Techniques for Modeling Human Performance in Synthetic Environments: A Supplementary Review — Source link

Frank E. Ritter, Nigel Shadbolt, David G. Elliman, Richard M. Young ...+2 more authors

Institutions: Pennsylvania State University

Published on: 01 Jun 2003

Topics: Cognitive model, Work related, Context (language use) and Usability

Ritter, F. E., Shadbolt, N. R., Elliman, D., Young, R., Gobet, F., & Baxter, G. D. (2001, in press). Techniques for modeling human performance in synthetic environments: A supplementary review. Wright-Patterson Air Force Base, OH: Human Systems Information Analysis Center.

Previously cited as Ritter, F. E., Shadbolt, N. R., Elliman, D., Young, R., Gobet, F., & Baxter, G. D. (1999). Techniques for modelling human performance in synthetic environments: A supplementary review (Tech. Report No. 62. ESRC CREDIT, Dept. of Psychology, U. of Nottingham.

Techniques for modeling human performance in synthetic environments: A supplementary review

Frank E. Ritter[1], Nigel R. Shadbolt[2], David Elliman[3],

Richard Young[4], Fernand Gobet[3], Gordon D. Baxter[6]

13 May 2001

[1]Penn State, [2]University of Southampton [3]University of Nottingham

[4]University of Hertfordshire [4]University of York

Point of contact:

Frank E. Ritter
School of Information Sciences and Technology
The Pennsylvania State University
State College, PA 16801
+1 (814) 865-4453
+1 (814) 865-5604 (FAX)
ritter@ist.psu.edu

Abstract

We summarize selected recent developments and promising directions for improving the quality of models of human performance in synthetic environments. The potential uses and goals for behavioral models in synthetic environments are first summarized. Within that context, we examine relevant, current work related to modeling more complete performance, for example, on cognitive modeling of emotion, advanced techniques for testing and building models of behavior, new cognitive architectures, and agent and Belief, Desires and Intentions (BDI) technology. The report also considers the usability of these systems as an important but neglected aspect of their performance. A list of projects with high payoff for modeling human performance in synthetic environments is noted.

Acknowledgments

This work was supported by DERA (Bedford, UK) under contract LSA/E20307, and by DSTO (Australia). Stephen Croker and Peter Lonsdale provided useful comments and helped assemble this report. In addition to the workshop participants listed, we thank Christina Bartl, Andrew Lucas, Robin Hollands, Nicholas Howden, Ralph Rönnquist, and Colin Sheppard for their help or comments. The conclusions reported here are solely the responsibility of the authors.

Table of Contents

Listing of Acronyms and abbreviations

1. Tasks and Objectives for Modeling Behavior in Synthetic Environments (SEs)

There are now numerous models of human behavior in synthetic environments, and they serve a multitude of uses. It is worthwhile reconsidering where and how to improve these models to provide more realistic human behavior.

This report provides a more recent review of work than Pew and Mavor (1998), and provides a more detailed source of further ideas and suggestions. We particularly draw the reader's attention to the importance of integration of models and the usability of models. We extend Pew and Mavor's results by examining a few architectures (e.g., Cogent, Jack) that were not included or available when Pew and Mavor compiled their report, and summarizing several promising areas for further work that have arisen since their report went to press.

This report directly reflects the biases and specific expertise of the authors as they attempt to identify a wide range of potential problems and provide possible solutions. Some of the proposed projects are high risk and not all of the authors agrees that these projects are doable. We could all agree, however, that if possible they would be rewarding. Given the diversity of human behavior, there remain many issues not covered here. For example, many aspects of teamwork are important but not covered. Most of the systems and architectures reported here are continually evolving. Because of the rapid pace of development in this area, our reviews may underestimate the capabilities of these systems, and several of our suggestions may already be incorporated in them.

1.1 The role of synthetic forces

The possible expected uses and potential goals for the synthetic forces programme is quite diverse. There are several commonly acknowledged uses of synthetic forces. These uses have included at least the range shown in Table 1. This is a wide set. Pew and Mavor (1998) focused on the application of synthetic forces to training, partly because the major applications and successes have been with respect to training. Further uses have been outlined in other reviews (Computer Science and Telecommunications Board, 1997; Lucas & Goss, 1999; Synthetic Environments Management Board, 1998).

Table 1. The potential uses of models of behavior in synthetic force environments.

- Training leaders,
- Joint and combined training,
- Training other personnel (e.g., support and logistics),
- Testing existing doctrine,
- Testing possible future procurements,
- Testing new doctrine, and perhaps even
- Serving as a formal, runnable description of doctrine.

The user community for synthetic forces would be better served if all these uses were supported by a single system or approach. Currently, the models of behavior in these systems have often been developed without a long-term plan, and are only usable within the simulation they were developed for. Historically, few single systems have supported more than one or two of the uses noted in Table 1. This is wasteful and can lead to different behaviors being taught or used in different simulations when they should be exactly the same behavior. The use of the DIS (distributed interactive simulation) protocol for distributed simulation is a step towards integration, but it does not apply to behavior itself.

While having a single system or approach is highly desirable, there are good reasons why multiple systems are currently used (in addition to a multitude of bad reasons as well). Perhaps the most important reason why models of behavior currently vary is that existing approaches to modeling cannot support all of the uses in Table 1 equally well. Models that focus on aggregate, or large unit behavior, do not support low-level simulations very well. Models that predict average behavior are much less useful for practicing tactics and procedures. Models that are good for training provide detailed data that have to be extensively summarized and aggregated to be of use for planners. Planners and evaluators, for example, may find useful data in large simulations such as Purple Link, but such simulations cannot yet be convened within an afternoon or even a week to examine how a new platform performs. This report will attempt to make suggestions on all of these levels, but does not intend to be comprehensive.

1.2 Definition of terms

There are several terms used in this report that have meanings specific to the domain of behavioral modeling. The term *model*, for example, will refer exclusively to cognitive models, and the term *simulation* will refer exclusively to task simulations. We review these terms here, starting by introducing synthetic forces. We briefly explain ModSAF to provide a common system as a point of reference. We then define the terms we will use with respect to models of behavior.

1.2.1 Synthetic forces

Synthetic forces exist in military simulations, sometimes alongside real forces that have been instrumented and placed in the simulation. There are now synthetic force simulations covering all of the armed services. Synthetic forces can be separated into two components, the physical aspects and the behavioral aspects. The physical aspects represents the movement and state of platforms (objects) in the simulation, including such aspects as maximum speed and the set of actions that can be performed in the world. The physical aspects provide constraints on behavior. Physical are fairly complete now for most purposes, although they remain important in their own right (Computer Science and Telecommunications Board, 1997; Synthetic Environments Management Board, 1998).

The behavioral aspects of a synthetic force platform determine where, when, and how it performs the physical actions, that is, its behavior. Many human and entity behaviors can be simulated, such as movement and attack, but behavior has been less veridically modeled than physical performance. The next step to increase realism is not only to include further intelligent behavior but also to match more closely the timing and sequence of human behavior when performing the same tasks.

1.2.2 ModSAF

Modular Semi-Automated Forces (ModSAF) is a system for simulating entities (platforms) on a simulated battlefield (Loral, 1995). It is perhaps the most widely used behavioral simulator in military synthetic environments. The goal of ModSAF is to replicate the behavior of simulated platforms in sufficient detail to provide useful training and simulation of tactics.

ModSAF includes the ability to simulate the most common types of physical platforms, such as a tank, and external effects on those platforms like weather and smoke. The terrain is defined in a separate database, which is shared by other simulators in the same exercise using the DIS simulation protocol. Multiple platforms can be simulated by a single ModSAF system.

The local platforms interact with remote platforms by exchanging approximately twenty different types of information packets. Examples of packet types include: announcing where the platform is (the other platforms compute whether the originator can be seen), radar being emitted, and shots being fired. Thus, the features of the packets vary. Each simulation is responsible for updating its own position and computing what to do with the information in each packet. So, for example, a tank does not directly shoot another tank. Attackers send out projectile packets, and the target tank computes that it would be damaged by their projectiles.

Some semi-intelligent behaviors are included in ModSAF through a set of about twenty different simple scripts. These scripts support such activities as moving between two points, hiding, and patrolling.

ModSAF is a large system. It can be compiled into several major versions, include versions to test networks, and specific versions for each service. The terrain databases each include up to 1 gigabyte of data. Simulating multiple entities required in 1999 a relatively fast workstation (100 MHz+) with a reasonable amount of memory (32 MB+).

A major problem is usability. ModSAF is large and has a complicated syntax. Users report problems learning and using it. A better way to provide its functionality needs to be found, or its usability needs to be improved directly.

1.2.3 Frameworks, theories, models, and cognitive architectures

It is common in cognitive science to differentiate between several levels of theorizing (e.g., Anderson, 1983; 1993, Ch. 1) and defining these levels now will help us in the remainder of this report. *Framework* refers to the specification of a few broad principles, with too many details left unspecified to be able to make empirical predictions. For example, the idea that human cognition acts as a production system offers a framework for studying the human mind.

Theory adds more precision to frameworks, and describes data structures and mechanisms that at least allow qualitative predictions to be made. For example, the production system principles presented in Newell and Simon (1972) together form a theory of human cognition.

Models are theories implemented as computer programs or represented mathematically to apply to specific situations or types of situations. While generally more limited in their domain of application than theories, models typically also provide more accurate, quantitative predictions.

