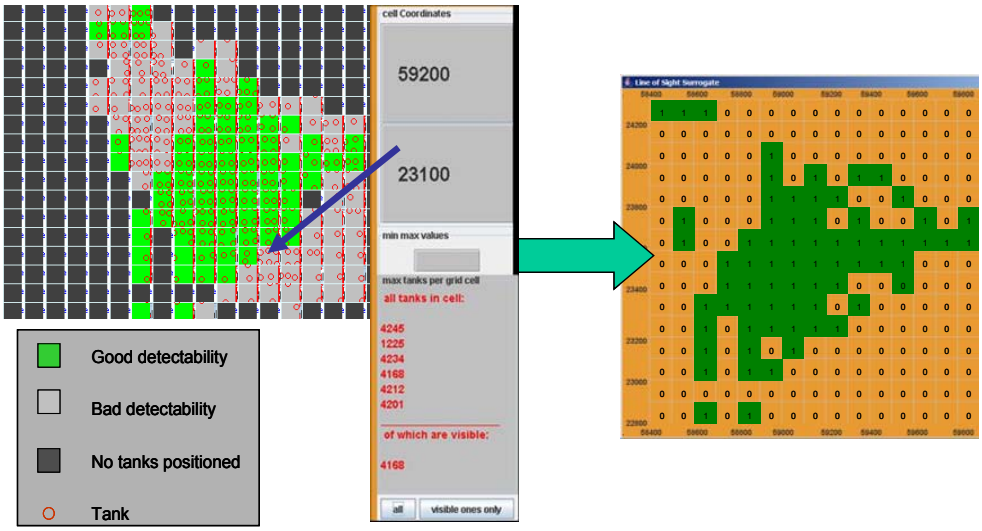


Figure 32. Assessment of terrain attributes

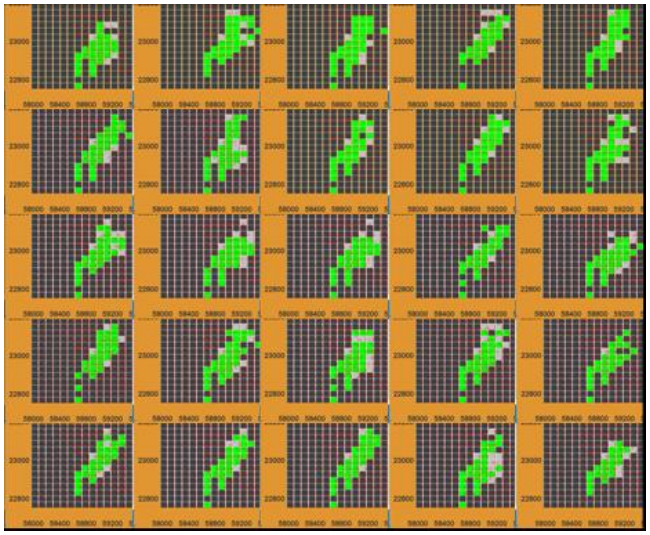


Figure 33. Variation in terrain attributes per repetition

Since the detection is stochastic, the attributes for the terrain cells are aggregated. A terrain attribute also depends on the location of both the observer and the target. In our example, none of the tanks, including the observer's is moving over the course of a single run. Therefore, we also conducted several runs in which the target tanks randomly changed position within their 100 x 100 m cells. The number of tanks per cell, however, was kept constant. The mean of the six runs conducted was forty-seven detected tanks, plus or minus seven tanks, in a 1.4 x 1.4 km square. The number of cells

containing tanks and having line of sight to the observer tanks is 121. Thus, the variation per cell is in average less than half a tank. Table 3 shows the results of the comparisons.

A terrain cell earned the attribute "good," indicated by green in Figure 33 , when in 90% of the cases half or more of the tanks were detected. In all other cases the terrain cells were attributed as "bad." When there was no detection at all, the cell was colored dark gray; in the remaining cases, light gray.

	Run 1	Run 2	Run 3	Run 4	Run 5	Mean	S-Dev
Replication 1	35	35	40	36	38	36.80	2.17
Replication 2	47	47	51	49	50	48.80	1.79
Replication 3	39	40	44	41	42	41.20	1.92
Replication 4	49	49	55	46	53	50.40	3.58
Replication 5	50	49	56	52	53	52.00	2.74
Replication 6	40	40	44	42	44	42.00	2.00
Replication 7	37	37	41	36	39	38.00	2.00
Replication 8	46	45	49	47	48	47.00	1.58
Replication 9	55	55	59	56	59	56.80	2.05
Replication 10	56	55	61	58	61	58.20	2.77
Mean	45.4	45.2	50	46.3	48.7	46.73	2.24
S-Dev	7.38	7.07	7.59	7.67	7.97		

total number of cells containing tanks: 121

Table 3. The number of detected tanks per run and replication

We also evaluated how well this approach performs. In order to do so, we looked at 10 replications of a single scenario involving a group of moving hostile tanks. This scenario is one of two generic scenarios we used in the experiments. This one, with the name "final4", is explained later. We examined all consecutive observations that had a change of terrain cell attributes associated with them. That means, when an observation i occurred in a "good" terrain cell and the observation i+1 occurred in a "bad" terrain cell then the number of tanks in each observation are recorded. This was done in the same way when the terrain cell attribute change occurred from "bad" to "good." When no change occurred, nothing was recorded. At the end of the scenario replications the mean values for changes from "good" to "bad" and from "bad" to "good" were determined and put into Figure 34. This figure displays on the left side the mean values of differences in number of tanks observed vertically. The x-axis displays the replications.

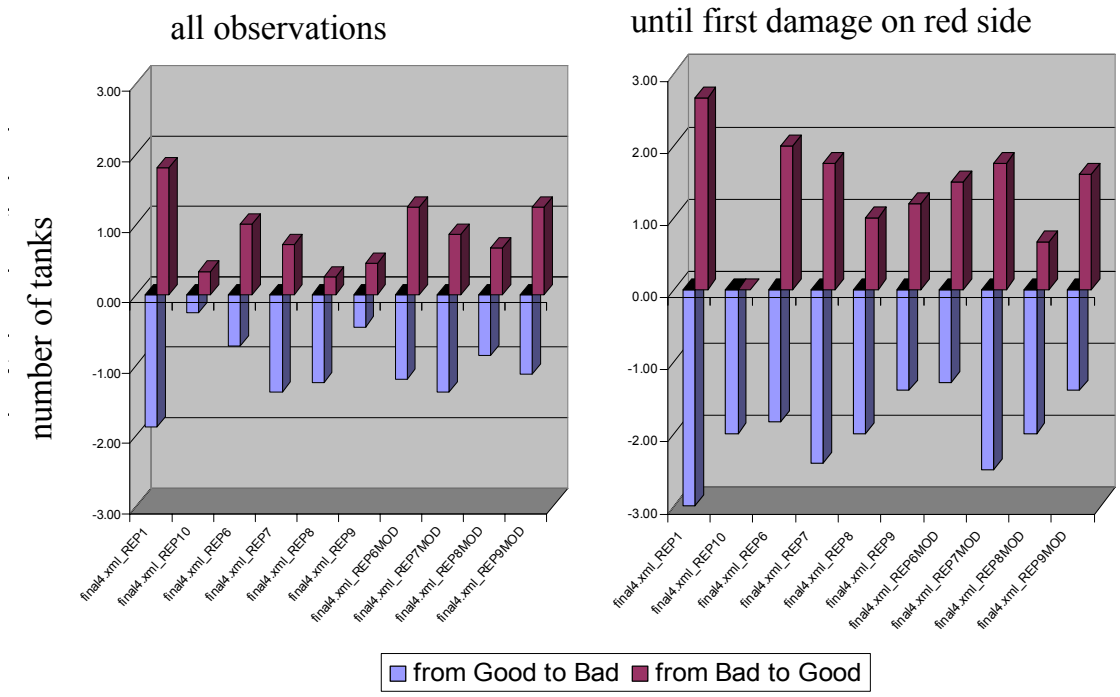


Figure 34. Changes in number of tanks observed around a terrain cell attribute change

Above the zero line are the values for changes from "bad" to "good" and below vice versa. It is apparent that all means are either above or below the zero line indicating that it can be assumed when going, for example, from a "good" to a "bad" terrain cell in average less tanks can be expected in the next observations. We consider this as an extremely valuable new feature in combat simulation environment. The right chart displays a truncated version of the data. Truncation was done when the first damage occurred. The right chart shows more clearly the difference between good and bad terrain cells because the maximum number of tanks observable decreases after damage occured.

C. EXPERIMENTS

We conducted four experiments. They all used the same (general Combat XXI) scenario. The first experiment responded to the question whether there would be a difference in prediction accuracy as a function of the number of state machines. This is a more technical experiment and was used to make design decisions. The next experiment, which involved human subjects, compared the prediction accuracy of the model to that of humans. This and the next experiment address the model's validity by comparing its per-

formance to human performance. The third experiment examined how the tools, provided to the participants and mandatory for the model to work, impact the human predictions. The fourth experiment compared the firing behavior from humans and the model based on experiment 3.

1. Scenario

The general scenario in Figure 35 was used in all three experiments: it features an area in the so-called Fulda Gap, close to the former Inner-German Border. This area was chosen because Combat XXI has digital terrain data available there. Also, the terrain data is very detailed and thus allowed a good visual assessment of the behavior presented by Combat XXI. Furthermore, the German Armed Forces have map tools for this area, so the terrain features in Combat XXI can be verified.

This scenario has two variants named final2 and final4. Both are xml-files so that their name may also appear on charts as final2.xml and final4.xml. The difference between them is the state transition variation. The sensors had slightly different locations so that the detections occurred differently. In the first one there occur mainly observations with few tanks, and the latter one has a higher percentage of observations with three or four tanks.

The forces depicted in the simulation were a blue platoon and a red platoon. The blue platoon consisted of four M1 tanks, which defended against a red armored company of thirteen T-72 tanks in total. Choosing these two tank types ensured that the database in Combat XXI, which is unclassified, would provide full support. No error occurred due to a lack of data.

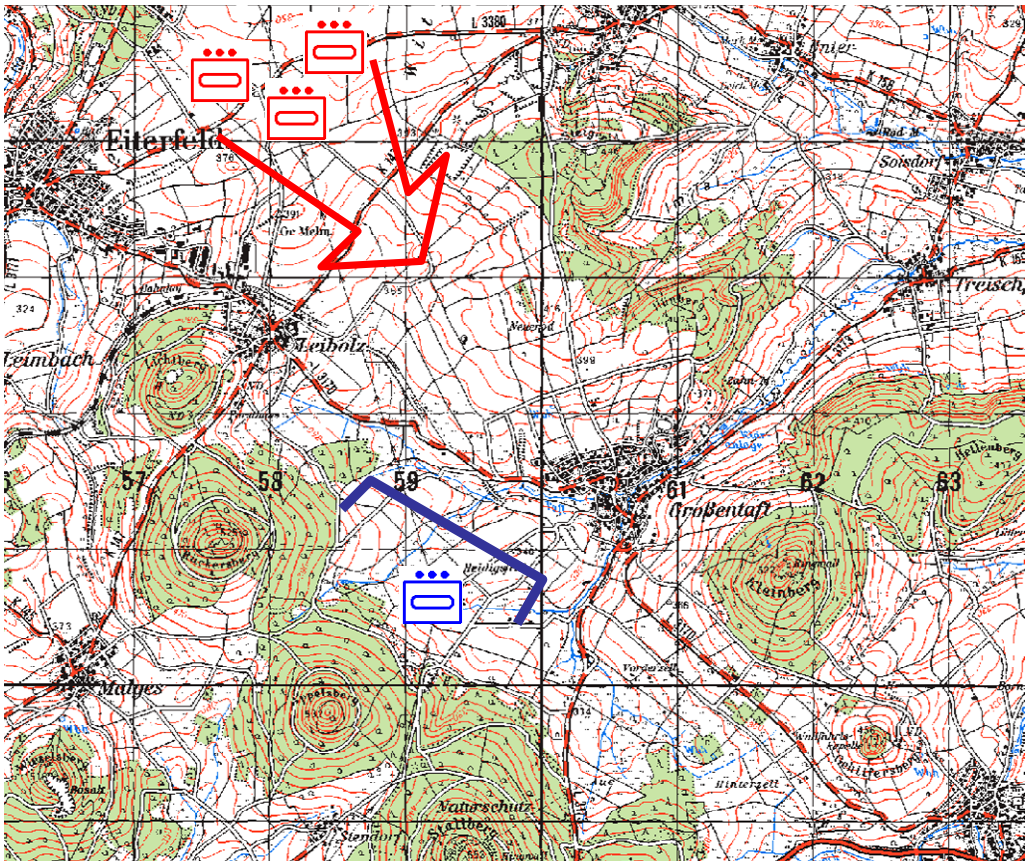


Figure 35. The scenario used for the experiments.

The red platoon attacked in three formations, one platoon each. The route of attack was a scripted path. The red tanks' behavior was constructed using Combat XXI's built-in infrastructure, with the single exception of the outflanking behavior. The tanks marched in line and changed to a column formation at a given phase line. If flanking fire killed the first red tank, the remaining, undetected three tanks outflanked the blue platoon along an alternate scripted path. However, if the blue tanks began delayed firing after more than one red tank was spotted, it was too late to outflank the blue tanks, and the ensuing "duel" situation had to be resolved immediately.

2. Purpose and Scope of the Experiments

a. Experiment 1: Different Number of Markov Chains

This experiment answered the system design question of how many Markov Chains to use. This experiment was intended to determine whether or not the

number of models/Markov Chains used had a significant impact on the blue tanks' prediction capability. The choices were either one model for all enemy platoons or separate models for each individual platoon.

In our scenario, the sensors were allocated equally to each approach route. Three red platoons approached southward. We ran the simulation ten times with five replications each with one Markov Chain in total (M_1) and the same number of runs with separate Markov Chains per individual formation (M_3). After that we compared the mean values for the percentage of correct predictions and conducted a t-test with the data obtained.

The Null-Hypothesis was that the mean for using separate Markov Chains for each formation is no better than the mean using one Markov Chain in total, which leads to $H_0 = M_3 \leq M_1$. The alternative Hypothesis was that using separate Markov Chains for each formation would increase the ratio of correct predictions, which leads to $H_1 = M_3 > M_1$.

Table 4 shows the results from the t-test for the percentage of correct predictions involving one Markov Chain for all formations (entitled: M1) and with an individual Markov Chain for each formation, that is, for each platoon (entitled: M3).

	M3	*M1*
Mean	62.1	60.7
Variance	28.32222	58.9
Observations	10	10
Pearson Correlation	0.977447	
Hypothesized Mean Difference	0	
df	9	
t Stat	1.629916	
P(T<=t) one-tail	0.068779	
t Critical one-tail	1.833113	

Table 4. t-test for comparing the mean values.