Cognitive architecture has two meanings: (a) specifications of the main modules and mechanisms underlying human cognition; and (b) the computer program implementing these specifications. These meanings are separate and distinct, but usually are used as equivalent. Cognitive architectures, as proposed by Newell (1990),, offer a platform for developing cognitive models rapidly while keeping the theoretical coherence between these models intact. These cognitive architectures are often seen as equivalent to Unified Theories of Cognition, a way to pull all that is known about cognition into a single theory. In Appendix 3 we include brief descriptions of ACT-R and Soar.

There exists no generally agreed definition of *hybrid architectures*. Some use the term when a cognitive architecture includes symbolic features (e.g., a production system) as well as non-symbolic features (e.g., neural net spreading of activation among memory elements); others, such as Pew and Mavor (1998), use the term when two or more architectures of any kind are combined (e.g., Soar and EPIC). We use the latter definition in this report.

When comparing theoretical proposals, it is essential to keep in mind the level at which the proposals were formulated: typically, a framework will cover a large amount of empirical regularities without specifying many details, while a model will cover a small amount of data with great precision. It is generally agreed that models are more useful scientifically than theories or frameworks, because they make clear-cut predictions that can be tested with empirical data, and hence are less amenable to ad hoc explanations (Popper, 1959). Models are, however, harder to create and use.

1.3 Structure of this report

Section 2 provides a summary review of Pew and Mavor (1998). This provides the context for much of the remainder of the report. Section 3 on providing more complete performance reviews amplifications, updates, and additions to Pew and Mavor's list of psychological regularities that should be included in models of human behavior. Section 4 notes problems integrating models with simulations as well as integrating them with each other to make larger more complete models. Section 5 takes up the issues surrounding usability of behavioral models. Usability of the models themselves was considered to be outside the scope of Pew and Mavor's report (1998, p. 10). We will argue that usability of these models is not only desirable but necessary for the success of modeling itself. Section 6 considers techniques and cognitive architectures for modeling human behavior in synthetic environments with respect to the aims of the previous two sections. Section 7 concludes with a list of projects to address problems identified in the previous four sections.

2. Summary of Pew and Mavor

While the reader of this report is likely to have seen Pew and Mavor's (1998) *Modeling Human and Organizational Behavior*, we review it briefly here to provide background for readers not familiar with it, and to provide some useful context.

2.1 Their brief

In their 1998 book Pew, Mavor, and their panel review the state of the art in human behavior representation as applied to military simulations, with an emphasis on the areas of cognitive, team, and organizational behavior. Their book is based on a panel that met for 18 months, and drew extensively on a wide range of researchers. It is available as a hard cover book, as well as online <pompeii.nap.edu/bo_book/0309057477/html/>.

In their book, Pew and Mavor looked not just at representing behavior, but also at methods for generating it. They provide a review of the uses of models of behavior in synthetic environments. They include a review of the major synthetic environments in use by the US military. These environments by example describe the range of current and potential uses and levels of simulation.

The book provides a useful summary of integrated (cognitive) architectures. It is comprehensive enough and clear enough that we have used it to teach undergraduate students. The summary includes a table comparing the architectures. We will attempt to apply the same table to review several additional architectures.

The book then reviews the important areas to modeling human behavior in synthetic environments. This is a very wide range, encompassing nearly all of human behavior. The book reviews attention and multitasking, memory and learning, human decision making, situation awareness, planning, behavior moderators (such as fatigue and emotions), organizational (small group) behavior, and information warfare (e.g., how the order of information presentation influences decision making). The book concludes with a framework for developing models of human behavior followed by a set of conclusions and recommendations. Each of these reviews is clearly written and limited only by the space they are allowed. The reviews are quite positive, suggesting that major aspects of behavior are either already being modeled, or can and will be modeled within a few years. This is in stark contrast to an earlier similar review, which could only note open questions (Elkind, Card, Hochberg, & Huey, 1989).

2.2 What *Modeling Human and Organizational Behavior* does well

Pew and Mavor's book is a useful and, we believe, seminal book for psychology and modeling. The book is useful because the reviews it provides, while they could be extended, are unusually clear and comprehensive, covering the full range of range of relevant behavior. They would serve as a useful textbook for professionals in other areas to teach them the current results and current problems in these areas of psychology and in modeling.

The book is seminal because the authors lay out a complete review of cognition that is widely usable. While their review is similar to Newell's (1990) or Anderson's (1998) reviews, Pew and Mavor's review is not situated within a single architecture; the result is a more global but only slightly less directed view.

The reviews of the models and of the data to be modeled together, because of their scope and potential impact constitute a call to arms for modelers of synthetic forces. The juxtaposition of the data and ways to model it is enticing and exciting. This approach of modeling behavior will significantly influence psychology in general if the modeling work continues to be successful. Models of synthetic forces in the near future will subsume enough general psychology data that they will simply represent the best models in psychology.

2.3 Where *Modeling Human and Organizational Behavior* can be improved

There are surprisingly few problems with Pew and Mavor's review. However, they do not review all of the possible regularities of human behavior. We will be able to add a few important aspects, and provide further arguments to support many of their main conclusions. They could have referenced, for example, Boff and Lincoln (1986) for a wide ranging list of existing general regularities in perception and performance. In the area of human decision making, Dawes' (1994) review is also valuable. They do not appear to cite a quite relevant report on how this type of modeling is also being developed as entertainment (Computer Science and Telecommunications Board, 1997), and, not surprisingly, they are unaware of a concurrent similar UK review (Synthetic Environments Management Board, 1998).

On a high level and early on, they explicitly note that they will not review the usability of behavioral models. We will argue that improved usability is necessary for these models to achieve their potential.

They did not have the space to review all the integrative (cognitive) architectures. While it would be unfair to call this book dated at this point in time, there are already a few architectures worth considering that were not available to them.

They do not dwell on the ability to describe human behavior, focusing instead on how to generate it. There remains some need to be able to describe the behavior before generating it, which we will take up below.

Finally, they did not have the space to lay out very detailed projects to fulfill their short-, medium-, and long-term goals. We provide a more detailed but still incomplete set.

3. Current Objective: More Complete Performance

There are a wide range of behaviors that have yet to be incorporated into existing models. Included in this list are numerous additional relevant regularities available about human behavior (see Boff & Lincoln, 1986, for a subset). The question that must be addressed is which behaviors are the most important and most accessible to incorporate. We note here several of the most promising or necessary behaviors to be included next in models of human performance based on our experiences and previous work.

The suggestions we make later tend to be based on the individual. Much of the behavior being modeled currently in synthetic environments is different because it is based on the platoon or higher level and is aggregated across time or situations. As smaller time scales and more intricate and fine-grained simulations are developed and used, such as for modeling urban terrorism, the behavioral issues noted here will become more important.

We start with learning. While Pew and Mavor include learning as a useful aspect of performance, we believe learning to be essential. We also expand the case for including models of working memory, perception, emotions, and erroneous behavior. We then can examine higher-level aspects of behavior to be considered, starting with integration of models and ending with information overload.

3.1 Learning

Learning is mentioned as important in several ways by Pew and Mavor (1998). Learning (i.e., training) is the largest role of the military in peace time (i.e., rehearsal, p. 30); it is essential for multitasking behavior (pp. 114-115); in its own right it is an important aspect of human behavior (Ch. 5); and it is important within groups (Ch. 10). We cover learning again here.

Pew and Mavor mention several of the advantages of learning, but do not make a complete case for its inclusion. There are several additional advantages that they do not mention or that we can emphasize. Tactics are influenced by learning. There is a home-field advantage. Working within your own territory, because you know it, makes additional tactics feasible and provides generally improved performance. (Working within your own territory would also provide some additional motivation.)

Including learning in models would provide a mechanism for producing different levels of behavior. Experienced troops, for example, would not be different not in some numeric way in that they react faster (although this is probably true), but in a more qualitative way in that they know more and use different strategies. Learning modifies, constrains, and supports the use of computer interfaces (Rieman, Young, & Howes, 1996); similar effects may be found in exploring terrain and implementing tactics in new geographic spaces.

Programming -- that is, creating the model directly -- may be too difficult. It may be the case that it is easier for models to learn behaviors than for these behaviors to be programmed directly. This argument has been put forward by connectionist researchers for some time.

Theoretical work in this area of learning has direct implications for training within the military and within schools. Models that learn can be used to understand and optimize learning (Ohlsson, 1992). If we can program models to learn, the behavior and knowledge that result may be different from the initial knowledge that the system started with, or from the expert performance that we currently teach. This final knowledge may be useful for teaching. In the case of photocopying (Agre & Shrager, 1990), for example, better strategies arise through practice, but are not valuable enough to teach. In military domains, it may be useful to teach improved strategies that arise from grossly extended practice, that is, tactics that are better, but that no person has had enough practice to learn them.

3.2 Expertise

Expertise, behavior where no more learning is needed, has an important role to play in models of human performance (Shadbolt & O'Hara, 1997). One of the Western powers' greatest strengths is training in depth *and* breadth. Practice influences speed of processing and error rates, particularly under stress. If synthetic forces are to be used to test doctrine, the effect of training on expertise must be included.