At a significance level of $\alpha = 0.1$, we can reject the Null Hypothesis and accept the alternative. At the .05-significance level we cannot reject the Null Hypothesis. In order to reject the Null Hypothesis at the 0.05 level the sample size has to be increased. For example, to detect a 1% difference in mean prediction capability 90% of the

time (assuming $\sigma_D \sim s_D = 2.71$) the sample size would need to be 80 runs. In order to detect a 1.5% difference 90% of the time the sample size would need to be 36 runs. However, given the terrain influence on the detections, the higher mean M_3, and the intuitive belief, that using separate Markov Chains for each platoon would be more accurate than one Markov Chain for all platoons, we choose for the rest of the thesis to use separate Markov Chains for each platoon.

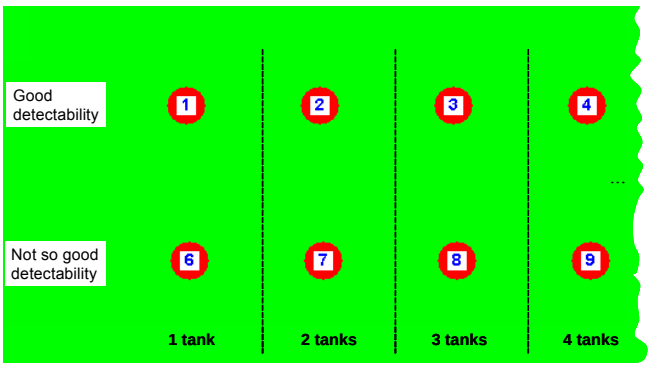
b. Experiment 2: Prediction Accuracy of the Model vs. Humans

This experiment allowed us to compare the prediction accuracy of the state machine used in the model to a real-world scenario involving human subjects. The experiment was set up as follows.

Each participant received three tools when conducting the experiment:

- A Markov Chain with the corresponding transition probabilities,
- A loss chart, and
- Terrain assessment of 100 m by 100 m cells.

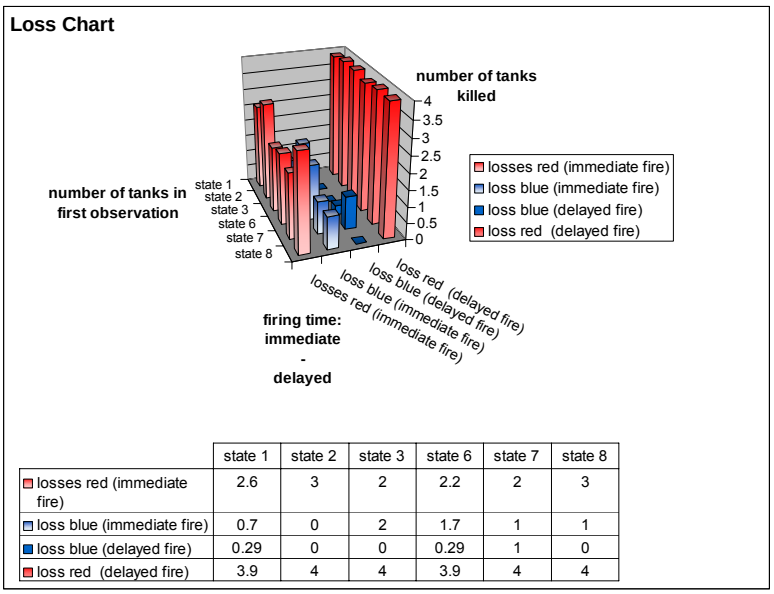
The scenario was run twenty times in Combat XXI. Each run is called a replication which varied only in the observation sequence. These runs yield an aggregated Markov Chain over all runs. This was the first tool and was provided as a table that listed the transitions with the respective probabilities. Figure 36 displays the state machine provided.



from to	prob	from to	prob
x=0,y=6 =	1.0	x=6,y=1 =	0.2
		x=6,y=2 =	0.0
x=1,y=1 =	0.2	x=6,y=3 =	0.1
x=1,y=2 =	0.2	x=6,y=4 =	0.1
x=1,y=3 =	0.1	x=6,y=6 =	0.3
x=1,y=4 =	0.0	x=6,y=7 =	0.2
x=1,y=6 =	0.3	x=6,y=8 =	0.1
x=1,y=7 =	0.1	x=6,y=9 =	0.0
x=1,y=8 =	0.1		
		x=7,y=1 =	0.1
x=2,y=1 =	0.4	x=7,y=2 =	0.0
x=2,y=2 =	0.1	x=7,y=3 =	0.1
x=2,y=3 =	0.2	x=7,y=4 =	0.1
x=2,y=4 =	0.1	x=7,y=6 =	0.3
x=2,y=6 =	0.1	x=7,y=7 =	0.3
x=2,y=7 =	0.1	x=7,y=8 =	0.1
x=2,y=8 =	0.0	x=7,y=9 =	0.0
x=3,y=1 =	0.2	x=8,y=1 =	0.1
x=3,y=2 =	0.1	x=8,y=2 =	0.1
x=3,y=3 =	0.2	x=8,y=3 =	0.1
x=3,y=4 =	0.1	x=8,y=4 =	0.1
x=3,y=6 =	0.3	x=8,y=6 =	0.2
x=3,y=7 =	0.1	x=8,y=7 =	0.2
x=3,y=8 =	0.1	x=8,y=8 =	0.2
x=4,y=1 =	0.0	x=9,y=1 =	0.2
x=4,y=2 =	0.1	x=9,y=3 =	0.4
x=4,y=3 =	0.1	x=9,y=4 =	0.4
x=4,y=4 =	0.3	x=9,y=7 =	0.1
x=4,y=6 =	0.3		
x=4,y=7 =	0.1		

Figure 36. The Markov Chain provided with the transition probabilities

The second tool, the loss chart, displayed the blue and red losses in terms of initial state and firing times. Figure 37 displays this loss chart. The first horizontal axis depicts the state in which the system started with the first observation.



	state 1	state 2	state 3	state 6	state 7	state 8
losses red (immediate fire)	2.6	3	2	2.2	2	3
loss blue (immediate fire)	0.7	0	2	1.7	1	1
loss blue (delayed fire)	0.29	0	0	0.29	1	0
loss red (delayed fire)	3.9	4	4	3.9	4	4

Figure 37. The loss chart for the comparison of the prediction accuracy

This covered the states ‘1’ to ‘3’ in good terrain cells and ‘6’ to ‘8’ for bad terrain cells. The number of tanks observed was at most three tanks. That is why no state ‘4’ or ‘9’ is displayed in the chart. The second horizontal axis displays the firing times. The firing times chosen were “immediate” firing, which is the default option in Combat XXI, and a “delay” firing of between three and five seconds. The vertical axis depicts the blue and red losses dependent on the parameters. The loss chart demonstrates that in most cases a delay in firing was very beneficial with respect to a platoon’s own losses. The third tool that participants could use was the terrain attribute, which was displayed directly on the screen as seen in Figure 38. The horizontal lines around the “tanks” in the gray box indicate “bad” terrain-cell attributes, and the vertical lines, “good” terrain-cell attributes. The screenshot displays the positions of the blue platoon, indicates which one is currently observing, and displays the current targets in red and former observations in gray (not visible on the hard copy).

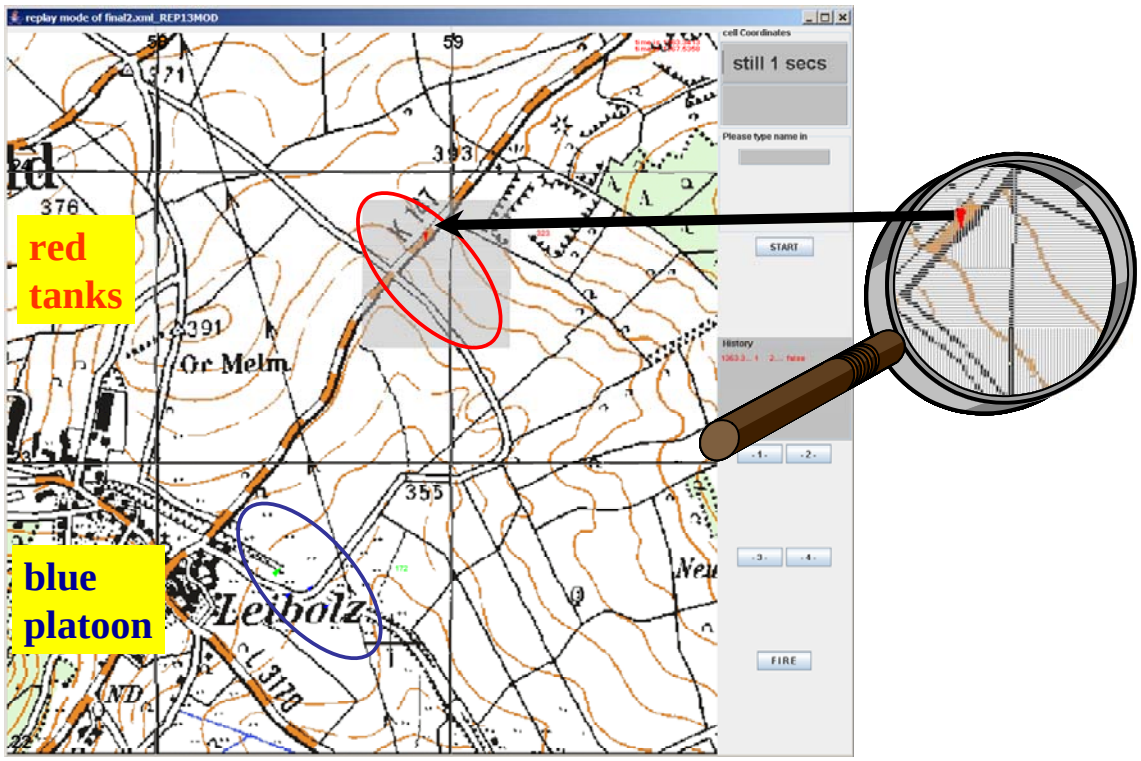


Figure 38. The GUI from the prediction comparison between the model and human subjects.

The participants, therefore, had as available tools the state machine with the transition probabilities, the loss chart from fifty runs, and the terrain attributes. With all that information provided, the participants had to predict the number of tanks in the next observation and determine when they would fire. After their predictions were entered, the next observation from the replication used was displayed. So the participants saw immediately whether or not their predictions were correct. No feedback in regard to firing times was provided. It was also of no concern whether the firing was a hit or not. The data was automatically stored and analyzed. Our method of assessing prediction accuracy is described below. Figure 39 consists of an abstract depiction of the Experiment 2 results. Its purpose is to display visually the accuracy of a human prediction. The left two columns display the prediction of the model with the two predictors (one random number draw from the Monte Carlo simulation (entitled '1x') and the mode of a hundred-times-replicated Monte Carlo simulation of the next three observations (entitled 'sequence')).

The next column denotes the scenario name with the replication and the right columns display the predictions from the participants.

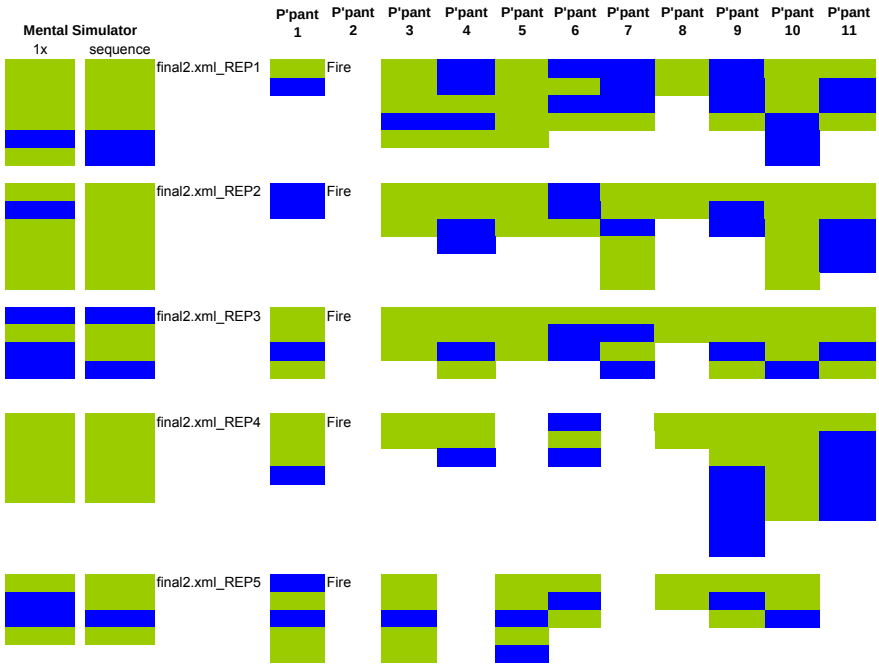


Figure 39. Results from Experiment 2

The horizontal bars indicate the prediction for the next event. “Green (gray)” means the prediction was correct while “blue (black)” indicates a wrong prediction. A prediction is said to be correct when the next state has more tanks than the current one and the predicted number is also higher than the current one. A prediction is also correct when the predicted state has an equal or fewer number of tanks and the predicted number is equal to or less than the current one. A prediction is wrong otherwise. The file names indicate which repetition of the scenario was used. The colored fields to the right of that show the predictions of the participants. The number of predictions a participant chose to make also indicates how long he waited before he started firing. Note that participant 2 always fired on first observation, and therefore made no recorded predictions. Table 5 and Table 6 quantify these results.

	first prediction human			all predictions human		
Scenario	predicted correctly	predicted wrongly	ratio	predicted correctly	predicted wrongly	ratio
final2.xml_REP1	6	4	0.60	23	18	0.56
final2.xml_REP2	8	2	0.80	29	16	0.64
final2.xml_REP3	10	0	1.00	26	9	0.74
final2.xml_REP4	6	1	0.86	20	18	0.53
final2.xml_REP5	6	0	1.00	18	8	0.69
		mean	0.85		mean	0.63
		SDev	0.17		SDev	0.09

Table 5. The prediction results from the human participants

Overall, the human participants' rate of correct predictions was 63 percent. The model yielded a success rate of 67 percent overall. However, when only the first prediction is considered, the humans scored in 85 percent of the cases, while the model achieved 80 percent.

	first prediction Model			all predictions Model		
Scenario	predicted correctly	predicted wrongly	ratio	predicted correctly	predicted wrongly	ratio
final2.xml_REP1	2	0	1.00	9	3	0.75
final2.xml_REP2	2	0	1.00	7	5	0.58
final2.xml_REP3	0	2	0.00	3	5	0.38
final2.xml_REP4	2	0	1.00	10	0	1.00
final2.xml_REP5	2	0	1.00	5	3	0.63
		Avrg	0.80		Avrg	0.67
		SDev	0.45		SDev	0.23

Table 6. The prediction results from the Mental Simulator

The data does not allow hard statistical derivations because the sample size is relatively small. But overall the data allow the conclusion that the model is acceptable and can be subject to further research.

c. Experiment 3: Prediction Accuracy Dependent on the Tools Provided

The third experiment was conducted to test the impact that the "tools" provided to the participants had on the prediction accuracy. The simulation was run again twenty times. The participants are largely disjoint from the ones of experiment 2. The GUI for the participants was still the same as in the previous experiment. In the first four replications, they started predicting without any detailed information such as terrain, the loss chart, or transition probabilities. Although the terrain features were already visible in

that run, they did not know the meaning of the lines and were told to ignore them. In the second four replications, they were provided with the three tools as in experiment 2. All participants did the replications in the same order from top to bottom.