Expert behavior has been studied extensively in recent years, and a great deal is known about it (Chipman & Meyrowitz, 1993; Ericsson & Kintsch, 1995; Hoffman, Crandall, & Shadbolt, 1998). Some essential characteristics of expertise are that it includes highly developed perception for the domain material, selective search for solutions in that domain, and a good memory for domain-related material. In most domains, problem solving behavior (search) differs as well: novices tend to search backward from the situation to find solutions, and experts to search forward from the situation to find solutions (Larkin, McDermott, Simon, & Simon, 1980). Finally, transfer of expertise to other domains is limited.

Klein and his colleagues (e.g., Klein, 1997) have studied real-time performance in real settings (as opposed to laboratory settings) in detail, and have essentially found that the characteristics mentioned above are also critical in these situations. A number of rather extensive reviews have been undertaken of Klein's approach, which is often referred to as Naturalistic Decision Making (NDM) (see, for example, Hoffman & Shadbolt, 1995). A method to elicit this type of knowledge has been developed by Klein and his associates. It is known as the Critical Decision Method and is described in Hoffman et al. (1998). The specifically real-time challenges of acquiring knowledge relating to perceptually cue-rich decision making are discussed in a second DERA report by Hoffman and Shadbolt (1996).

Given the fact that it takes a long time to become an expert -- the rule of 10 years or 10,000 hours of practice and study is often mentioned by people such as Simon (Simon & Chase, 1973) -- the size of the data set has made it difficult indeed to study real-time learning on the road to expertise. Real-time learning in simpler problem solving tasks has been studied, however, and modeling accounts have been given (Anzai & Simon, 1979; John & Kieras, 1996; Nielsen & Kirsner, 1994; Ritter & Bibby, 2001). Some of these results may apply to expert learning in more complex tasks as well.

While experts vastly outperform non-experts in most domains, exceptions to this rule have been found, in domains such as clinical diagnosing, clinical prediction, personnel selection, and actuarial predictions (Dawes, 1988). In these domains, experts perform only slightly better than non-experts, and typically perform worse than simple statistical methods, such as regression analysis. One other aspect of behavior that distinguishes experts from novices is the ability to recover from errors. An important question is which category military diagnosing and prediction belong to because of the uncertainties involved, and, based on this result, what can be done (either by providing formal tools or by improving training) to remedy this situation and assist error recovery.

The effect of learning local environments and strategies (own and opponent's) must also be included. Having learnt the local terrain probably explains much of the home-field advantage. How does this learning occur?

Within the sub-field of knowledge-engineering there have been considerable efforts to produce methodologies for the acquisition, modeling and implementation of knowledge-intensive tasks. It is a moot point how far some of the resulting decision support systems are cognitively plausible. Nevertheless the methodologies do now represent powerful ways of constructing complex systems that exhibit task oriented behavior. To this end anyone engaged in engineering large scale synthetic environments should look at the principles laid down in the most recent of this work. The most accessible source is probably Schreiber et al. (in press).

3.3 Working memory

Central to all questions about human cognition and performance is the role of working memory. Working memory is implicated in almost all aspects of cognitive performance (Newell & Simon, 1972; Just & Carpenter, 1992; Boff & Lincoln, 1986, Section 7; Wickens, 1992). It is widely agreed that limitations of working memory are a major determinant of limitations of cognitive performance. Definitions of working memory are varied, but for present purposes we can take it to refer to the mechanisms that maintain and provide access to information created or retrieved during the performance of a task.

Modern approaches to the psychological study of human working memory often take as their starting point the famous paper by Miller (1956) and argue that people can retain only around "7 +/- 2" items in short-term memory. Later work has tended to revise that estimate downwards, towards 3 to 4 items of unrelated information (Crowder, 1976; Simon, 1974).

A more recent and influential line of work by Baddeley (1986; 1997) presents working memory as a dual system for the rehearsal of information, consisting of (a) a phonological loop, that contains around 2 seconds of verbalizations, for the rehearsal of phonological, acoustic, or articulatory information (e.g., useful for repeating a phone number until you dial it), and (b) a visuo-spatial scratchpad, of somewhat indeterminate capacity (e.g., useful when searching for an object that you have just seen), to play an analogous role for the maintenance of pictorial and spatial information.

Other approaches within experimental psychology place more emphasis on the role of working memory in both storing and manipulating temporary information (Daneman & Carpenter, 1980; Just & Carpenter, 1992). An important recent extension to the notion of working memory comes from the study of expertise, where Ericsson and Kintsch (1995) argue that after extensive practice in a particular domain people can, through specialized *retrieval structures,* use long-term memory for the rapid storage of temporary information (*long-term working memory*).

A recent book (Miyake & Shah, 1999) reviews a range of current approaches to the modeling of human working memory, although many of the models do not have the explicitness and generality needed to support the simulation of human performance in complex tasks. Of those that do, their view of working memory varies widely. Some, such as ACT-R (Anderson & Lebiere, 1998) and CAPS (Just & Carpenter, 1992), consider working memory not as a separate structural entity, but rather as an activated region of a larger, more general memory system, in which the limitations of working memory derive from a limited total quantity of activation. Just and Carpenter (1992, and more recently Lovett, Reder & Lebiere, 1999) extend that view to the modeling of individual differences, where different people are assumed to have different maximum quantities of available activation. A number of these ideas are put together by Byrne and Bovair (1997), who model (in CAPS) the way that a class of performance errors, in which people forget to complete subsidiary aspects of a task (such as removing the original from a photocopier), is affected by working memory load.

In contrast to these resource-limited models, Soar (Newell, 1990) imposes no structural limitation on working memory. Using Soar, Young and Lewis (1999) explore the possibilities of working memory being constrained not by physical resources, but by functional limitations and by specific kinds of similarity-based interference.

In summary, the current position is that human performance is known to be highly dependent on working memory and working memory load, and to be susceptible to factors such as individual differences (Just & Carpenter, 1992), distractions (Byrne & Bovair, 1997), emotion and stress (Boff & Lincoln, 1986), and expertise (Ericsson & Kintsch, 1995). Many existing models of human performance (e.g., as reviewed in Pew & Mavor, 1998) do not directly model the role of working memory. Models exist (Miyake & Shah, 1999), and some approaches to cognitive modeling (ACT-R, CAPS, Soar) have potential for improving predictions of human performance in realistic task situations by including more accurate theories of memory. There remains a need for the investigation and development of more explicit and complete models, with broader scope, of the role of working memory in human performance.

3.4 Emotions

Emotion, affect, and motivation are increasingly being seen as factors that can and often do influence cognition. This view has receiving attention amongst a range of computer scientists and psychologists. Pew and Mavor (1998, Ch. 9) lay out an initial case for including emotion as an internal moderator of behavior. The British HCI Group sponsored a one-day meeting on "Affective computing: The role of emotion in human computer interaction", which attracted 70 people to University College, London (Monk, Sasse, & Crerar, 1999). Picard's (1997) recent book provides a useful review of emotions and computation in general. Sloman's (1999) review of the book and Picard's (1999) response are useful summaries. A further case is also made here in the section on the Sim_Agent Toolkit.

We present here an additional argument for including a model of emotions in models of synthetic forces, note two potential problems with existing models, and sketch an initial theory.

3.6.1 Further uses of emotions

Models of emotions may be necessary for modeling non-doctrinal performance such as insubordination, fatigue, errors, and mistakes. Many authors have also noted the role of emotion in fast, reactive systems (Picard, 1997, provides a useful overview). Individual differences in emotions may be related to personality and differences in problem solving. That is, the range of emotions may be best explained as an interaction that arises between task performance and situation assessment and an agent's likes, desires, and personal cognitive style. An argument is starting to be put forward that changes in motivation based on temporally local measures of success and failure may help problem solving (Belavkin & Ritter, 2000; Belavkin, Ritter, & Elliman, 1999).

3.6.2 Working within a cognitive architecture

Emotions arise from structures related to cognition, and should be closely related to and based on cognitive structures. All of the arguments for creating a unified theory of cognition (Anderson, Matessa, & Lebiere, 1998; Newell, 1990) also apply to creating a unified theory of emotion as well. Emotions are presumably not task specific, so their implementation belongs in the architecture, not in the task knowledge.

Theories of emotions should thus be implemented within a cognitive architecture. This will allow them to realize all the advantages of being within a cognitive architecture, including being reusable and being compared to and incorporated within other models. Some models of emotions have been built within a cognitive architecture (Bartl & Dörner, 1998; Belavkin, Ritter, & Elliman, 1999; R. Jones, 1998; Rosenbloom, 1998). Being created within an information processing model has required them to be more specified than previous theories. Being part of a model that performs the task has also allowed them to make more predictions.

3.6.3 A sketch of a computational theory of emotions

An important aspect of cognition is to process sensory information, assign meaning to it, and then decide upon a plan of actions in response. This is a real-time process in which new sensory information arrives continuously. This view is similar to the view put forward by Agre and Chapman about representationless thinking. The plan must therefore be dynamically reconfigurable and will often be abandoned in favor of a better plan mid-way through its execution. Elliman has a speculative view of the role of emotions in cognition, similar to Rasmussen's (1998) stepladder framework of behavior, which makes the following assumptions:

1. The amount of sensory data available at any moment is too large for attention to be given to a more than a small fraction of the data.