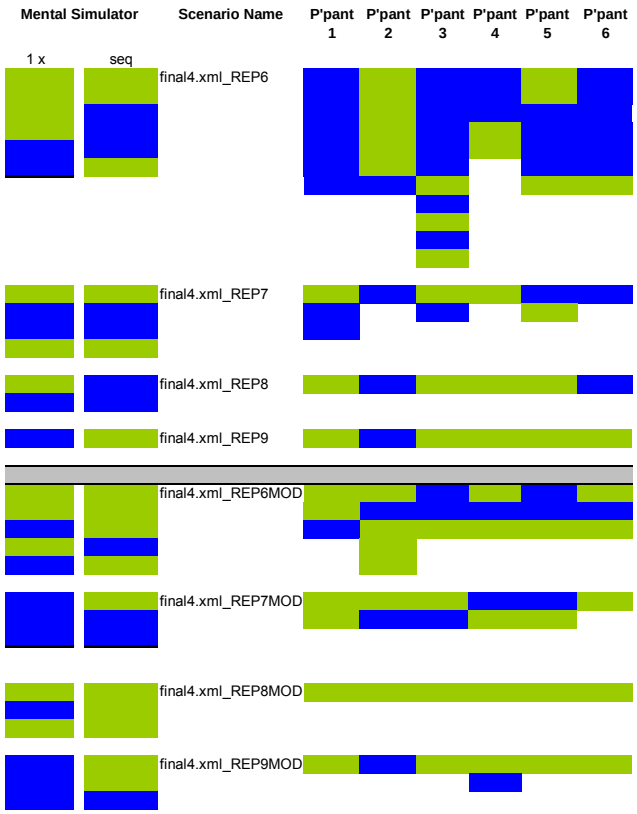


Figure 40. Results from Experiment 3

Figure 40 displays the results of this experiment qualitatively. The figure shows that in the first run the participants predicted continuously right or wrong, at least until the sixth observation. In this run the first six observations contained only one tank each. It was not always the same tank, but always only one. In the comments during the experiment the participants explained that they were waiting for more than one tank. Since the risk was assessed as relatively low, they were very likely to wait and predicted in most cases two or three tanks. Since they had no information about transition probabilities or terrain attributes, they held on to their judgment. The model, however, is able to completely employ the available transition probabilities, terrain attributes and results from the loss chart. Certainly, the graphical user interface conveying this information to

the human subject was in no way optimized through human factors engineering to transfer such understanding to the human. Moreover, whereas the model is only able to deal with information provided by the simulation processing, the human subjects brought other knowledge and experience to the experiment, such as ability to read and infer information from the tactical map background that was not represented in the simulation. In particular, the army officers examined the terrain and created expectations the model could not provide. An example for this would be the covering of other tanks while proceeding. They saw one tank and expected, depending on the terrain map, other tanks in a certain location to cover their movement. These tanks were expected in the next observation, which in most cases did not happen. Such differences between what the model based its simulation on and what the human subjects based their decisions on are open questions for further study.

Table 7 quantifies the prediction results above and displays the comparison between the participants and the mental simulator only for the first prediction.

	Scenario	only first prediction Human			only first prediction Mental Simulator		
		predicted correctly	predicted wrongly	ratio	predicted correctly	predicted wrongly	ratio
no tools	final4.xml_REP6	2	4	0.33	2	0	1.00
	final4.xml_REP7	3	3	0.50	2	0	1.00
	final4.xml_REP8	4	2	0.67	1	1	0.50
	final4.xml_REP9	5	1	0.83	1	1	0.50
			mean	0.58		mean	0.75
with tools	final4.xml_REP6MOD	4	2	0.67	2	0	1.00
	final4.xml_REP7MOD	4	2	0.67	1	1	0.50
	final4.xml_REP8MOD	5	0	1.00	2	0	1.00
	final4.xml_REP9MOD	5	1	0.83	1	1	0.50
			mean	0.79		mean	0.75
			Sdev above	0.22		Sdev above	0.29
			Sdev below	0.16		Sdev below	0.29

Table 7. Prediction Accuracy with respect to tools considering only the first prediction for the human participants and the mental simulator

Table 8 displays the prediction results from Figure 40 using all predictions.

	Scenario	all predictions Human			all first prediction Mental Simulator		
		predicted correctly	predicted wrongly	ratio	predicted correctly	predicted wrongly	ratio
no tools	final4.xml_REP6	9	16	0.36	6	4	0.60
	final4.xml_REP7	3	3	0.50	2	0	1.00
	final4.xml_REP8	4	2	0.67	1	1	0.50
	final4.xml_REP9	5	1	0.83	1	1	0.50
			mean	0.59		mean	0.65
with tools	final4.xml_REP6MOD	10	7	0.59	5	1	0.83
	final4.xml_REP7MOD	7	4	0.64	1	3	0.25
	final4.xml_REP8MOD	6	0	1.00	2	0	1.00
	final4.xml_REP9MOD	5	1	0.83	1	1	0.50
			mean	0.76		mean	0.65
			Sdev above	0.21		Sdev above	0.24
			Sdev below	0.19		Sdev below	0.34

Table 8. Prediction Accuracy with respect to tools considering all predictions for the human participants and the mental simulator.

The data shows that the participants' prediction accuracy improved by 36 percent in the first prediction case and by 15 percent in the all prediction case. The quantitative data analysis is preliminary. The data may reflect a learning effect that was not controlled for due to the fact that all participants did the runs in the same order. Performing this experiment with random ordering of the runs for each participant would minimize this effect. The data is also censored since after a firing decision was made no more predictions were done. The sample size was small due to time and resource constraints.

d. Experiment 4: Firing Behavior

The fourth experiment was conducted to compare the firing behavior of the Mental Simulator to the human participants. This experiment uses the data collected in experiments 2 and 3. There, the participants predicted the next observation and decided to fire when an observation sequence met their individual criteria for a firing decision. In experiments 2 and 3 the model did not make any firing decisions. The participants were not influenced by the Mental Simulator's behavior. In experiment 4 now, the model decided to fire according to a particular path through the implemented decision tree (see details on page 78 ff). The decision criteria in the tree are threat, prediction, terrain, and casualties expectation.

Figure 41 and following display the results from the firing comparison in a graphical way for both scenarios final2 and final4. The underlying predictions are the same as in experiment 2 and 3.

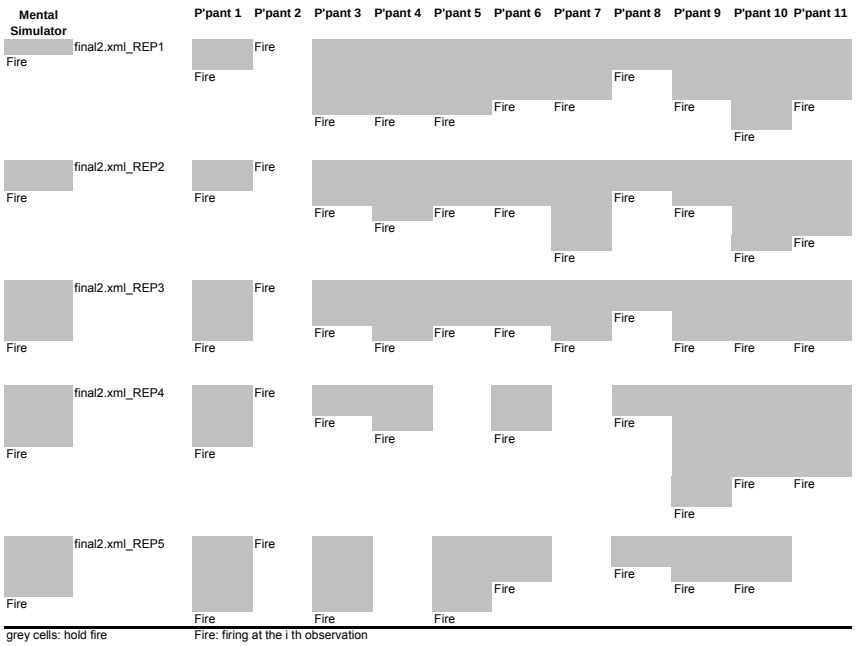


Figure 41. The comparison between human participants and the model with respect to firing decision in the scenario final2.xml

The left column indicates when the model fired, the next column denotes the run name and the remaining columns indicate when each participant fired. The word "Fire" indicates the start of the engagement process at the current observation. In other words, the last colored cell was a hold fire. A run was ended when a firing decision was made. There was no assessment whether the firing resulted in a hit or not. As already described in experiment 2, participant 2 fired always immediately. Figure 42 shows the results quantitatively. The x-axis denotes the various replications of the scenario "final2." The difference in the replications lies in the detections the Combat XXI model provides, based on the stochastic element, which is the AQUIRE algorithm. In other words, the number of tanks seen in the single observations vary. All the other parameters like mission, location, and routes remain the same. The order of the replications also represents the chronological order of the experiment. The y-axis denotes the number of observations during which the model or the human participants waited, before finally firing. The red and the blue curves, the model's behavior and the average of the human participants' behavior, respectively approach one another from replication three to five. The dashed lines above and below indicate the error range of one standard deviation per decision point.

Not depicted in the chart is the firing time chosen by the built-in (default) logic of Combat XXI, which would have always fired at the first observation.

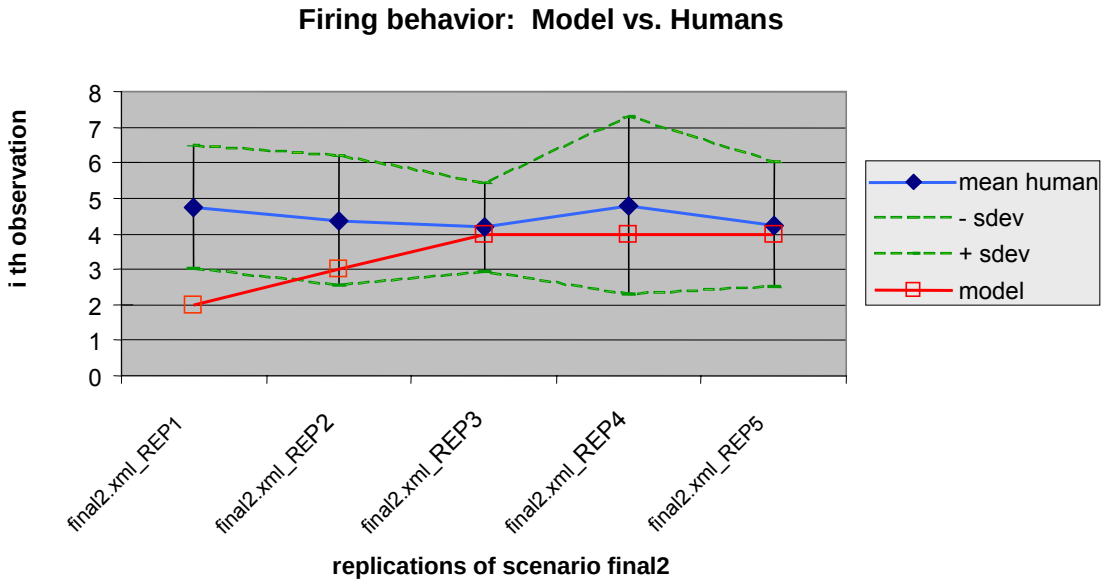


Figure 42. Firing behavior for scenario final2

The data points for replication 3 to 5 also show, when the standard deviation of the human behavior is higher, then the coupling distance also spreads. The data points for replications 1 and 2 are further apart. One could be inclined to say the model is learning from replication to replication and adjusting to the human behavior. However, it is not the case. In that respect, the model has no learning ability. The model's learning ability is incorporated in the Markov Chain and is not adjusted anymore at this stage of a scenario run. The sample size is the same as in experiment 2, but from this behavior we claim, based on preliminary data, the firing behavior is within the human range. We achieved a better result than the one for the scenario final2 for the scenario final4. Figure 43 shows the qualitative analysis for scenario final4.

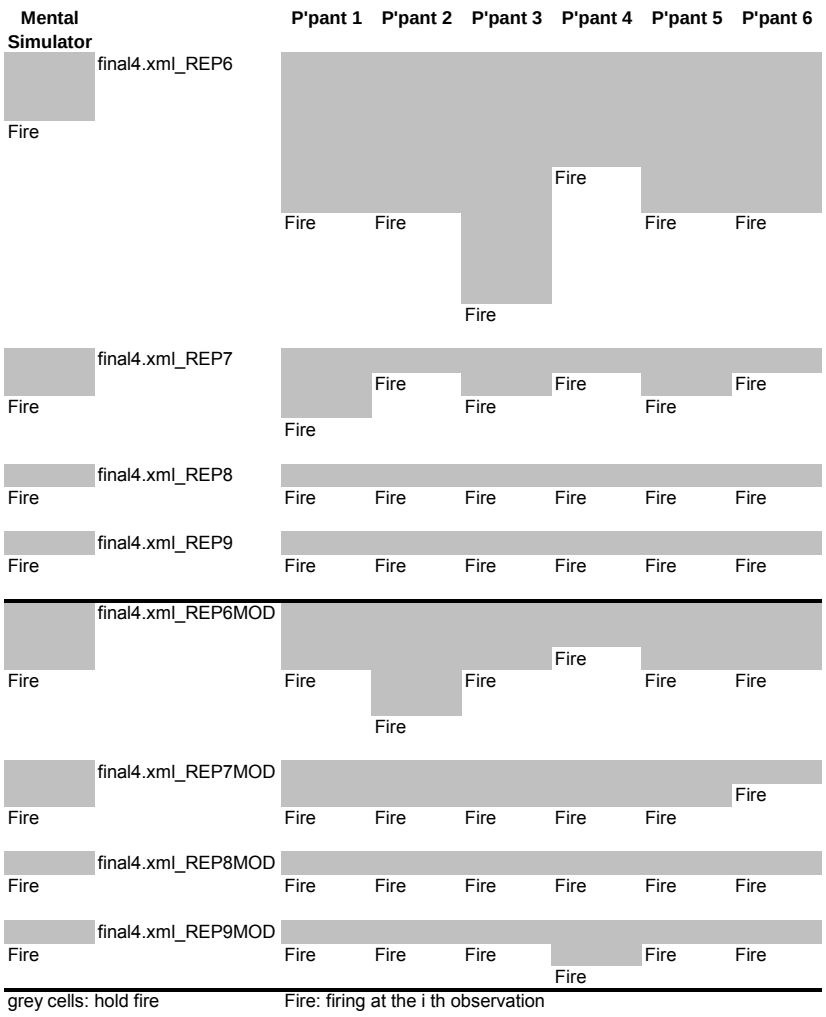


Figure 43. The firing decisions from human participants and the model with respect to tools provided (qualitative view)

Figure 44 shows the results quantitatively. The x-axis denotes the various replications of the scenario "final4." The left four replications denote the runs without tools for the participants and the right four replications with the tools provided. The y-axis indicates at what observation the human participants fired on average and in addition when the model fired. In the right four replications one can argue that the humans with the tools basically mimic the model's algorithm. However, then the left data points, REP_7 to REP_9, are hard to explain since the tools were not available to the human participants at that time. The first data point, REP_6 is explainable similarly to the prediction experiment. Having no information about transition probabilities and terrain cell attrib-

utes makes it hard to estimate. Furthermore, some participants applied their knowledge of a map this scale to their decision making process without considering that this knowledge is not incorporated in the combat simulation system. Except for the first data point, all decisions of the model to fire are within one standard deviation of the human participants' mean displayed as a yellow hyphened line.