2. The conscious consideration of the results of perception is an expensive process in terms of the load on neural hardware and is also time consuming.
3. Most sensory processing is unconscious in its early stages in order that expensive conscious processes need consider only the *results* of perception. These results might include labeled objects with a position in space, for example "a tank moving its turret in that clump of trees". Conscious processes might well add further detail such as the type of tank and the range of its gun.
4. Attentional mechanisms are needed to direct the limited high-level processing to the most *interesting* objects. These may be novel, brightly colored, fast moving, or potentially threatening.
5. Planning is an especially heavy computational process for the human mind, and one that is difficult to carry out effectively under combat conditions. (Perhaps the best way to explain why military doctrine is useful is that it distills the best generic practice and trains the soldier to behave in a way that might well have been a chosen and planned behavior if the individual had the time and skill to formulate the action himself. The danger is that no doctrine can envisage all scenarios in advance, and on occasion the use of doctrine in a rigid manner may be harmful.)
6. From an evolutionary perspective this system of unconscious processing of sensory input, attentional mechanisms, and cognitive planning (together with speech-based communication) is a masterstroke of competence for survival. However, it has one crippling disadvantage -- it is too slow to react to immediate and sudden attack.

Rapid reaction to possible threat without the time for much in the way of cognitive processing is clearly of huge value. In this framework emotion can be seen as kind of labeling process for sensory input. Fear particularly fits this pattern and is a label that causes selected sensory input to literally *scream for attention.* In order for this process to work rapidly it needs to be hard-wired differently than higher-level cognitive processes. There is strong evidence that the amygdala is intimately involved in the perception of threat, and is able to trigger the familiar sensation of fear (e.g., Whalen, 1999). If this organ of the brain is damaged, individuals may find everyday events terrifying whilst not perceiving any need for alarm in life threatening situations.

This rapid, emotive response to sensory data is inevitably relatively crude and prone to false alarms. Reactive behavior is triggered that may be involuntary, for example, the startle reaction and physiological changes due to the release of noradrenalin. After the reaction response it takes time for cognitive processes to catch up and make a more informed assessment of the situation and actual threat. If this emotive, reactive stimulation is excited in a chronic manner then susceptible individuals may become less effective, with impaired ability to think and plan clearly. Any kind of anxiety is a form of stress. Because individuals have a finite capacity for absorbing it, excessive stress results in fatigue.

3.5 Errors

Ideally, military behavior is functional and normative, that is, what should be done. Human behavior does not always match the normative ideal of military behaviors. One of the most important aspects of human performance, which has often been overlooked in models of behavior and problem solving, is errors (although see, for example, Cacciabue, Decortis, Drozdowicz, Masson, & Nordvik, 1992; Freed & Remington, 2000; Freed, Shafto, & Remington, 1998). There is a consensus building about the definition of errors -- for most people an error is something done that was not intended by the actor, that was not desired, and that placed the task/system beyond acceptable limits (e.g., Senders & Moray, 1991).

Part of the reason for omitting errors from models of behavior is the fallacy that they are produced by some special error-generating mechanism that can be bolted on to models once they are producing correct behavior on the task at hand. Often, however, the actions that precede errors would have been judged to be correct if the circumstances had been slightly different. In other words, as Mach (1905) observed, knowledge and error both stem from the same source.

Evidence shows that novices and experienced personnel will often make the same errors when exposed to the same circumstances. The difference lies in the ability to notice and recover from these errors. Experienced personnel are more successful at mitigating errors before the full consequences arise. In other words, it is the management of errors that is important and needs to be trained (Frese & Altmann, 1989), rather than vainly trying to teach people how to prevent the inevitable.

3.7.1 Training about errors

In any complex, dynamic environment, such as a military battlefield, the consequences of uncorrected errors are potentially disastrous. While normally a string of mistakes is required to create a disaster, the rapid pace of the battlefield and adversaries allow single mistakes to become more catastrophic.

There is, therefore, a real need to learn how to manage these errors in an environment in which the consequences are less severe. An advantage of using synthetic environments is that comparative novices can experiment in unfamiliar situations, with restrictions approximating the real environment in time, space, enemy capabilities, and so on, but with the knowledge that the consequences of any errors can be recovered. In addition, multiple scenarios can be played out over a compressed time period, thereby providing the novice with a variety of experiences that would take many years to accumulate through exposure to situations in the real world. This can be a great training aid, literally giving years of experience in far less time. When novices were trained in aircraft electrical system troubleshooting using a simulated system, they were able to acquire years of experience in months because the tutor let them practice their diagnostic skills without practice their disassembly skills (Lesgold, Lajoie, Bunzon, & Eggan, 1992).

3.7.2 Models that make errors

There are several process models that are complete enough to make errors, depending to some degree on the definition of error. Models that include errorful behavior exist in EPAM (Feigenbaum & Simon, 1984; Gobet & Simon, 2000), ACT-R (Anderson, Farrell, & Sauers, 1984; Anderson & Lebiere, 1998; Lebière, Anderson, & Reder, 1994) and Soar (Bass et al., 1995; Howes & Young, 1996; Miller & Laird, 1996), although each generates errors in quite different ways and at quite different levels. Fewer models exist that model error recovery, although this is clearly the next aspect to model.

A problem with models and humans is that the erroneous behavior is often task specific; given a new task, both models and humans might not generate the same behavior. In other words, the erroneous behavior arises as a result of the combination of human, technological, and organizational (environmental) factors. Vicente (1998) delineates some of the problems in this area.

There are various taxonomies of errors that could be incorporated into models of performance. There are also other constraints that reduce the level of performance that are worth exploring, including working memory (Young & Lewis, 1999), attention, and processing speed due to expertise.

3.6 Adversarial problem solving

Adversarial problem solving is different from simple problem solving and makes additional requirements for modeling behavior in synthetic environments. Planning is not done within a static environment, but done in an environment with active adversaries.

Research on adversarial problem solving (e.g., Chase & Simon, 1973; de Groot 1946/1978; Gobet & Simon, in press; Newell & Simon, 1972) has identified several aspects of cognitive behavior that have been shown to generalize to other domains, including the military domain (Charness, 1992). A key result is that players do not follow an strategy such as minimax, but that they satisfice (Simon, 1955), that is, they satisfy themselves with a good-enough solution, which can be far from the optimal solution (de Groot & Gobet, 1996; Gobet & Simon, 1996b). This satisficing behavior can be explained by the processing and capacity limits of human cognition, such as time to learn a new chunk or the capacity of short-term memory (Newell & Simon, 1972).

A second, related aspect is that a player's search is highly selective: only a few branches of the search tree are explored. The choice of subspace to search seems to be constrained by pattern-recognition mechanisms (Chase & Simon, 1973; Gobet, 1998; Gobet & Simon, 1996b). A consequence is that misleading perceptual cues may result in the exploration of an incorrect subspace. For example, Saariluoma (1990) reported that chess masters found a suboptimal solution when the features of the position led them to look for a standard, although inferior subspace. The consequence for understanding combatant behavior is that pattern recognition may influence the course of action chosen as much as the detail of the way the search is carried out. In fact, de Groot (1946/1978) did not find differences in the macrostructure of search of chess players at different skill levels.

A third important result is that chess players reinvestigate the same sequence of actions several times, interrupted or not by the analysis of other sets of actions. De Groot (1946) has called this phenomenon *progressive deepening*. It is related to the selective search shown by experts in other areas (Charness, 1991; Ericsson & Kintsch, 1995; Gobet & Simon, 1996b; Hoffman, 1992). De Groot and Gobet (1996) propose that progressive deepening is due both to the limits of human cognition (limited capacity of short-term memory, slow encoding time in long-term memory) and to the fact that with this searching behavior information gathered at various points of the search may be propagated to other points, including previously visited points (this could not be done with a search behavior such as minimax).

These features of cognition, identified in adversarial problem solving, also occur in rapid decision making, in domains such as fire fighting, combat, and chess players in time trouble. Interestingly, the model developed by Klein and his colleagues (see Klein, 1997 for a review) singles out the same features as the model developed by Chase and Simon (1973) to explain expert chess playing: pattern recognition, selective search, and satisficing behavior.

While some aspects of adversarial problem solving are well understood, others have yet to be studied in any depth. Such aspects include the way the function used to evaluate the goodness of a state (the evaluation function) changes as a function of time, the link between the evaluation function and pattern recognition, or the learning of domain-specific heuristics, which all have direct implications for combat behavior.

Relatively little research has been done on how players take advantage of the thinking particularities of their opponent, in particular by trying to outguess him. Jansen (1992) offers interesting results. He has developed a computer program that takes advantage of some features and heuristics of human cognition in simple chess endgames, such as the tendency, in human players' search, to avoid moves that lead to positions with a high branching factor, and to prefer moves that lead to forced replies. Using these features and incorporating them in its evaluation function, the program was able to win faster (in won positions) or to avoid defeat (in lost positions) more often against human players than by using a standard alpha-beta search. In principle, such an approach could be extended to include both skill-related and individual differences in synthetic environments.

In comparison to perception and memory in games, relatively little computer modeling of human behavior has been done with adversarial problem solving (if one excludes pure AI research, in which adversarial problem solving has been a favorite subject of research). One may mention the previous work of Simon and colleagues (Baylor & Simon, 1966; Newell, Shaw, & Simon, 1958), and the programs of Pitrat (1977), Wilkins (1980), and Gobet and Jansen (1994). All these programs were created for chess and most cover only a subset of the game.

There are implications of adversarial search variation for performance (i.e., how well a planner models an opponent). This would be a natural place to model various levels of experience in opponents.