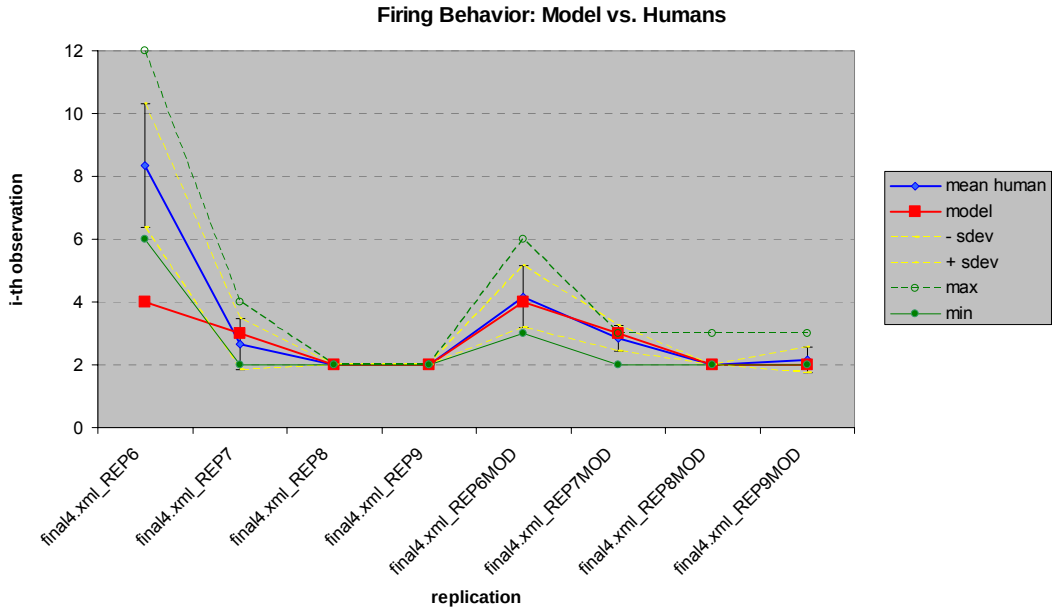


Figure 44. Firing behavior for scenario final4

The results show that not only the predictions but also the firing decisions perform in the human range. It is obvious that in none of the cases above the model immediately fired. Neither did the human participants. When in a replication the humans fired later or early then the model decided similar. The results from the experiment were not used to calibrate the model. The decision tree was developed independent of the results from the human participants. However, human tank experts were considered prior to the development of the decision tree. We consider this a favorable result for our model.

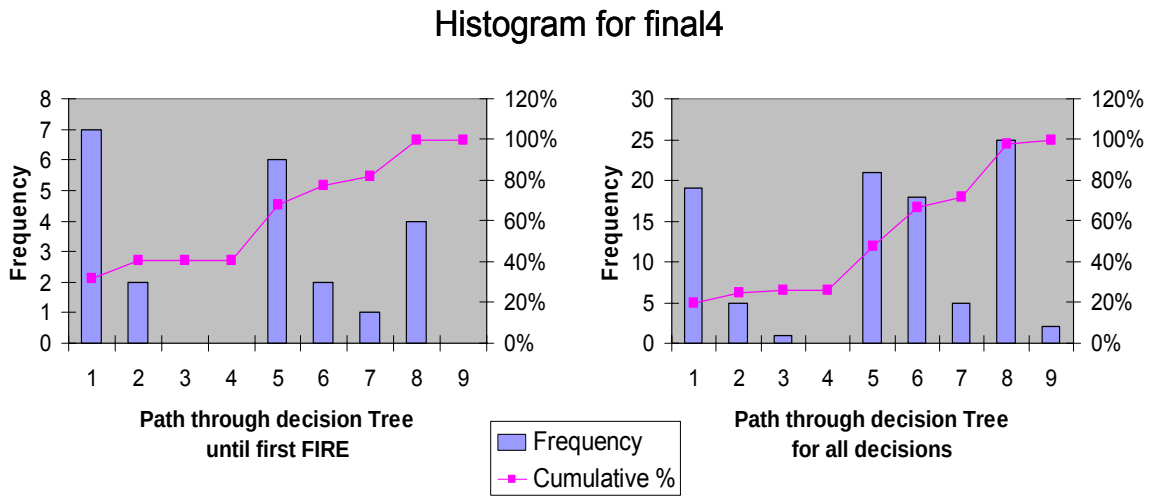


Figure 45. Histogram for the path through the decision tree in scenario final4

As a sanity check we created a histogram for the decision tree and looked how often were the various paths chosen. The left chart of Figure 45 shows the frequency of the paths that were chosen until the first firing decision was to "fire." The right chart shows all paths that were chosen until the entire replication was finished. It can be observed that eventually all paths are chosen except path 4. This is not an error. This path gets chosen when the number of tanks detected is greater than four. Path 9, when an immediate threat is determined, was eventually chosen in the later time of the replication, because the red tanks came closer and crossed the threat threshold which lead to immediate fire.

D. RESULTS

This section displays in an aggregated form the results of the preliminary prediction experiments that were conducted and shows the firing decisions made by the decision tree implementation. The results from the terrain and design experiments are in section IV.B.2.b. Terrain Attributes and in IV.C.2.a. Experiment 1. They did not involve human participation.

1. Experiment 2: Prediction Accuracy of the Model vs. Human Participants

This experiment utilized 5 different replications of the scenario final2 with eleven human participants. The task for each participant was to estimate the next observation in each replication and eventually to determine, according to one's own judgment, when to fire. The participants were equipped with the state machine and the probability distribution of the state transitions, the assessment in terms of red and blue losses of previous situations depending on whether it was fired immediately or delayed, and the terrain information. The analysis of the data collected was done in terms of how often the prediction was correct. The aggregated results are shown in Table 9.

		scenario final2	
		first prediction	all predictions
Model	mean	0.80	0.67
	sdev	0.45	0.23
Human	mean	0.85	0.63
	sdev	0.17	0.09

Table 9. The means and standard deviations for experiment 2

The differences between the model and the human participants is in both cases within 10%. The participants' firing decisions are analyzed in experiment 4.

2. Experiment 3: Prediction Accuracy in Dependence of the Tools Provided

This experiment utilized 8 different replications of a modified scenario. The difference from the former scenario lies in the greater variation in the number of observed tanks. That means that observations with a high number of tanks occurred more often. Six participants conducted this experiment. The task was similar to experiment 2 with the twist that in the first four replications, no tools were provided. In the second four replications all tools were provided. The participants from experiment 3 are disjoint with those from experiment 1. The analysis of the data collected was done in terms of how often the

prediction was correct. This was assessed again only for the first prediction of each replication and for all common number of predictions. The mental simulator of course got the tools in both cases.

Scenario final4		Human		Model	
		first prediction	all predictions	first prediction	all predictions
no tools provided	mean	0.58	0.59	0.75	0.65
	sdev	0.22	0.21	0.29	0.24
tools provided	mean	0.79	0.76	0.75	0.65
	sdev	0.16	0.19	0.29	0.34

Table 10. Results from experiment 3

Providing the tools to the participants increased their percentage of correct predictions. The participants' firing decisions are analyzed in experiment 4.

3. Experiment 4: Firing Behavior

This experiment post-processed the human firing decisions with the model's decision to fire. The model decided to fire according to a particular path through the implemented decision tree (see. page 73). The decision criteria in the tree are threat, prediction, terrain, and casualties expectation.

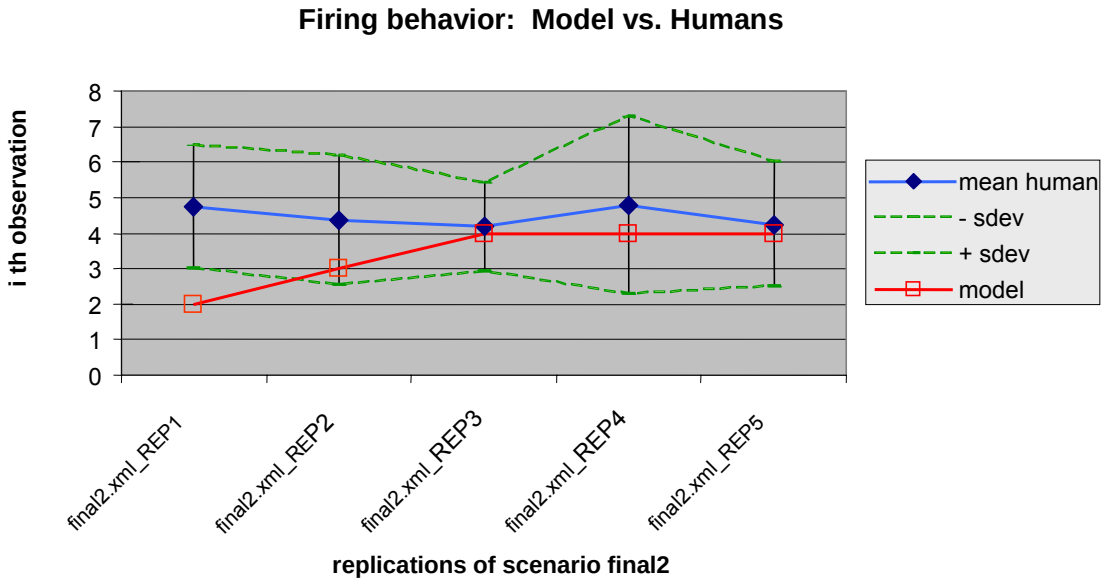


Figure 46. Firing Behavior for scenario "final2"

Figure 46 and Figure 47 show the firing behavior of the human participants in average and when the model fired in the two scenarios "final2" and "final4." Although the sample size is small, the data is censored, and the probable underlying learning effect has not been captured, with one exception each the model performs within 1 standard deviation around the mean of the human participants.

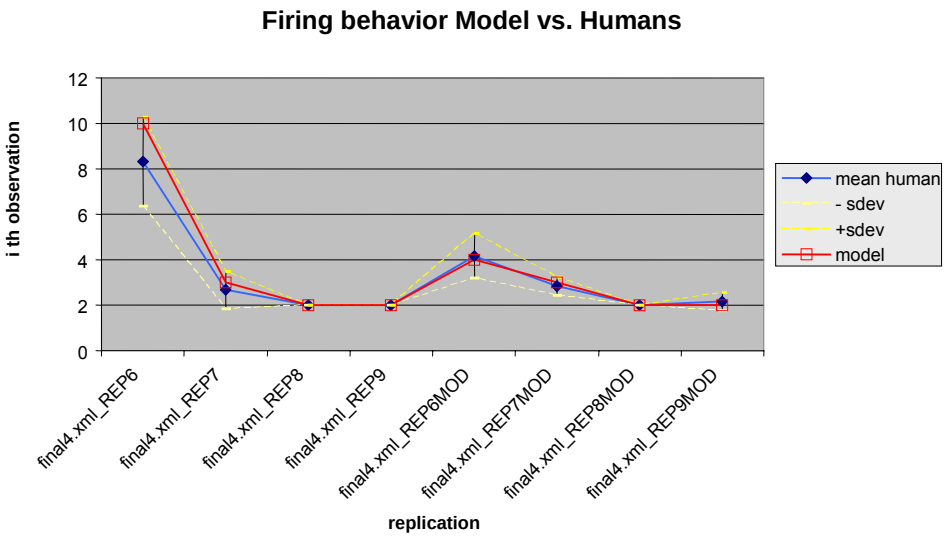


Figure 47. Firing behavior for scenario “final4”

The results show that not only the predictions but also the firing decisions perform in the human range.

V. CONCLUSIONS AND FUTURE WORK

A. CONCLUSIONS

For some combat models a simplistic model of human behavior seems to be sufficient. Sometimes errors in behavior seem to cancel one another out. One example we discussed in Chapter I was when a blue tank fires too early but the red tanks do not react accordingly and follow their original path and do not to this new situation. A real red platoon might, for example, have called in indirect fire. In simplistic behavior representations that means that the stock of artillery ammunition in the model as compared to reality is incorrect. This is bad. In order to represent more sophisticated combat situations, it is mandatory to base the decisions in the system on more accurate entity representations. This first approach to the computational modeling of mental simulation is far from being perfect or comprehensive. However, it contributes in the following way:

- Our research resulted in an implementation of the first computation model of mental simulation as described in the psychological theory of naturalistic decision making applied to entities in a simulated combat environment. We implemented a subset of mental simulation, namely projecting the past into the future, used three variables like people usually do, and provided the simulated entities with experience in order to perform mental simulation.

- Our research, based on statistical data, shows that simulated entities that are capable of "looking ahead" into the near future perform more realistically than those that do not include even knowledge of the past, but only use information of the present. Simulated behavior is considered "more realistic" when entities reason about a larger number of relevant factors (e.g. expectations), are able to adjust to sudden changes in the environment (e.g. react to an enemy that is currently not visible), and are able to use information or knowledge gained during a considered period, that is, a simulation run. Knowledge gain, also called learning, improves the overall performance of the software agent. Unlike strictly rule-based systems in which everything is predefined, prediction of the near future in our model is based on information and knowledge

that the software "learns," or derives, during the runs. This learning, or adaptive, capability, which is uncommon in combat simulation models, affords the system with greater flexibility and fine-tunes the agent for more reason-based actions. Adding a learning capability also reduces the amount of predefined data required for a run, and thus the amount of manpower effort.

- The model can be implemented as a module in the actual simulation engine. Because our mental simulation model is executed in a post processing mode, that is, after the simulation run, it uses the logged data from the simulation run as input. Since all the data required to run the model is available within the simulation model itself, it is left to the software developers to integrate it. However, it should be easy to insert into a combat model with a modular architecture. The computational cost of an add-on like this should be relatively low. Depending on the specific orientation of the mental simulation, that is, the type of events to be predicted and the nature of the behavior to be affected, it could be integrated within the interactions of certain modules. As in our scenario, where we enable the agent to predict the next tank observations, which, in turn, influences his firing behavior, the mental simulator could be inserted between the detection module and the engagement module.

- The model is applicable to a specific human decision-making moment, in our case, whether to fire or not in a given situation. Though this is a narrowly defined application in the current implementation, it is also a new and useful development in constructive simulations. The various experiments that we conducted show that the model performs within a decision-making range common to humans.

- The terrain is examined empirically in a preprocess which extends beyond merely having a line-of-sight feature. The ACQUIRE algorithm uses various parameters to determine whether a specific sensor detects a specific target. In our model, the terrain assessment, given the presence of a line-of-sight, enables entities to assess how likely it will be to detect a target in a certain terrain before the target actually arrives. To a certain degree, that ability to pre-

dict likelihood, or probability, mimics the anticipation of "undetection." This capability is important when modeling, for example, the behavior of human tank gunners in a "duel" situation, in which they monitor targets before shooting them. In known constructive combat simulation environments to date, doing this has not been possible, since the observations occur in a manner similar to a radar sweep of a certain sector. But with the terrain assessment performed in our model, an agent can address the idea that targets will go out of sight in a predictable, and thus anticipated, amount of time, rather than the agent simply recognizing eventually that the targets are gone.