3.7 Variance in behavior

Including more variety in how a model performs a task is one of the next steps for improving the realism of synthetic forces. Currently, many models will execute a task the same way every time and for every equivalent agent. In the real world, this is not the case. The choice of strategies and the ordering of substrategies will vary across agents and vary for a given agent across time. This lack of variance makes advisaries and allies too predictable in that they always do the same thing.

Including variance in behavior is also necessary when behavior is less predictable. Novices, with less knowledge, have greater variance in behavior (Rauterberg, 1993). In the past, variance was intentionally suppressed in simulations because it was thought that variance in real behavior was suppressed through doctrine and training. Including variety in behavior is of increasing importance when modeling less prepared and less trained forces, and now for improving model accuracy as variance in real behavior is admitted.

Variance in behavior is also important when modeling non-combatant agents, such as white forces and civilians. These agents may be producing their behaviors deterministically, but the determiners are often hidden from other agents, making them appear relatively unpredictable. Finally, the ability to model a variety of behaviors is necessary for sensitivity analysis.

Variance will arise out of several factors. It may arise from different levels of expertise, which is covered above. It may arise from different strategies, which will require including multiple strategies and noting where orders do not have to be followed and when panic leads orders to be ignored. Variance may also arise as a type of error, of applying a right action in the wrong circumstances.

In any case, variance in agent behavior in synthetic environments particularly needs to be included in training materials. Humans are very good pattern recognizers -- although they do not always look for or know the right pattern -- and will take advantage of models that do not vary their behavior. The real opponents will not usually be so predictable.

3.8 Information overload

Problems with information overload have been noted numerous times (e.g., Woods, Patterson, Roth, & Christoffersen, 1999). Problems resolving clutter, workload bottlenecks, and problems finding significance in incoming data, are not yet problems for many models of human performance. Currently, most cognitive and synthetic force models do not face information overload. The situation has more typically been of a model seeing only a limited set of information and knowing how to perform only one or a few tasks.

In the near future, the models will have more complex eyes with which to see as well as more knowledge to interpret the eyes' input. This will lead to more incoming information with a more difficult problem of deciding which objective to pursue next and how to choose the best strategy based on a larger set of knowledge. We will also find that models will start to have trouble with information overload, clutter, and situation assessment. Their tactics in this area will be particularly important when there are time pressures, which are common in synthetic environments and the worlds they model.

4. Current Objective: Better Integration

There are theoretical and practical problems integrating models with simulations and with other models. The problems can appear to be simply software issues, but deeper theoretical issues often go hand in hand with these problems. We thus note a few of these problems of getting models to interact with simulations as well the basic problem of aggregating models.

4.1 Perception

At least since de Groot's early work (1946), perception has been deemed to play an essential role in cognition. Neisser (1976, p. 9) aptly summarizes it as "perception is where cognition and reality meet". This point of view has been buttressed in recent years with the emphasis given by Nouvelle AI (e.g., Brooks, 1992), which is based on reactive architectures, perceptual mechanisms, and on their coupling with motor behavior. Neuroscience (e.g., Kosslyn & Koenig, 1992) teaches us that, due to evolutionary pressure, a large part of the brain deals with perception (mainly vision); hence, an understanding of perception is essential for understanding the behavior of combatants.

Perception-based behavior offers a series of advantages: it is fast, attuned to the environment, and optimized with respect to its coupling with motor behavior. However, its disadvantages include its tendencies to be stereotyped and to lack generalization. In addition, from the point of view of the modeller, it is a difficult behavior to simulate well. This is in part due to the fact that low-level perception is still poorly understood (Kosslyn & Koenig, 1992), although see recent progress in robotics and agents behavior for examples of successful implementation of basic perceptual mechanisms for use by cognition (e.g., Brooks, 1992, Zettlemoyer and St. Amant, 1999, and St. Amant and Riedl, in press).

Perception may be seen as the common ground where various aspects of cognition meet, including motor behavior, concept formation and categorization, problem solving, memory, and emotions. In several of these domains, computer simulations illustrating the role of perception have been developed. Brooks (1992) and others have investigated the role of perception in motor behavior with simple insect-like robots. The link between concept formation and (high-level) perception has been studied using the EPAM architecture (Gobet, Richman, Staszewski, & Simon, 1997). The role of perception in problem solving has been studied using CHREST, a variation of EPAM (Gobet, 1997; Gobet & Jansen, 1994) that also accounts for multiple memory regularities. Eye movements are simulated in detail in CHREST, but not the low-level aspect of perception. (We will deal with the relation between problem solving and perception in Section 3.5.) A more detailed simulation of low-level aspects of perception such as feature extraction is an important goal for the future of research on the relation of perception to other aspects of cognition. In addition, little work has been done on perception in dynamically changing environments, and on the effects of stress, emotion, motivation, and group factors on perception.

It is useful to separate *perception* from *cognition* in modeling human performance. The border between the model of the person and that of his environment can (arguably) be drawn at the boundary between cognition and perception, with perception belonging to a large extent in the environment model. This is true for psychological reasons (Pylyshyn, 1999). It is also true to support tying models to simulations and for use of the resulting knowledge by cognition in problem solving (Ritter et al., 2000). The typical acts performed by perception and motor action, what objects are in view, what are their shapes and sizes, and manipulating them, are most easily performed where the objects reside. This forces the implementation of theories of interaction into the simulation language instead of the modeling language.

It would be useful to have realistic stochastic distributions of differences in perception amongst individual agents, and also the ability to augment perception with instruments from field glasses to night sights. These devices could be modeled as *plug-ins* to the perception model. Models of perception in synthetic environments are typically simple, being a function of distance from observer to object (if there is a clear line of sight and the absence of cover and smoke). On the other hand, human perception changes in important ways with the ambient level of light, and with the part of the retina on which an image falls. The edges of the retina are particularly sensitive to the detection of a moving object, whilst the fovea has the best resolution for identifying distant objects and is most sensitive to color. The distance at which an object can be seen depends on its brightness, its size, and its contrast to the background as well as the permeability of the air to light. Thus a detonation will be visible from a much greater range than a moving tank, which in turn will be much easier to spot than a motionless camouflaged soldier.

Situation awareness is a term that is still the subject of much debate in the human factors and ergonomics communities (e.g., see the Special Issue of Human Factors, Volume 37, Issue 1). Situation awareness is a term that is still the subject of much debate in the human factors and ergonomics communities. Pew and Mavor (1998) consider situation awareness to be a key concept in the understanding of military behavior. We agree, but also believe that situation awareness should be modeled at a finer level of detail than is currently often done (see Pew & Mavor, 1998, Ch. 7, for a current review).

4.2 Combining perception and problem solving

Pew and Mavor (1998) note that an important constraint on problem solving is perception, but they do not explore this idea in detail. As mentioned in our discussion on expertise, perception plays an important role in skilled behavior: experts sometimes literally *see* the solution to a problem (De Groot, 1946/1978).

We may use Kosslyn and Koenig's (1992) definition: higher-level visual processing involves using previously stored information; lower-level visual processing does not involve such stored information, and is driven only by the information impinging on the retina. We focus here on higher-level perception and, thus, we will not consider mechanisms used for finding edges, computing depth, and so on.

Neisser's *Cognition and Reality* (1976) describes what is often referred to as the perceptual cycle. This approach underpins a vast amount of the cognitive engineering literature and research. At its simplest, the perceptual cycle is a cycle between the exploration of reality and representing this reality as schemas (in the general sense). Schemas direct exploration (perceptual, haptic, etc.) that involves sampling the object (looking at the real world), which may alter the object, which means that the schemas have to be modified. (See Neisser, 1976, p. 21 or p. 112 for the fuller picture.) This work suggests that an important aspect of behavior has been missing from many theories and models of problem solving that have not included perception.

It is natural that researchers have attempted in recent years to combine perception and problem solving in artificial systems. One can single out three main approaches: robotics, problem-solving architectures incorporating perception, and perceptual architectures being extended to problem solving.

In robotics, Nouvelle AI has attempted to build robots able to carry simple problem-solving behavior without explicit planning, but rather by linking sensor and motor abilities tightly (e.g., the behavior-based architecture of Brooks, 1992). Robots based on this approach are excellent at obstacle-avoiding behavior. It is, however, unclear how far this approach can be extended to more complex problem solving without incorporating some sort of planning.

Including perception in behavioral models is a useful way to add natural competencies and limitations to behavior. Pew and Mavor note that there are few models of how perception influences problem solving. Their summary can be extended and revised in this area, however. We have seen in existing cognitive models (Byrne, in press; de Groot & Gobet, 1996; Gobet, 1997; Jones, Ritter, & Wood, 2000; Ritter & Bibby, 2001; Ritter & Bibby, 1997; Ritter & Young, in press; Salvucci, in press) and in AI models (Elliman, 1989; Grimes, Picton, & Elliman, 1996; St. Amant & Riedl, in press) that perception is linked to and can provide competencies. While Pew and Mavor note that they are unaware of any attempt in Soar to model the detailed visual perceptual processes in instrument scanning (Pew & Mavor, 1998, p. 181), such models exist (Aasman, 1995; Aasman & Michon, 1992; Bass et al., 1995), and some are even cited by Pew and Mavor (1998, p. xx) for other reasons.