B. FUTURE WORK

We consider the modeling of mental simulation in various application areas as still subject to further research. Our research is not comprehensive and the model developed is not perfect by far. However, we took the initial step of specifically addressing the mental simulation in a combat simulation environment and made room in a simulation environment for adding expectations and imagination to better imitate human behavior. We showed some possible paths for extending this approach to other application domains. We left enough room for further investigation in that direction. The results we provide are of a preliminary nature, we believe they are a useful basis for extensive and thoroughly designed experiments which were beyond the available time in this research.

Having laid out the foundation in this research, we see potential topics for follow-on work generally in any simulation that represents human behavior and in particular within combat simulation models.

One direction is to extend the experiments with a complete design of experiment approach in order to capture underlying effects and other variables. This could cover as an example the likely learning effect of participants with increasing number of replications. The experiments can also be extended to an armor school in order to increase the sample size and to include more specific armor considerations that were not addressed so far. We used among other model design elements the Monte Carlo Markov Chain simulation. There exist other predictive techniques that can be evaluated in case the state space

gets bigger and the scaling problem state machines can have, arises. We used two predictors within the mental simulator. There are other possible predictors than one random draw or taking the mode of N simulations. How the model can self-select the adequate predictor depending on the current situation could be subject to more experimentation.

During experiment 3 when no tools were provided to the human participants, it appeared that especially Army soldiers overestimated the combat simulation's ability to incorporate the terrain information. Differences between what the combat simulation model based the entities' behavior on and what the human subjects based their decisions on are open questions for further study and would improve the scope of the simulation.

There is also room for extension within the application domain of this model. So far the breadth of the model was self-limited to the capabilities of Combat XXI in order to conduct the proof-of-principle. With the continuing development of Combat XXI this research can be extended to additional functional areas and to more complex decisions. Especially, close coupling of this model with Combat XXI will allow decisions to feed back to the simulation system before the simulation continues.

We outlined already at the end of Chapter III some general applications where we can see this research beneficial. Especially, in training simulation systems where the use of avatars is on the rise, the necessity of realistic human behavior is becoming more and more important. A mental simulation component is a valuable enrichment to this.

APPENDIX: SOFTWARE STRUCTURE

All programs are written in Java (JDK 1.5). The entire model consists of three individual programs:

A. PlatoonCommander

B. GridCommander

C. GridBatchCommander

These programs run individually and independently. However, the results from the GridCommander and GridBatchCommander programs yield the empirical terrain assessment data for the PlatoonCommander. The difference between the GridCommander and GridBatchCommander program lies in the batch run mode. GridCommander is a single run solution to check intermediate results and was also the pre-version for terrain assessment. GridBatchCommander is the final version which uses the RunManager functionality from Combat XXI. The final output from the GridCommander and GridBatchCommander programs is a serialized object that is automatically read in by PlatoonCommander.

The entire PlatoonCommander program consists of 73 Java classes. This program needs an installed version of Combat XXI in order to perform all functionalities.

A. PLATOONCOMMANDER

This section gives an overview of the functionality groups and depicts how the program is used in a single run mode and in batch run mode. The complete code is not provided here. It can be requested from the MOVES-Institute at the U.S. Naval Postgraduate School in Monterey, CA.

The program can be categorized into the seven functionality groups:

1. Control (yellow)
2. GUI (green)
3. File input/ Log file reading (blue)

4. Data objects (none)
5. Batch Run Mode (purple)
6. Analysis (orange)
7. Result Display (purple-green)

1. Top Level Classes

The main class is called ReadControl. It is mutually referenced by the other top level classes that control the analysis (AnalysisManager), collect the overall transitions (StateManager), construct the main GUI including several components (ReadDisplay), read in all log files and create the objects that hold this information (DataReadManager), display the force structure (DrawReadData), display the results from the batch run mode and provide additional functionalities via GUI (BatchDataDisplay), control the batch run mode (RunManagerDriver) and read in the force structure as initial input to the system (ReadForceStructure). These classes also control the subsequent classes in the respective functionality groups. Figure 48 displays the top level classes.

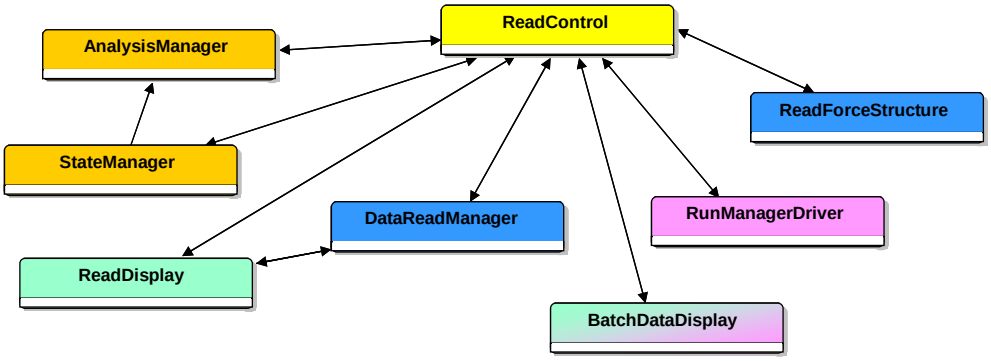


Figure 48. Top Level Classes

Spread over all functionality groups control or coordinating classes like Cubby-Hole or implemented interfaces coordinate the various threads used in this program.

2. Control Classes

This group contains the main class and the classes that synchronize and control the threads and receive the output from the combat simulation model.

CubbyHole.java

FileConstants.java

ReadControl.java (Main class)

StreamGobbler.java

The class CubbyHole is a modified class from SUN MicroSystems to coordinate and synchronize some of the threads. The class StreamGobbler is taken from available sources on the Internet. Implemented interfaces or listeners are also assigned to this functionality group.

3. GUI

This group constructs the initial front end to the user. This group includes classes that inherit from the JPanel class and classes that are needed in the display in Figure 49.

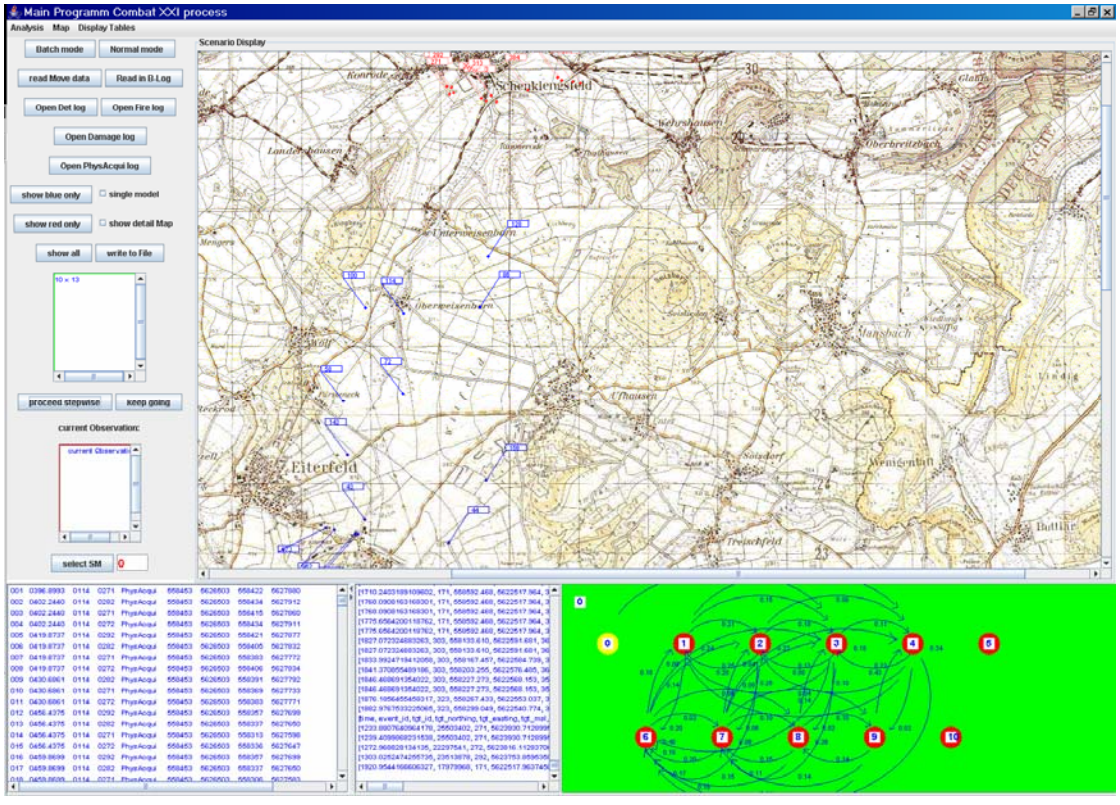


Figure 49. Screenshot of the GUI before the analysis is started with a pre-loaded state machine.

Figure 50 displays the classes (green) required to create the GUI in Figure 49.

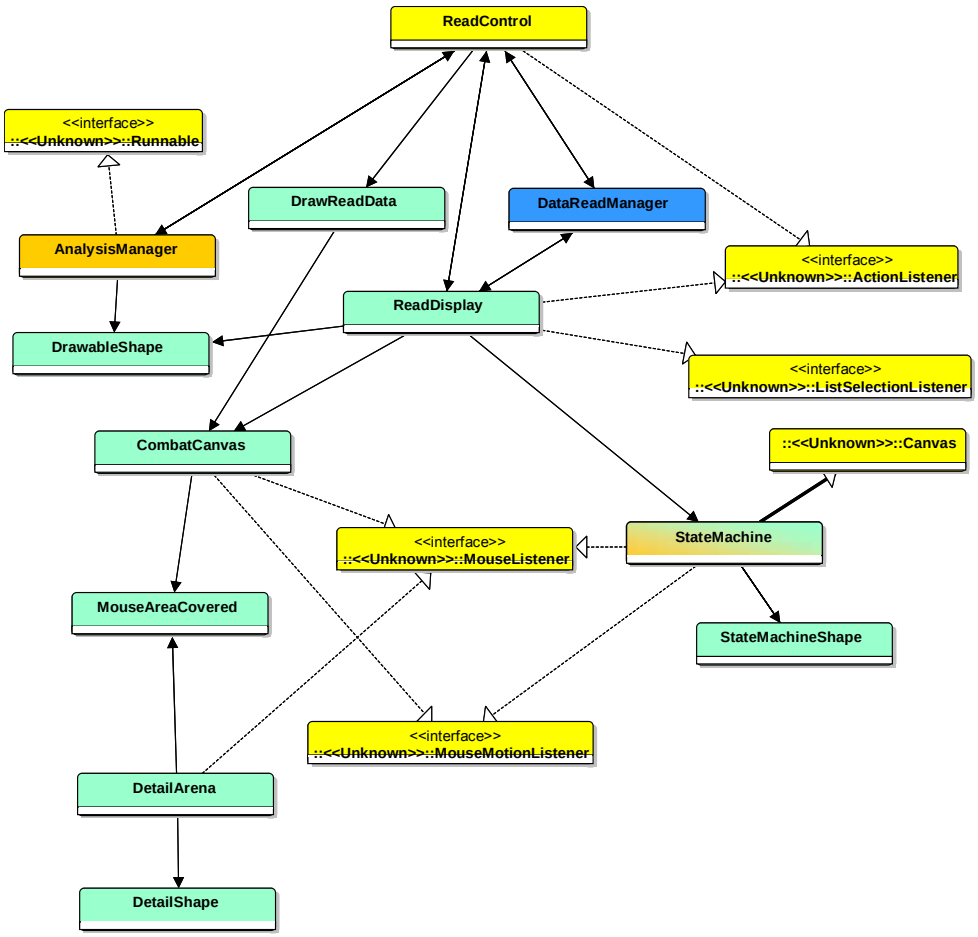


Figure 50. GUI classes

The top level class is ReadDisplay which inherits from the JPanel class and implements the ActionListener, ListSelectionListener and the FileConstants interface. The latter one holds all constants and file paths for running the program on several machines without the necessity of manually changing paths for the input and output. The classes with the ending ‘Shape’ are the objects that draw themselves on the canvas. This are the units, the labels, the states, and also the arcs and text fields from the state machine. The GUI is mouse supported and therefore, it implements various mouse listeners. The JPanel-class CombatCanvas displays the map and all entities with the identification tags. It also displays the gravitation centers used in the situational awareness component. The

Canvas class StateMachine displays the Markov Chain with the transition probabilities provided by the analysis group. The class DetailArena in Figure 50 displays the context for a decision. This panel, when desired, pops up at every decision point and displays the appropriate information, that is used within the mental simulator (see also Figure 25).

4. File input/ Log file reading

The I/O from Combat XXI to our model can work in several ways. These are 'normal mode', 'Read in B-Log', and 'Batch mode'. The Batch mode is explained later separately. Combat XXI outputs the log files differently when using the option 'Run Model' versus the RunManager version. These first two options enable the program to deal with the appropriate output-files. In both cases only the file 'spawned.log" has to be selected and the other output files are read in automatically via the DataReadManager class. The file spawned.log contains the force structure.

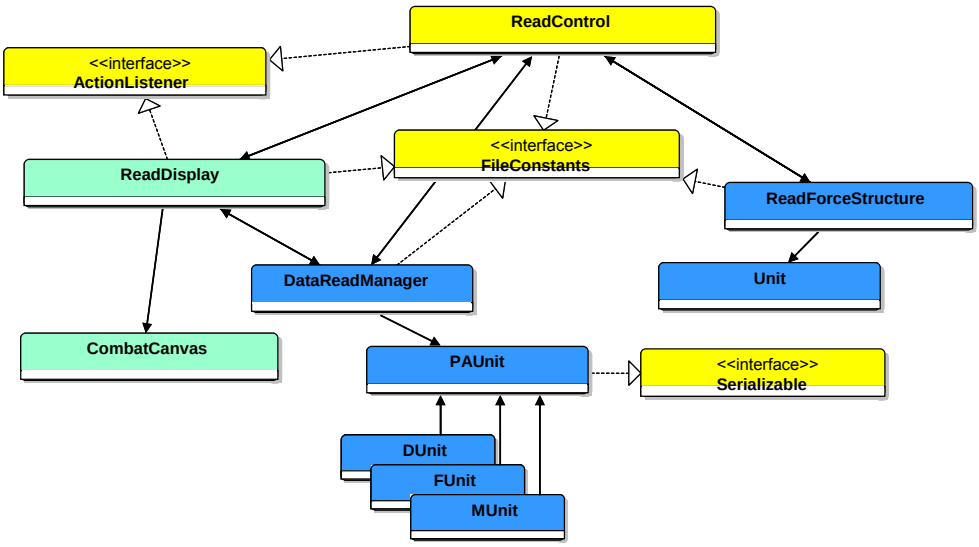


Figure 51. File I/O

The classes PAUnit, DUnit, FUnit and MUnit store the content of the various log files in objects, searchable by the simulation time. PA, D, F, and M are the abbreviations for physical acquisition (observations), damage, fire and movement logs. The class ReadForceStructure also converts and scales the UTM coordinates for the display in the

GUI. The class Unit stores the elements from the force structure log (spawned). The output from our model for permanent storage is done in ReadControl and explained in the next section.