The Soar model reported by Bass et al. (1995), scans a simple air traffic control display to find wind velocity. The model learns (chunks) this information and uses it and the display to track and land a plane through airport air traffic control. The model then reflects on what it did to consider a better course of action. This model shows tentative steps towards using Soar's learning mechanism for situation learning and assessment based on information acquired through active perception (see Pew & Mavor, 1998, p. 197). Work in this area, modeling visual cognition within Soar, continues at ISI (Hill, 1999) and at Penn State.

The EPAM architecture (Feigenbaum & Simon, 1984), the initial goal of which was to model memory and perception, has recently been extended into a running production system (Gobet & Jansen, 1994; Lane, Cheng, & Gobet, 1999). The chunks learned while interacting with the task environment can later be used as conditions of productions. The same chunks are also used for the creation of schemas and for directing eye movements.

Recently, there have been several attempts to move the perception component from models into the architectures, regularizing and generalizing the results in the process. Prominent cognitive architectures Soar and ACT-R have been extended to incorporate perceptual modules. With Soar, a perceptual module is available based on EPIC (Chong & Laird, 1997) and another based loosely on a spotlight theory of attention (Ritter et al., 2000). With ACT-R, two perceptual modules have been developed independently: the Nottingham architecture (Ritter et al., 2000) and ACT-R/PM (based on but also extending EPIC) (Byrne, in press; Byrne, 1997). This approach creates situated models of cognition, that is, models that interact with (simulations of) the real world.

None of these approaches has been tested with complex, natural, and dynamically changing environments. The robotics approach is the only one currently demonstrated to cope with natural, albeit rather simple environments. The two other approaches interact with computer interfaces, which are complex and dynamic (e.g., Salvucci, in press).

4.3 Integration of psychology theories

A glance at almost any psychology textbook reveals that the study of human cognition is conventionally divided into topics that are presented as if they have little to do with each other. There will be separate chapters on attention, memory, problem solving, and so on. However, the range and variety of tasks undertaken by people at work, and also those tackled by synthetic agents, typically require the application and interplay of many different aspects of cognition simultaneously or in close succession. Interacting with a piece of electronic equipment, for example, can draw upon an agent's capacity for perception, for memory, for learning, for problem solving, for motor control, for decision making, and many more. The question of how to integrate these different facets of cognition is therefore an important one for the simulation of human behavior.

Integrating theories across different topics of cognition is an issue that has rarely been addressed directly, and provides an important focus for future work. Agents in synthetic environments (e.g., R. Jones, Laird, Nielsen, Coulter, Kenny, & Koss, 1999) implicitly integrate multiple aspects of behavior. What research exists has been carried out, appropriately enough, under the heading of unified theories of cognition using architectures such as Soar and ACT-R. Soar in particular offers a promising basis for such integration. Its impasse-driven organization enables it to access different areas of cognitive skill as the need arises, and its learning mechanism (which depends on cognitive processing in those impasses) enables relevant information from the different areas to be integrated into directly applicable knowledge for future use.

4.4 Integration and reusability of models

Integration of theories can be also viewed as integration of models as software, sometimes called reuse. It has been true for years that reuse is important; this is true for two fundamental reasons. First, reuse saves effort. In the field of Object Oriented Software Development figures are often quoted for the costs associated with developing with reuse in mind. The extra time spent in initial development is something like 20%. When the code is reused, an application can be created in 40% of the development time for new code. Second, and perhaps more importantly in these domains, reuse ensures consistency across simulations and across time.

There are also serious problems restricting the reuse of cognitive models. Cognitive models are not generally reused, even when they have been created in a cognitive architecture designed to facilitate their reuse. There are exceptions. Pearson's Version 2 of his Symbolic Concept Acquisition model and its explanatory displays is an exception that helped inspire this work (available at ai.eecs.umich.edu/soar/soar-group.html). Other exceptions include PDP toolkits such as O'Reilly's PDP++ (http://www.cs.cmu.edu/Web/Groups/CNBC/PDP++/PDP++.html). But overall cognitive modeling does not have the level of system reuse and visual displays that the AI and expert systems communities now take for granted.

There are some examples of reuse that should be emulated and expanded. ACT-R now maintains a library of existing models (<act.psy.cmu.edu>). We have found that the mere existence of a library of student models (www.nottingham.ac.uk/pub/soar/nottingham/) has led to increasingly better student projects. Work by Young (1999) on building a zoo of runnable cognitive models is another example of such use done broadly. There is little reason to believe that these results would not scale up. These improvements to the modeling environment have moved learning Soar (Ritter & Young, 1999) and ACT-R (Anderson & Lebiere, 1998) from being a lengthy apprenticeship to being something that can be taught in courses.

Such integration is illustrated most clearly in a model of natural language sentence processing (Lewis, 1993), in which lexical, syntactic, semantic, pragmatic, and domain-specific knowledge are brought together in learned rules to guide language comprehension. Probably the model that has gone furthest in demonstrating this kind of integration is the cognitive model of the NASA Test Director, the person responsible for co-ordinating the preparation and launch of the space shuttle. Nelson, Lehman, and John (1994) describe a Soar model of a fragment of the Test Director's performance, which incorporates problem solving, listening to audio communications, understanding language, speaking, visual scanning (through a procedure manual), page turning, and more. Such models are also being created in ACT-R (Anderson & Lebiere, 1998).

Integration of a slightly different flavor -- across capabilities rather than across textbook-like topics of cognition -- is illustrated in another Soar model, this one being of exploratory learning of an interactive device (Rieman et al., 1996). At first glance, it might seem that exploratory learning is not especially relevant to the human behavior that is, apart from questions of training, the main focus of this report. Fighter pilots and tank commanders are highly trained and expert individuals, and presumably do not learn significantly from single experiences. However, component skills such as comprehending a novel situation, looking around to discover relevant options, and assessing a course of action -- which are fundamental components of expert skill -- are also precisely what are required for exploratory learning and reactive planning in uncertain environments.

Hoffman and Shadbolt (1996) provide a review of work on information overload in real-time, high work-load military contexts. They also discuss challenges that information overload raises for knowledge acquisition in the context of synthetic forces environments.

Rieman et al. (1996) describe the IDXL model, which models an experienced computer user employing exploratory learning to discover how to perform specified tasks with an unfamiliar software application. IDXL searches both the external space provided by the software and the internal space of potentially relevant knowledge. It seeks to comprehend what it finds, and approximates the *rationally optimal* strategy (Anderson, 1990) for exploratory search. A typical sequence of interrelated capabilities would be for the model first to learn how to start a spreadsheet program from external instruction; then to use that new knowledge as a basis for analogy in order to discover how to start a graph-drawing package; and then to build on its knowledge by learning through exploration how to draw a graph. The model works with a limited working memory, employs recognition-based problem solving (Howes, 1993), and acquires display-based skill (Payne, 1991) in an interactive, situated task.

These problems of reusability are even more acute when creating models for synthetic environments because of the size and type of models. This is true for several reasons: The knowledge is more extensive and exact than many laboratory domains previously studied. The models must interact with complex, interactive simulations. The work may be classified, which will add an additional constraint in hiring someone with multiple skills. Scenarios may simulate hours of behavior rather than the minutes of typically modeled tasks. This represents a lot of knowledge, and can make troubleshooting more difficult. Finally, there are many cases where an explanation facility is required to explain the model's behavior for other observers.

4. 5 Summary

A framework to assist with integration and reuse will have to be developed. It would be common in the sense that the appropriate simulation entities and analysis tools would be available, and for a given application or analysis developers would plug them together. The DIS protocol and ModSAF are being used in this way to some extent, but there are not many tools, they are hard to use, and they do not support the desired level of ease of use nor the level of cognitive realism.

5. Current Objective: Improved usability

In addition to improving the match of synthetic forces to human behavior itself, there are several aspects of these models that must be improved so they can be developed, tested, and used by modelers and analysts. A large amount of time is often required to build models and understand their behavior, more than we believe it should be. The difficulties of simply creating and manipulating models of behavior can preclude us from spending more time on using these models in training or for performing what-if analyses.

While Pew and Mavor (1998, p. 10) initially note that they will not address usability, they later (p. 282) note the need to have quickly reconfigurable models. They also discuss (p. 292) ease of use. This inconsistency is completely appropriate because usability is important. Models that are too difficult to be used are not used. This issue is also being taken up in the next generation of simulation models in the US (Ceranowicz, 1998). There are several aspects to usability that we will take up next.

5.1 Usability of the behavioral models

As we have noted before (Ritter, Jones, & Baxter, 1998b; Ritter & Larkin, 1994), cognitive models suffer from usability problems. Few lessons from the field of human-computer interaction (HCI) have been reapplied to increase the understanding of the models themselves, even though many results and techniques in HCI have been discovered using cognitive modeling.

Modelers have to interact with the model several times and in several ways over the lifetime of the model. As a first step, the models must be easy to create. As part of the creation and validation process, the models must be debugged on the syntactic level (will it run?), on the knowledge level (does it perform the task?), and on a behavioral level (does it perform the task like a human?). All of these levels are important if the costs of acquiring behaviors are to be reduced. While we can point to some recent advances in usability (e.g., Anderson & Lebiere, 1998; Kalus & Hirst, 1999; Ritter et al., 1998b), further work will be required. It is also probably fair to say that cognitive models can often be difficult to explain and understand.

5.2 Desired accuracy of the behavior models

Another problem is knowing when to stop improving the model. In science for science's sake, there is no limit -- the model is continually improved. In the case of engineering-like applications, such as behavioral models in synthetic environments, knowing when to stop is a valid question. In many cases we do not know how accurate these models have to be in order to be useful, and at what point additional accuracy is no longer worthwhile.

The purpose and goals of each modeling project will help determine when to stop development, so they need to be carefully laid out when developing a model of behavior. The stopping rule also applies to the synthetic environment as well as the model -- there is no point in developing a simulation that is too detailed. This question is becoming more important as the models become more accurate and more modifiable.

5.3 Aggregation and disaggregation of behaviors

A clear requirement for simulations in synthetic environments is the ability to aggregate or summarize subunits and in other situations the ability to disaggregate and place the subunits from a larger grouping. When the tanks in a platoon are each simulated in a platform-level simulation, they must be aggregated to display them as a platoon on a more abstract or larger-scale map. Similarly, higher-level units may have to be placed into a simulation when moving a larger unit into a platform-level simulation. This aggregation (or disaggregation) may need to occur multiple times when crossing levels of resolution in order to provide the right level for a report.

This area has received a limited amount of study, yet it is a common need across multiple types of simulations. None of the cognitive architectures examined in Pew and Mavor (1998, Table 3.1) or here offer any insight. The only related actions are that several of the architectures (e.g., COGNET, Soar) are designed to support multiple agents.

5.4 Summary

Environments for interacting with existing modeling architectures are generally poorer than those now provided for most programming languages. The requirements for modeling are greater than general programming, including the need for adjustable accuracy, different levels of analyses, and multiple measurements from running programs. These factors contribute to making modeling difficult. We need new models, and new techniques for building and using models.

6. Recent developments for modeling behavior

In addition to the architectures and approaches identified by Pew and Mavor (1998), there are a few other architectures that are worth examining. In this section we note them, including the lessons they provide. Our reviews also explicitly consider ease of use (i.e., model populating).

We focus our comments on cognitive architectures because they have been created for modeling the strengths and limitations of human behavior. Any system built for other reasons that was adapted in this way -- for example, other AI systems -- would start to approach these systems in capabilities and limitations. It is quite likely that the cognitive architecture that best matches human behavior will vary by the type of behavior and level of aggregation. For example, different architectures will be preferred for modeling a soldier performing simple physical tasks and for modeling a deliberate and reflective commander.

There will continue to be a range of architectures created. We agree completely with Pew and Mavor that further work is necessary before settling on an architecture. That is not to say that architectures will not continue to converge. We start, however, by examining ways to summarize data, and some advanced AI techniques to help create models. We then examine several architectures.

6.1 Data gathering and analysis techniques

Scattered throughout Pew and Mavor (e.g., pp. 323-325) are comments about the need for data to develop and test models. This requirement should be highlighted because there are different kinds of data, and each kind is important. Also, there is a need for analysis techniques because data alone are meaningless.

Data to develop models can come from a wide variety of sources. Data can come from speaking to experts and having them do tasks off-line, so-called knowledge acquisition (Shadbolt & Burton, 1995). Data can also come from having experts talk aloud while performing the task (Ericsson & Simon, 1993). Talking aloud is a more accurate way to acquire the knowledge because it is based on actual behavior rather then someone's impression and memory of behavior. It is, however, a more costly approach because the modeller must infer the behavior generators. Data for developing models can also come from non-verbal measurements of experts while they perform the task. Non-verbal measurements are probably the least useful data (but still useful in some circumstances) for developing models. These data are useful, however, in testing models that make timing predictions. Data can also come from previously run studies, reviews, and compendia of such studies (e.g., Boff & Lincoln, 1986). A useful review of data types and analysis methods in this area is provided by Hoffman (1987).

A major requirement will be a balance between the experimental control of the lab and the richness of the real world. This can be achieved by gathering data in the same micro-world simulations in which the models will be deployed, such as synthetic environments. These environments can be used to model all the salient aspects of the real world, whilst still providing some level of experimental control.

Once the data are in hand, they will often have to be aggregated or summarized. Expert summaries from knowledge acquisition already represent summarized data, but protocol analysis has developed a wide range of techniques for summarizing such data.

Reviews and suggestions in this area are available (Ericsson & Simon, 1993; Sanderson & Fisher, 1994), but there exists a very wide range of techniques that vary based on how advanced the theory is, the purposes of the research, and the domain. One example of an advanced technique to examine protocol data for temporal patterns for later inclusion and comparison against model behavior (Kuk, Arnold, & Ritter, 1999).

With data in hand, the next step is either to develop a model or to test an existing model. There is little formal methodology about how to create models. Some textbooks attempt to teach this creative task either directly (vanSomeren, Barnard, & Sandberg, 1994) or by example (McClelland & Rumelhart, 1988; Newell & Simon, 1972). There are summaries of the testing process (Ritter & Larkin, 1994) and of some possible tests (Ritter, 1993a), but repairing a model based on the results of the tests can be a task requiring a lot of creativity.

6.2 Advanced AI approaches

There are some existing tools that could be used to create, augment, or optimize models of performance. We note here three that we are particularly familiar with. These include approaches for creating behaviors, such as genetic algorithms (Burke, Elliman, & Weare, 1995) and traditional AI programs (Bai, Burke, Elliman, & Ford, 1993).

6.2.1 Genetic algorithms

Genetic algorithms (GAs) are search methods that can be used in domains in which no heuristic knowledge is available and in which an objective function exhibits high levels of incoherence. That is to say, a small change to the solution state may often result in large changes to the objective function or fitness measure. These algorithms are expensive in machine resources and exhibit slow (but often steady) convergence to a solution. They might be used as a search strategy of last resort for plan formation.

Heuristics can be used with GAs to seed the initial population in a non-random way or to guide the crossover process in a way that changes the distribution of offspring. Doing this results in a *memetic* algorithm (one that manipulates basic blocks of information or *memes*). As has been common experience throughout the history of AI, this introduction of domain knowledge can drastically transform the performance of the GA. Such algorithms have been found to exceed the performance of previous approaches in a number of domains (Burke et al., 1995). There may be scope for using this as a search strategy in planning.

6.2.2 Tabu search

Tabu search, as developed by Glover (Glover & Laguna, 1998), is a general purpose approach that is remarkably effective for difficult problems where the objective function has some local coherence. It is surprising how often hill-climbing approaches such as the A* algorithm are used in current plan building algorithms, despite the domains being prone to local maxima. Tabu search uses the novel concept of *recency* memory to prevent moves in a solution space from being tried when some component of that state has recently been changed in a previous move. This surprisingly simple idea forces the search away from a local maximum. Long-term memory is used to hold the best solution state found so far, and this may be used to re-start the search far away from any previous exploration of the state space.

The tabu search approach would almost certainly lead to improved solutions with reasonable computational complexity. It would be worth using this approach to search for strategies and plans at various levels in a synthetic environment from the individual combatant to the highest level source of command and control.

Soar is impressive in its ability to reuse parts of problems that have been solved in the past, and to plan in a goal-directed way that can seem ingenious. Real human problem solving can be less structured, however, and can leap from one approach to another in a manner that is difficult to model. Tabu search has this characteristic, however, as part of its diversification strategy. Including tabu search in a cognitive architecture would be interesting. There may be some advantages to be gained by grafting on other similar systems that modify the beliefs of a cognitive architecture so as to maintain various types of logical consistency in the set of facts held.

6.2.3 Sparse Distributed Memory

Subtle issues such as the *tip-of-the-tongue* phenomena (Koriat & Lieblich, 1974) and the fact that we know if we know something (feeling of knowing) before becoming aware of the answer are not often modeled (although, see Schunn, Reder, Nhouyvanisvong, Richards, & Stroffolino, 1997 for a counter example). These effects may be captured using memory models such as Kanerva's (1988) sparse distributed memory (SDM), and Albus's (1971) CMACCerebellar Model Arithmetic Computer). The way in which a combatant's experience of the world is stored and modeled is important. An SDM seems to offer powerful human-like ways of recalling nearest matches to present experience in a best-first manner. This can even be seen as a kind of thinking by analogy that has a uniquely human-like ability to find a close match rapidly without exhaustive or even significant time spent in search.

6.3 Multiple criteria heuristic search[1]

Heuristic search, one of the classic techniques in AI, has been applied to a wide range of problem-solving tasks including puzzles, two-player games, and path finding problems. A key assumption of all problem-solving approaches based on utility theory, including heuristic search, is that we can assign a single utility or cost to each state. This in turn requires that all criteria of interest can be reduced to a common ratio scale.

The route planning problem has conventionally been formulated as one of finding a minimum-cost (or low-cost) route between two locations in a digitized map, where the cost of a route is an indication of its quality (e.g., Campbell, Hull, Root, & Jackson, 1995). In this approach, planning is regarded as a search problem in a space of partial plans, allowing many of the classic search algorithms such as A* (Hart, Nilsson, & Raphael, 1968) or variants such as A*epsilon (Pearl, 1982) to be applied. However, while such planners are complete and optimal (or optimal to some

[1] This section was drafted by Brian Logan, and was revised by the authors.

bound ε), formulating the route planning task in terms of minimizing a single criterion is difficult.