5. Data Objects

The results from the various modes can be saved as serialized objects in Java. It is possible, for example, after a batch run of N replications, to store all relevant data necessary to replay the analysis in one data file. All objects to be stored implement the Serializable interface and can be read in via ReadControl.

The class that holds all data dynamically is LossRecordSummary. This class holds the following objects/data:

1. blue and red losses from the simulation of the various course of actions, for each run
2. decisions of the tanks over time
3. aggregated observations
4. first blue firing time
5. run name
6. remarks from the runs
7. scenario name
8. I/O path information
9. complete log files

The firing decisions are not contained in the data object. They are recreated through the BatchDataDisplay class.

6. Batch Run Mode

The program has the ability to operate in a batch mode. The top level class for running batches is RunManagerDriver. Figure 52 displays the RunManagerDriver object and its associations. The UI classes are from the Combat XXI model and modified for the specific needs.

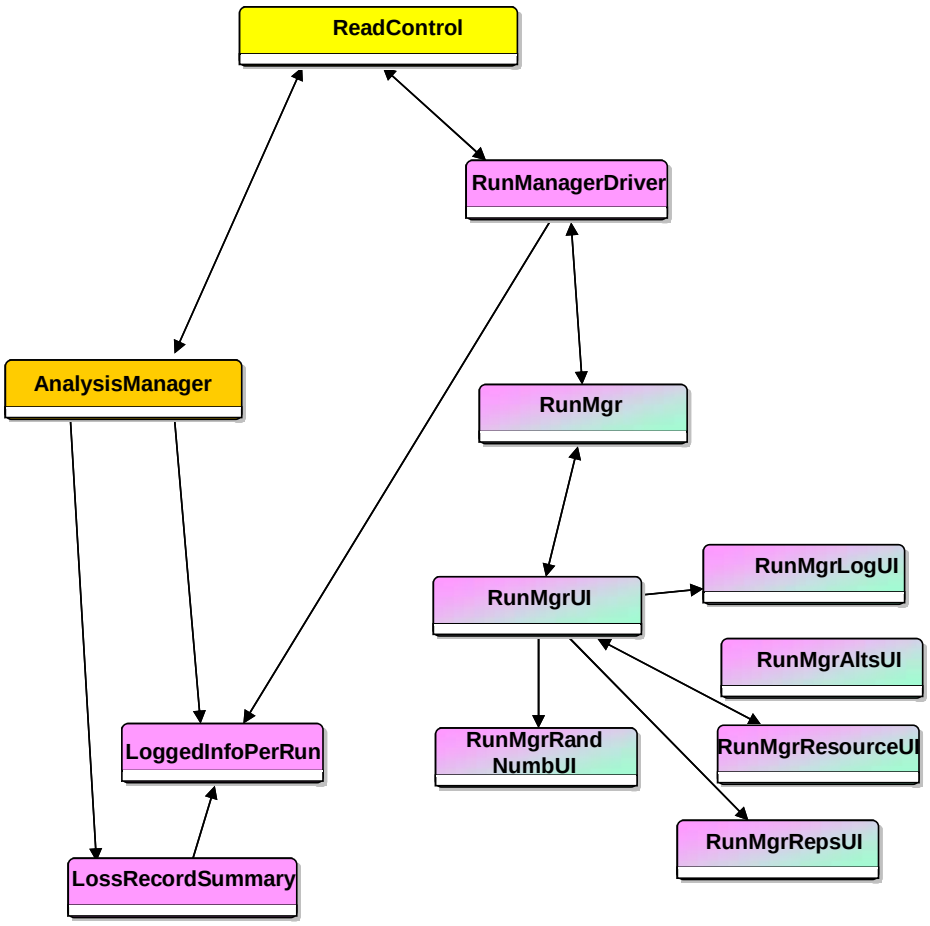


Figure 52. The RunManagerDriver object and its associations

The batch mode requires the remote call of the Combat XXI simulation model. In the non-batch mode case Combat XXI is run and then the output is read in manually. In batch mode the entire process has to be automated. What does this mean in detail? Each run in Combat XXI varies with respect to the observation sequence and the respective firing times. Therefore, a single hardwiring of the firing time or outflanking time will not work. Either they outflank too early or too late in most cases. It is necessary to adjust the firing delay time and the outflank time to each run. With this requirement, the following automated procedure, as displayed in Figure 53, has been developed.

Starting from the GUI (ReadDisplay class) the batch mode is selected ①. This event activates the runBatchMode method in ReadControl which creates a new RunManagerDriver object in a thread ②. The RunManagerDriver-object creates the GUI for the

user to enter the settings for Combat XXI. These settings include the scenario, the number of replications, the duration of the simulation run, the choice of random numbers, the desired log-files, and the location of the simulation model ③. When the user has finished the input of the settings the RunManagerDriver starts the individual replications ④. During the loop the simulation model is called twice. The first run does not apply the mental simulator ⑤, it is a regular run of the original scenario file.

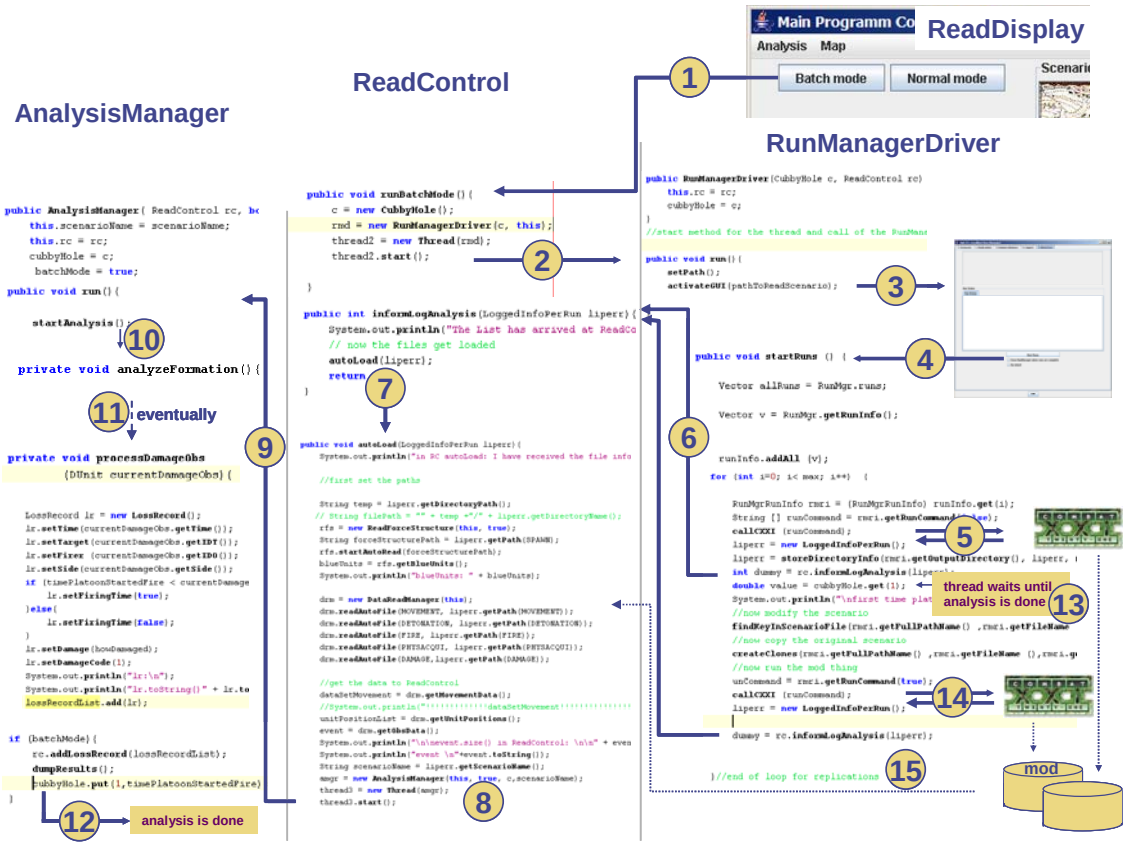


Figure 53. The top level flow in the batch mode

When Combat XXI has finished that simulation run the log files are stored in a particular folder and the information is passed to ReadControl ⑥. The RunManager-Driver-thread waits now until the analysis of this run has been processed. ReadControl activates the DataReadManager which reads in the appropriate log-files and creates the event list for the AnalysisManager ⑦. After this an object of the AnalysisManager is cre-

ated in a separate thread ⑧. The analysis of the log files starts with the creation of the situational awareness ⑨, the processing of the firing and damage events and prepares everything for the decision later on ⑩. The analysis provides also the first firing time of a blue tank which is stored and passed back to the RunManagerDriver thread ⑪. This thread continues and modifies the firing behavior rule of the scenario file ⑫ and uses it for the next run, which is the second call of Combat XXI per loop iteration ⑬. When Combat XXI has finished the run with the modified scenario, it gets also analyzed and stored ⑭. After this the next replication starts. When all runs are finished ⑮, this thread ends. ReadControl activates the class BatchDataDisplay which displays all results and provides the firing behavior to each replication. Figure 54 displays a screenshot of the BatchDataDisplay user interface.

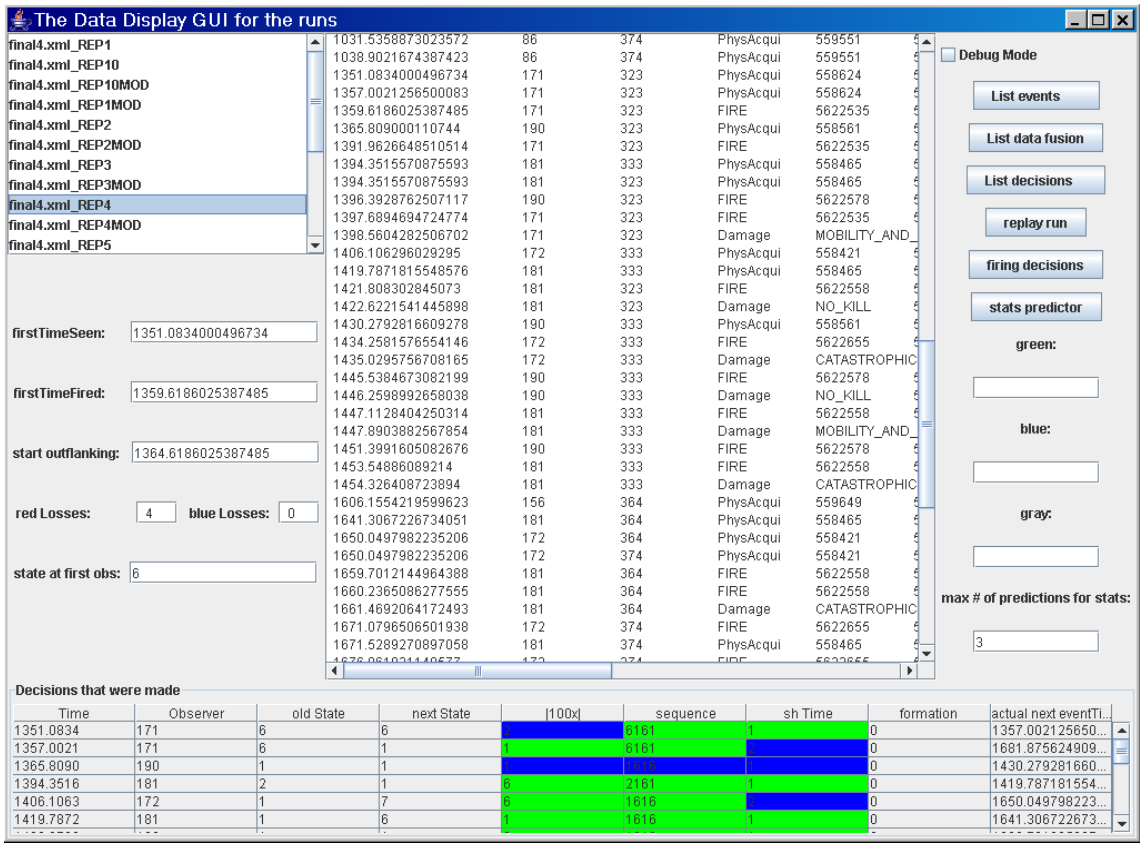


Figure 54. The GUI from BatchDataDisplay

7. Analysis

The central class of the analysis is the AnalysisManager class. The task is to create the situational awareness, provide data for the predictive model and activate the mental simulation component when a decision point occurs. It also stores the analyzed data into the data object and sends it to ReadControl. Figure 55 displays the associations with the AnalysisManager. The StateManager gathers the observations for the predictive model. In the subsequent subclasses of this object the occurred transitions are administered with respect to availability, probability, and retrieval. The objects that are displayed in the front end GUI are created and stored in a data structure to be drawn when the canvas gets refreshed. The object decision holds all predictors and creates the context for the decision later on. The object TankListStore enables the AnalysisManager to distinguish own sensors from tanks. The LoggedInfoPerRun and LossRecordSummary hold all relevant data of the analysis. The PredictionLogHandler administrates the PredictionLogStorage and retrieves past predictions and actual observations over time. The FormationBin object manages the data fusion of the observations before they are processed to the StateManager.

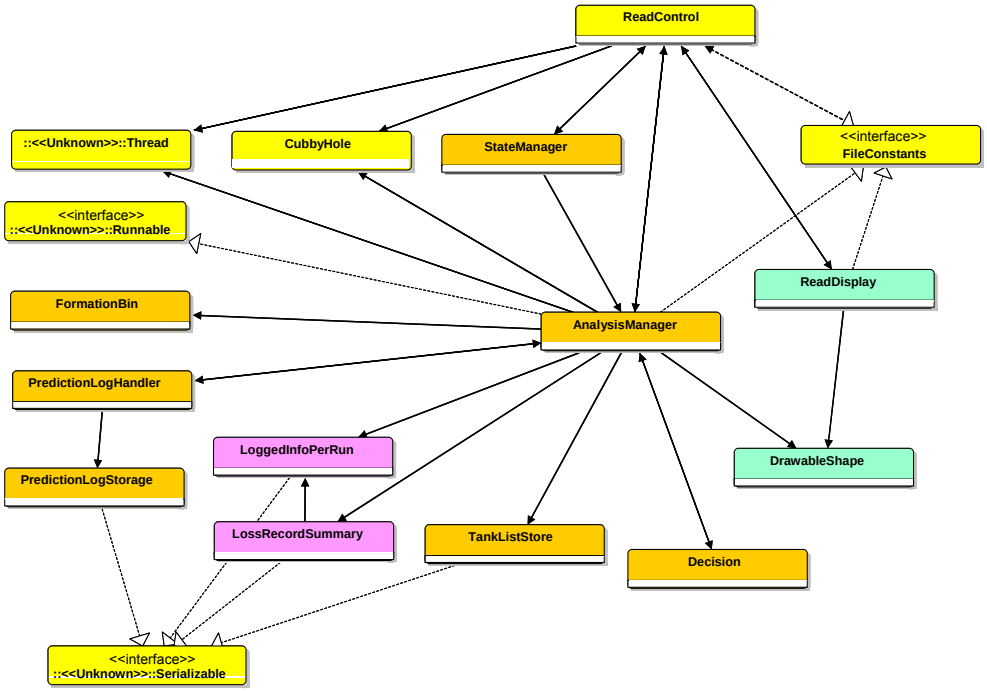


Figure 55. AnalysisManager

In all modes, either in the individual run or batch run mode, the AnalysisManager is a separate thread.

8. Result Display

The results are displayed after all the runs have been completed. Then ReadControl activates the class BatchDataDisplay. This happens also when a data set from previous runs is read in. The task of the BatchDataDisplay object is to

- enable the selection of particular runs,
- display the overall prediction behavior,
- display the firing behavior,
- provide statistical data about the predictions,
- display the observations graphically,
- provide the infrastructure for experiments,
- verify the data fusion,
- display important time points, and
- display the loses on blue and red side.

Figure 56 displays the BatchDataDisplay – object with its associations. The class FiringDecision holds the decision tree. At each decision point the log contains the time, the current observation, the prediction and whether the decision was to fire or to hold fire. The FiringDecision object also accesses the outcomes from previous simulations with the parameters decision, initial state and terrain cell attribute.

ReplayFrame is the top level class for the replay of replications and for conducting experiments. The results from the participants predictions and firing decisions are stored in ReplayExpRun and of all participants in ReplayExpRunSummary. The results are not exportable into a serialized object. They have to be copied into an editor and saved separately. The experiments are reproducible for each participant.

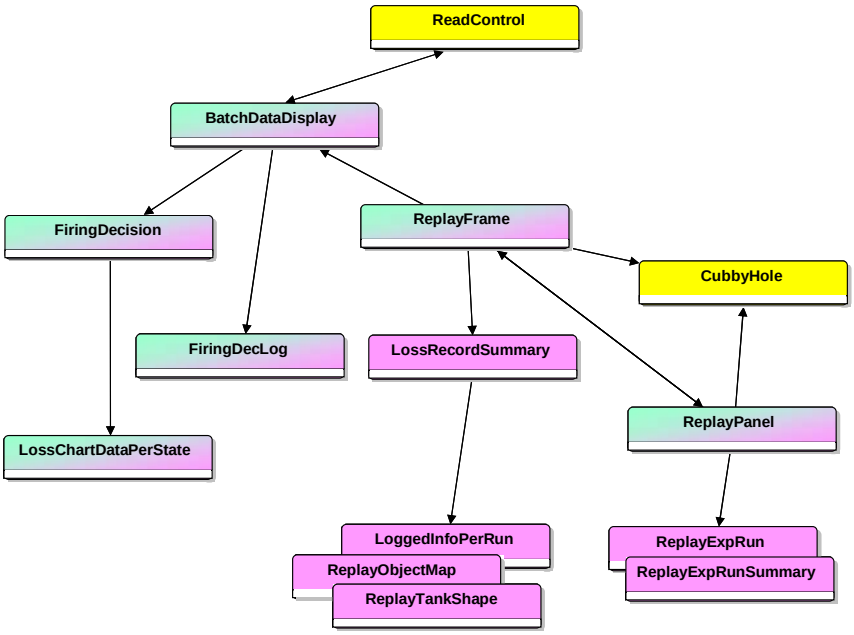


Figure 56. The class BatchDataDisplay and its associations

B. THE GRIDCOMMANDER PROGRAMS

The GridBatch- and the GridCommader programs differ only by the batch run functionality. We describe in this section the GridBatchCommander and rely on the reader's ability to transfer this also to the GridCommander program. Both programs use the basic infrastructure from the PlatoonCommander program.

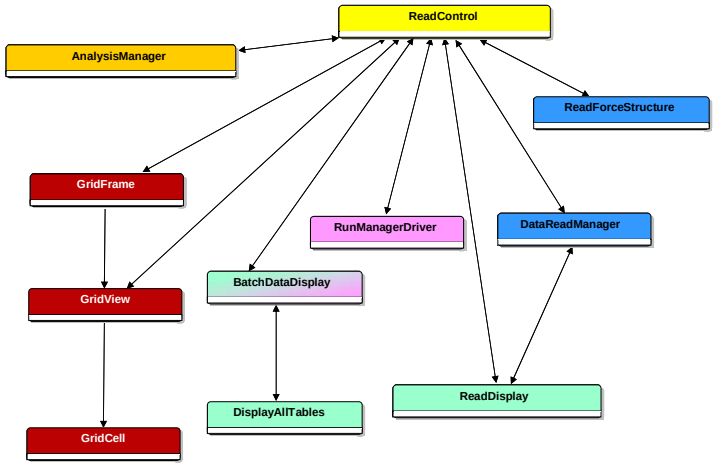


Figure 57. Embedding of the new Grid classes

Most of the Java class names are the same as in PlatoonCommander, however the classes as displayed in Figure 57 are not identical, and therefore, not interchangeable. The top level flow as depicted in Figure 58 is similar to the PlatoonCommander. It differs slightly within the RunManagerDriver, because Combat XXI is called only once per repetition, and it differs major at the end of the analysis in the AnalysisManager.

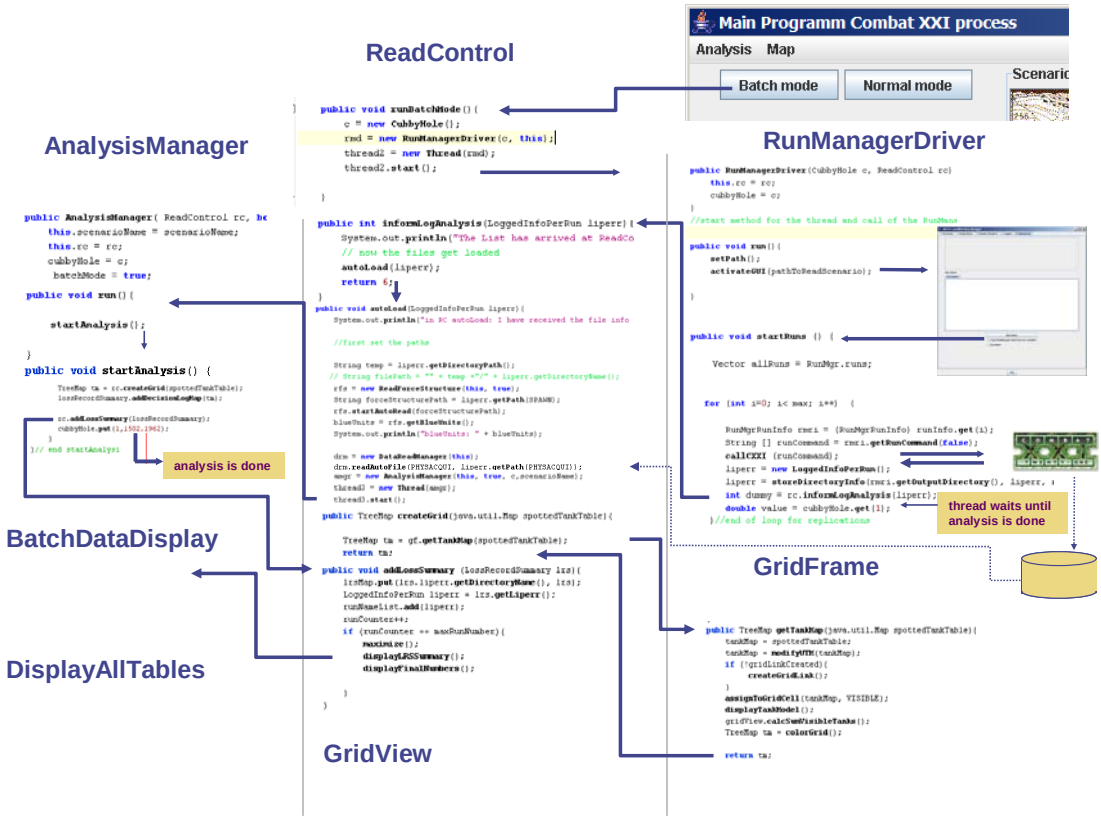


Figure 58. Top level flow in GridBatchCommander

When the analysis of the observations is finished then the data structure with the processed detections is sent via ReadControl to GridFrame. In GridFrame the coordinates are transformed for the display, the detected tanks are assigned to the 100m x 100m terrain cells, the colors of the cells are determined, and the grid model is populated. The grid model keeps all information about the cells, like real coordinates, canvas coordinates, the maximum number of tanks per cell, the number of detected cells, and the cell attribute (color). Before the AnalysisManager thread reports its termination to the RunManager-Driver, the complete data set is sent to ReadControl. When all replications the user re-

quested are done, then ReadControl activates BatchDataDisplay to list the individual detection maps. The gridData object is finally saved to the C:\ drive and can be read in automatically by ReadControl from PlatoonCommander.

LIST OF REFERENCES

ACQUIRE Range Performance Model for Target Acquisition Systems. (1995). Version 1 User's Guide, U.S. Army CECOM Night Vision and Electronic Sensors Directorate Report, Ft. Belvoir, VA.

Albers, M.J. (1999). Information design considerations for improving situation awareness in complex problem-solving. Proceedings of the 17th annual international conference on computer documentation, 1999, New Orleans, Louisiana.

Aumann, Yonatan., Etzioni,Oren., Feldman, Ronen., Perkowitz, Mike and Shmiel, Tomer. (1998). Predicting event sequences: Data mining for prefetching webpages. In submitted to KDD'98, March 1998.

Aven, T. (2002), On the Implementation of the Bayesian Approach in reliability and Risk Analyses, Third International Conference On Mathematical Methods In Reliability, June 17-20, 2002, Trondheim, Norway

Barlow, R. E., Proschan, F. & Hunter, L. C. (1965). Mathematical Theory of Reliability. New York: John Wiley & Sons, Inc.

Bilmes, J. & Zweig, G. (2002). The Graphical Models Toolkit: An Open Source Software System for Speech and Time-Series Processing. IEEE Int. Conf. on Acoustics, Speech, and Signal Processing, June 2002. Orlando, Florida.

Boné, R., Crucianu,M. (2002). Multi-step-ahead prediction with neural networks: a review. Publication de l'équipe RFAI, 9èmes rencontres internationales « Approches Connexionnistes en Sciences Économiques et en Gestion ». 21-22 Novembre 2002, Boulogne sur Mer, France. pp. 97-106.

Bose, S.K., (2001). Introduction to Queuing Systems. Kluwer/Plenum Publishers

Bouchaffra, D., Koontz, Krpasundar, E. V and ' Srihari, R.K. (1996). Incorporating diverse information sources in handwriting recognition postprocessing, in International Journal of Imaging Systems and Technology, special issue, John Wiley, Vol. 7, Issue 4, Winter 1996.

Box, G. E. P., Jenkins, G. M., and Reinsel, G. C. (1994). Time Series Analysis, Forecasting and Control, 3rd ed. Prentice Hall, Englewood Clifs, NJ.

Box, G.E.P, Jenkins, G.M. (1976). Time Series Analysis : Forecasting and Control. Revised Edition. San Francisco, California: Holden-Day.

Brezovic, C. P., Klein, G. A., & Thordsen, M. (1987). Decision making in armored platoon command (Contract MDA903-85-C-0327 for U.S. Army Research Institute, Alexandria, VA). Fairborn, Ohio: Klein Associates Inc.

Brockwell, P. J., and Davis, R. 1996. Introduction to Time-Series and Forecasting. Springer-Verlag.

Cioppa, T.M., Willis, J.B., Goerger, N.D., Brown, L.P. (2003). Research Plan Development For Modeling And Simulation Of Military Operations In Urban Terrain. Proceedings of the 2003 Winter Simulation Conference WSC 2003, December 7-10, 2003, New Orleans, LA.

Cohen, M. S., Freeman, J., Wolf, S., & Militello, L. G. (1995). Training metacognitive skills in Naval combat decision making (Technical Report under contract N61339-92-C-0092 for the Naval Air Warfare Center, Training Systems Division, Orlando, FL.). Arlington, VA: Cognitive Technologies, Inc.

Committee on Technology for Future Naval Forces, National Research Council Technology for the United States Navy and Marine Corps, 2000-2035 Becoming a 21st-Century Force: Volume 9: Modeling and Simulation. National Academy Press Washington, D.C. 1997

Davies, M & Stone, T. (Eds) (2001). Mental Simulation, Tacit Theory, and the Threat of Collapse. Final version (March 2001) of a paper to appear in a special issue of Philosophical Topics in honor of Alvin Goldman. URL = <http://www.lsbu.ac.uk/psycho/teaching/pdf/Mental-Simulation.pdf>.

Davison, B. D. (2002). The Design and Evaluation of Web Prefetching and Caching Techniques Ph.D. dissertation. Department of Computer Science, Rutgers University, New Brunswick, NJ. October 2002.

Deviren, M. & Daoudi, K. (2001).Structural Learning of Dynamic Bayesian Networks in Speech Recognition. In Eurospeech.

DMSO. (2005). Defense Modeling and Simulation Office, Online M&S Glossary (DoD 5000.59-M). URL = https://www.dmso.mil/public/resources/glossary/results?do=get&def=297 (last accessed November/2005).

DoD. (1995). Department of Defense, Modeling and Simulation Master Plan. URL = http://www.dmso.mil/briefs/msdocs/policy/msmp.pdf (last accessed November/2005).

Dodd, Lorraine., Moffat, J,. Smith J,. Mathieson, G., From simple prescriptive to complex descriptive models: an example from a recent command decision experiment: Proceedings from the 8th International Command and Control Research & Technology Symposium National Defense University, Washington. 17-19 June 2003.

Domeniconi C., Perng C., Vilalta R., Ma S. (2002). A Classification Approach for Prediction of Target Events in Temporal Sequences. Proceedings of the 6th European Conference on Principles and Practice of Knowledge Discovery in Databases (PKDD'02), Helsinki, Finland.

Dunsmore, A., & Roper, M. A., (2000). Comparative Evaluation of Program Comprehension Measures, Technical Report EFoCS-35-2000, University of Strathclyde, UK.

Ehrlich, W. K., Emerson, T. J., (1987). Modeling software failures and reliability growth during system testing, Proceedings of the 9th international conference on Software Engineering, p.72-82, March 1987, Monterey, California, United States

Ehrlich, W. K., Stampfel, J. P., Wu, J. R., (1990). Application of software reliability modelling to product quality and test process, Proceedings of the 12th international conference on Software engineering, p.108-116, March 26-30, 1990, Nice, France.

Endsley, Mica. (1995). Toward a theory of situation awareness in dynamic systems. Human Factors. 37.1 32 -64.

Ferguson, J. D. (1980). Variable duration models for speech. In Proc. Symposium on the Application of Hidden Markov Models to Text and Speech, pages 143-179.

Field Manual 17-15. (1996). Field Manual 17-15. TANK PLATOON. Headquarters, Department of the Army Washington, DC, April 3, 1996

Findeisen, R. & Allgoewer, Frank. (2002). An Introduction to Nonlinear Model Predictive Control. Proceedings of the 21st Benelux Meeting on Systems and Control Veldhoven, The Netherlands, March 19-21, 2002.

Finke, R.A. (1989). Principles of Mental Imagery. Cambridge, MA: MIT Press.

Forsythe, C. & Xavier, P. (2002). Human emulation: Progress toward realistic synthetic human agents. Proceedings of the 11th Conference on Computer-Generated Forces and Behavior Representation, Orlando, FL. 257-266.

Forsythe, C. (2004). The Future of Simulation Technology for Law Enforcement. FBI Law Enforcement Bulletin: Volume 73, Number 1. Federal Bureau of Investigation, Washington, D.C. January 2004.

Ganesh J Pai., Joanne Bechta Dugan. (2001). Enhancing Software Reliability Estimation Using Bayesian Networks and Fault Trees. 12th International Symposium on Software Reliability Engineering, Hong Kong, November 27-30,2001.

Gharamani, Z. (1997). Learning Dynamic Bayesian Networks. to appear in Giles and Mori: Adaptive Processing of Temporal Information. Lecture Notes in Artificial Intelligence. Springer Verlag.

Gibbs, Gary P. & Cabell, Randolph H. (2000). Experimental Evaluation of Controller Complexity in the Active Control of Turbulent Boundary Layer Induced Sound Radiation from Panels. 6th AIAA/CEAS Aeronautics Conference, June 12-14, 2000, Westin Maui, Maui, Hawaii.

Gordon, Robert M., (2001). Folk Psychology as Mental Simulation. The Stanford Encyclopedia of Philosophy (Spring 2001 Edition). Edward N. Zalta (ed.), URL = <http://plato.stanford.edu/archives/spr2001/entries/folkpsych-simulation/>.

Hausrath, A. H. (1971). Venture Simulation in War, Business and Politics. McGraw-Hill, New York, 1971.

Hovd, M. (2004). A brief introduction to Model Predictive Control. URL = http://www.itk.ntnu.no/fag/TTK4135/viktig/MPCkompendium%20HOvd.pdf

Huang, Q., (2003) Hållmats, J., Wallenius, K., Brynielsson, J., Simulation-Based Decision Support for Command and Control in Joint Operations.

Hutchins, S. G. (1996). "Principles for Intelligent Decision Aiding." Naval Command, Control, and Ocean Surveillance Center, RDT&E Division. Technical Report 1718. San Diego, CA.

Ilachinsky, A. (2004). ARTIFICIAL WAR: Multiagent-Based Simulation of Combat. World Scientific Publishing, Singapore.

Kaempf, G. L., Wolf, S., Thordsen, M. L., & Klein, G. (1992). Decision making in the AEGIS combat information center (Contract N66001-90-C-6023 for the Naval Command, Control and Ocean Surveillance Center, San Diego, CA). Fairborn, H: Klein Associates Inc.

Kahneman, D. & Tversky, A.. (1982). Judgment Under Uncertainty: Heuristics and Biases. Cambridge University Press. Cambridge, NY

Klein Associates Inc. (2003). KA PUBS 2003. URL= http://www.decisionmaking.com/approach/publications.html (last accessed November/2005).

Klein, G. (1999). Sources of Power: How People Make Decisions. Cambridge MA, MIT Press.

Klein, G. A. (1997). The recognition-primed decision (RPD) model: Looking back, looking forward. In C. Zsambok & G. Klein (Eds.), Naturalistic Decision Making. Mahwah, NJ: Erlbaum.

Klein, G., & Crandall, B. W. (1995). The role of mental simulation in problem solving and decision making. In P. Hancock (Ed.), Local applications of the ecological approach to human-machine systems, Volume 2: Resources for ecological psychology (Vol. 2, pp. 324-358). Mahwah, NJ: Lawrence Erlbaum Associates.

Kosslyn, S. M.: Mental Imagery. In: Visual Cognition - An Invitation to Cognitive Science Volume 2. Publ..: S. M. Kosslyn, D. N. Osherson. 2nd Edition., Massachusetts 1995, S. 267-296.

Kuck, I. (2003). Warfare Simulation: Status and Issues for Space, Parts 1-5. Air Force Research Laboratory, Directed Energy Directorate, U.S. Air Force Materiel Command, Kirtland AFB, N.M., Report AFRL-DE-TR-2003-1037.

Lebeck A. R., & Wood D. A., (1994). Cache Profiling and the SPEC Benchmarks: A Case Study. IEEE Computer, 27(10):15-26, October 1994.

Liang, Y., F. Robichaud, B. J. Fugere, and K. N. Ackles. (2001). Implementing a Naturalistic Command Agent Design. In Proceedings of the Tenth Conference on Computer Generated Forces, May 15-17, Norfolk, VA, pp 379-386.

Liehr, S., Pawelzik, K., (1999). Hidden Markov gating for prediction of change points in switching dynamical systems. In Proceedings of the Seventh European Symposium on Artificial Neural Networks, 21-22-23 April, Bruges, Belgium, pp 405-410.

Manning, Christopher D. & Schütze, Hinrich. (1999). Foundations of Statistical Natural Language Processing. Cambridge, MA: MIT Press.

Maybeck, P.S. (1979). Stochastic Models, Estimation, and Control. Volume 1. Academic Press, New York.

Minsky, M. L. 1986. The Society of Mind. Simon and Schuster, New York.

Mitchell, T. M. (1997). Machine Learning. McGraw-Hill, Boston, MA.

Modeling and Simulation: Linking Entertainment and Defense, Committee on Modeling and Simulation: Commission on Physical Sciences, Mathematics, and Applications. (1997). National Research Council. National Academy Press, Washington, D.C.

MSIAC. (2005) Modeling and Simulation Information Analysis Center. URL = http://www.msiac.dmso.mil/ (last accessed November/2005).

MSRR. (2005). Modeling and Simulation Resource Repository. URL = http://www.msrr.dmso.mil/ (last accessed November/2005).

Murphy, K. (2002). Dynamic Bayesian Networks: Representation, Inference and Learning. PhD thesis, Dept. Computer Science, UC Berkeley.

National Research Council Washington Dc Naval Studies Board (2000). Network-Centric Naval Forces: A Transition Strategy for Enhancing Operational Capabilities. Commission on Physical Sciences, Mathematics, and Applications (CPSMA):p. 56.

NATO. (1998). NATO Modeling and Simulation Master Plan. URL = ftp://ftp.rta.nato.int/Documents/MSG/NMSMasterPlan/NMSMasterPlan.pdf (last accessed November/2005).

NDM. (2005). Proceedings of the Seventh International NDM Conference (Ed. J.M.C Schraagen), Amsterdam, The Netherlands, June 2005.

Nigel, Thomas. Mental Imagery, The Stanford Encyclopedia of Philosophy (Winter 2001 Edition), Edward N. Zalta (ed.), URL = <http://plato.stanford.edu/archives/win2001/entries/mental-imagery/>.

Nigel, J.T. Thomas. (2003). Mental Imagery, Philosophical Issues About, in the Encyclopedia of Cognitive Science. Volume 2, pp. 1147-1153 - Editor in Chief, Lynn Nadel. London: Nature Publishing/Macmillan.

NIST/SEMATECH (2004). e-Handbook of Statistical Methods, URL = http://www.itl.nist.gov/div898/handbook/ (last accessed November/2005).

Norling, E., L. Sonenberg, and R. RÄonnquist. 2000. Enhancing Multi-Agent Based Simulation with Human-Like Decision Making Strategies. In Multi-Agent Based Simulation: Proceedings of the Second International Workshop, MABS 2000, Boston, MA, Springer, pp 214 - 228.

Patchett, C., Venkat. V.S.Sastry., Michael Bathe. (2003). The Performance of an Intelligent Agent in a Simulated Air Combat Environment. BRIMS 2003.

Patra, Susantra. (2003). A Neural Network Approach For Long-Term Software MTTF Prediction. FastAbstract ISSRE 2003. URL =

http://www.chillarege.com/fastabstracts/issre2003/129-FA-2003.pdf (last accessed November/2005).

Pedersen, M.N. (1999). A study of the practical significance of word RAM algorithms for internal integer sorting. Department of Computer Science University of Copenhagen, Denmark URL = http://www.diku.dk/forskning/performance-engineering/Publications/pedersen99.ps (last accessed November/2005).

Pew, R. W. and A. S. Mavor. (1998). Modeling Human and Organizational Behavior: Application to Military Simulations. National Academy Press, Washington D. C.

Pylyshyn, Zenon W. (2002). Mental Imagery: In search of a theory. Behavioral and Brain Sciences, 2002, 25(2), 157-237.

Qin, S.J. and Badgwell, T.A. (1996). An overview of industrial model predictive control technology. In J.C. Kantor, C.E. Garcia, and B. Carnahan, editors, Fifth International Conference on Chemical Process Control – CPC V, pages 232–256. American Institute of Chemical Engineers, 1996.

Rabiner, L. (1989). A Tutorial on Hidden Markov Models and Selected Applications in Speech Recognition. Proceedings of the IEEE, Vol. 77, No. 02, February 1989.

Rabiner, L.R., Juang B.H., (1986). An Introduction to Hidden Markov Models. IEEE ASSP Magazine, January 1986.

Ross, S.M., (1993). Introduction to Probability Models. 5th Edition, Academic Press Inc., San Diego.

Russell S, Norvig P. (2003) Artificial Intelligence: A Modern Approach. 2nd Edition, Prentice Hall, Upper Saddle River, New Jersey.

Sanna, L. J. (2000). Mental simulation, affect, and personality: A conceptual framework. Current Directions in Psychological Science, 9, 168–173.

Sanna, Lawrence J., & Meier, S. (2000). Looking for Clouds in a Silver Lining: Self-Esteem, Mental Simulations, and Temporal Confidence Changes. Journal of Research in Personality, 34 (2): 236.

Shepard, R. N., & Metzler, J. (1971). Mental rotation of three dimensional objects. Science, 171, 701-703.

Shooman, M. L., (1987).Yes, software reliability can be measured and predicted, Proceedings of the 1987 Fall Joint Computer Conference on Exploring technology: today and tomorrow, p.121-122, December 1987, Dallas, Texas, United States

Sokolowski, J. A. (2003). Modeling the Decision Process of a Joint Task Force Commander**.** Ph.D. Dissertation, Old Dominion University, Norfolk, VA. May 2003.

Sokolowski, J.A. (2002). "Can a Composite Agent be Used to Implement a Recognition-Primed Decision Model?," In Proceedings of the Eleventh Conference on Computer Generated Forces and Behavioral Representation, Orlando, FL, May 7-9 2002. pp. 473-478.

Sprinkle, J., C. vanBuskirk, and G. Karsai. (2000). Modeling Agent Negotiation. In IEEE International Conference on Systems, Man, and Cybernetics, October 8, 2000. Nashville, TN, pp 454-459.

Sutton, R. (2000). Toward Grounding Knowledge in Prediction or Toward a Computational Theory of Artificial Intelligence.
URL = www.anw.cs.umass.edu/~rich/Talks/CEC2000/CEC2000.ppt (last accessed November/2005).

Tambe, M., and Rosenbloom P. S., (1996). Architectures for agents that track other agents in multi-agent worlds. In M. Wooldridge, J. P. Muller, and M. Tambe, editors, Intelligent Agents Volume II, Lecture Notes in Artificial Intelligence, pages 156-170. Springer-Verlag.

Tani, J., Model-Based learning for mobile robot navigation from the dynamical systems perspective. IEEE Transactions on Systems, Man, and Cybernetics, 26:421-436, 1996.

Taylor, Shelley E. and Sherry K. Schneider (1989), Coping and the Simulation of Events. Social Cognition, v. 7, n. 2, pp. 174-194.

Taylor, Shelley E., Lien B. Pham, Inna D. Rivkin, and David A. Armor (1998). Harnessing the Imagination: Mental Simulation, Self-Regulation, and Coping. American Psychologist, v. 53, n. 4, pp. 429-439.

Thunholm, P., (2000). ETT STEG MOT EN FÖRESKRIVANDE MODELL FÖR MILITÄRT - TAKTISKT BESLUTSFATTANDE. 5th Conference on Naturalistic Decision - Making, Tammsvik, Sweden May 26 - 28, 2000.

VIC. (2005). Vector in Commander. URL = http://www.msrr.army.mil/index.cfm?top_level=ORG_A_1000094&taxonomy=ORG (last accessed November/2005).

Vilalta R. and Ma Sheng. (2002). Predicting Rare Events in Temporal Domains. Proceedings of the 2002 IEEE International Conference on Data Mining (ICDM'02), Maebashi Japan.

Warwick, W., S. McIlwaine, R. Hutton, and P. McDermott. (2001). Developing Computational Models of Recognition-Primed Decision Making. In Proceedings of the Tenth Conference on Computer Generated Forces, May 15-17, Norfolk, VA, pp 323-331.

Warwick, W., S. McIlwaine, R. Hutton, and P. McDermott. (2002). Developing Computational Models of Recognition-Primed Decisions: Progress and Lessons Learned. In Proceedings of the Eleventh Conference on Computer Generated Forces, May 7-9, 2002, Orlando, FL.

Weiss, Gary M., (1999). "Timeweaver: a Genetic Algorithm for Identifying Predictive Patterns in Sequences of Events". Proceedings of the Genetic and Evolutionary Computation Conference (GECCO-99), Morgan Kaufmann, San Francisco, CA, 718-725.

Weiss, Gary M. & Hirsh, Haym. (1998a). "Learning to Predict Rare Events in Event Sequences", Proceedings of the Fourth International Conference on Knowledge Discovery and Data Mining (KDD-98), AAAI Press, Menlo Park, CA, 359-363.

Weiss, Gary M. & Haym Hirsh (1998b). "Event Prediction: Learning from Ambiguous Examples". Presented at the 1998 Neural Information Processing Systems (NIPS) Workshop on Learning from Ambiguous and Complex Examples.

Whelan, Joseph G. (2001). MIT System Dynamics in Education Project.

Zsambok, Caroline E.. (1997). Naturalistic decision making: Where are we now? In Caroline E. Zsambok and Gary Klein, editors, Naturalistic Decision Making, pages 3–16. Lawrence Erlbaum Associates.

INITIAL DISTRIBUTION LIST

1. Defense Technical Information Center
   Ft. Belvoir, VA

2. Dudley Knox Library
   Naval Postgraduate School
   Monterey, CA

3. Dr. Christian J. Darken
   Department of Computer Science
   Naval Postgraduate School
   Monterey, CA

4. Dr. Rudolph P. Darken
   Director, MOVES Institute
   Naval Postgraduate School,
   Monterey, CA

5. LTC Dr. Tom Cioppa
   Naval War College
   Newport, RI

6. Dr. Thomas Otani
   Department of Computer Science
   Naval Postgraduate School,
   Monterey, CA

7. Professor John Hiles
   MOVES Institute
   Naval Postgraduate School,
   Monterey, CA

8. Oberstleutnant Dietmar Kunde
   MOVES Institute
   Naval Postgraduate School,
   Monterey, CA