For example, consider the problem of planning a route in a complex terrain consisting of hills, valleys, impassable areas, and so on. A number of factors will be important in evaluating the quality of a plan: the length of the route; the maximum negotiable gradient; the degree of visibility; and so on. In any particular problem, some of these criteria will affect the feasibility of the route, while others are simply preferences. Route planning is an example of a wide class of multi-criteria problem-solving tasks, where different criteria must be traded off to obtain an acceptable solution.

One way of incorporating multiple criteria into the problem-solving process is to define a cost function for each criterion and use, for example, a weighted sum of these functions as the function to be minimized. We can, for example, define a *visibility cost* for being exposed and combine this with cost functions for the time and energy required to execute the plan, to form a composite function that can be used to evaluate alternative plans. However, the relationship between the weights and the solutions produced is complex in reality, and it is often unclear how the different cost functions should be combined linearly as a weighted sum to give the desired behavior across all magnitude ranges for the costs. This makes it hard to specify what kinds of solutions a problem-solver should produce and hard to predict what a problem solver will do in any given situation; small changes in the weight of one criterion can result in large changes in the resulting solutions. Changing the cost function on a single criterion to improve the behavior related to that criterion often leads to changing all the weights for all the other costs as well because the costs are not independent. Moreover, if different criteria are more or less important in different situations, we need to find sets of weights for each situation.

The desirability of trade-offs between criteria is context dependent. In general, the properties that determine the quality of a solution are incommensurable. For example, the criteria may only be ordered on an ordinal scale, with those criteria that determine the feasibility of a solution being preferred to those properties that are merely desirable. It is difficult to see how to convert such problems into a multi-criterion optimization problem without making ad hoc assumptions. It is also far from clear that human behavior solely optimizes on a single criterion.

Rather than attempt to design a weighted sum cost function, it is often more natural to formulate such problems in terms of a set of constraints that a solution should satisfy. We allow constraints to be prioritized, that is, it is more important to satisfy some constraints than others, and soft, that is, constraints are not absolute and can be satisfied to a greater or lesser degree. Such a framework is more general in admitting both optimization problems (e.g., minimization constraints) and satisficing problems (e.g., upper bound constraints), which cannot be modeled by simply minimizing weighted-sum cost functions. Vicente (1998) suggests ways in which such constraints can be analyzed as part of a work domain analysis.

This approach provides a way for more clearly specifying problem-solving tasks and more precisely evaluating the resulting solutions. There is a straightforward correspondence between the *real problem* and the constraints passed to the problem-solver. A solution can be characterized as satisfying some constraints (to a greater or lesser degree) and only partially satisfying or not satisfying others. By annotating solutions with the constraints they satisfy, the implications of adopting or executing the current best solution are immediately apparent. The annotations also facilitate the integration of the problem-solver into the architecture of an agent or a decision support system (see for example, Logan & Sloman, 1998). If a satisfactory solution cannot be found, the degree to which the various constraints are satisfied or violated by the best solution found so far can be used to decide whether to change the order of the constraints, relax one or more constraints or even to redefine the goal, before making another attempt to solve the problem.

The ordering of constraints blurs the conventional distinction between absolute constraints and preference constraints. All constraints are preferences that the problem-solver will try to satisfy, trading off slack on a more important constraint to satisfy another, less important constraint.

The A* search algorithm is ill suited to dealing with problems formulated in terms of constraints. Researchers at Birmingham have therefore developed a generalization of A* called A* with bounded costs (ABC) (Alechina & Logan, 1998; Logan & Alechina, 1998), which searches for a solution that best satisfies a set of prioritized soft constraints.

The utility of this approach and the feasibility of the ABC algorithm have been illustrated by an implemented route planner that is capable of planning routes in complex terrain satisfying a variety of constraints. This work was originally motivated by difficulties in applying classical search techniques to agent route planning problems. However, the problems they identified with utility based approaches, and the solutions they propose, are equally applicable to other search problems.

6.4 Psychologically inspired architectures

We review here several psychologically inspired architectures that were not covered by Pew and Mavor (1998). These architectures are interesting because (a) they are psychologically plausible, (b) some of them provide examples of how emotions and behavioral moderators can be included, and (c) several illustrate that better interfaces for creating cognitive models are possible.

6.4.1 EPAM

EPAM (Elementary Perceiver and Memoriser) is a well-known computer model of a wide and growing range of memory tasks. The basic ideas behind EPAM include mechanisms for encoding chunks of information into long-term memory by constructing a discrimination network. The EPAM model has been used to simulate a variety of psychological regularities, including the learning of verbal material (Feigenbaum & Simon, 1962; Feigenbaum & Simon, 1984) and expert digit-span memory (Richman, Staszewski, & Simon, 1995). EPAM has been expanded to use visuo-spatial information, as in MAPP (Simon & Gilmartin, 1973).

EPAM organizes memory into a collection of chunks, where each chunk is a meaningful group of basic elements. For example, in chess, the basic elements are the pieces and their locations; the chunks are collections of pieces, such as a king-side pawn formation. These chunks are developed through the processes of discrimination and familiarization. Essentially, each node of the network holds a chunk of information about an object in the world. The nodes are interconnected by links into a network, with each link representing the result of applying a test to the object. When trying to recognize an object, the tests are applied beginning from the root node, and the links are followed until no further test can be applied. At the node reached, if the stored chunk matches that of the object then familiarization occurs, in which the chunk's resolution is increased by adding more details of the features in that object. If the current object and the chunk at the node reached differ in some feature, then discrimination occurs, which adds a new node and a new link based on the mis-matched feature. Therefore, with discrimination, new nodes are added to the discrimination network; with familiarization, the resolution of chunks at those nodes is increased.

The EPAM template theory, and its computer implementation CHREST (Gobet & Simon, 1996a), uses the same framework as the EPAM theory (Feigenbaum & Simon, 1984; Richman et al., 1995), with which it is compatible. As in the earlier chunking theory of Chase and Simon (1973), the EPAM template theory assumes that chess experts develop a large EPAM-like net of chunks during their practice and study of the game. In addition, EPAM assumes that some chunks, which recur often during learning, develop into more complex retrieval structures (templates) having slots for variables that allow a rapid encoding of chunks or pieces.

CHREST (de Groot & Gobet, 1996; Gobet & Simon, 1996a) is one of most current theories of memory developed from the ideas in EPAM. Gobet and Simon (2000) present a detailed description of the present version of CHREST and report simulations on the role of presentation time in the recall of game and random chess positions.

EPAM and its implementations are important to consider because they fit a subset of regularities in memory very well. This at least serves as an example for other theories and architectures to emulate. It may also be possible to include the essentials of EPAM in another system, such as Soar or ACT-R, extending the scope of both approaches.

6.4.2 Neural networks

Pew and Mavor (1998, Ch. 3) review neural networks. Here, therefore, we only provide some further commentary, introduce some more advanced concepts, and note a few further applications.

SDMs are Sparse Distributed Memories, which are a plausible model of brain architecture, particularly the cerebellum. They have the interesting property of storing memories such that recall works by finding the best match to imperfect data. They are also a natural way of storing sequences. They exploit interesting mathematical properties of binary metric spaces with a large number of dimensions. It is intriguing that SDMs have the human-like properties that they "know if they know" something before the retrieval process is complete. They also exhibit the tip-of-the-tongue phenomenon, and replicate the human ability to recall a sequence or tune given the first few items or notes. They can also learn actuator sequences that might be used in muscle control or reflex patterns of behavior.

Connectionist systems have demonstrated the ability to learn arbitrary mappings. Architectures such as the multi-layer perceptron (MLP) are capable of being used as a black box that can learn to recognize a pattern of inputs as a particular situation. This requires supervised training, and may involve heavy computational resources to arrive at a successful solution using the back-propagation algorithm. Training can be continued during performance as a background task, and thus an entity could have an ability to learn during action based on this approach. Recognition performance is relatively rapid and thus a multilayer perceptron might be used to model a reaction mechanism in which a combatant responds to coming under fire, or spotting the presence of the enemy, for example. It might also be used to activate particular aspect of military doctrine depending on the current circumstances.

More specialized connectionist systems have been used for modeling low-level vision, the learning of grammar rules, the pronunciation of words, character recognition, and so on. These models do not seem to have immediate application in military simulation.

Recurrent nets such as the Elman (1991) net have the ability to generate sequences of tokens as output. These seem to offer some promise of detecting an input situation and producing a series of behavioral actions as a response. This may be useful for modeling the reactive behavior of an entity over a short time period, whilst a symbolic cognitive model is used for the higher-level cognitive processes that occur over a longer time span.

6.4.3 PSI and architectures that include emotions

PSI is a relatively new cognitive architecture designed to integrate cognitive processes, emotion, and motivation (Bartl & Dörner, 1998). The architecture includes six motives (needs for energy, water, pain avoidance, affiliation, certainty, and competence). Cognition is modulated by these motive/emotional states and their processes. In general, PSI organizes its activities similar to Rasmussen's (1983) hierarchy: first, it tries highly automatic skills if possible, then it skips to *knowledge-based* behavior, and as its *ultima ratio* approach it uses trial-and-error procedures. It is one of the only cognitive architectures that we know about that takes modeling emotion and motivation as one of its core tasks.

The PSI architecture is currently incomplete, which raises interesting questions about how to judge a nascent architecture. PSI does not have a large enough user community and has not been developed long enough to have a body of regularities is has been compared with let alone adjusted to fit. How can PSI be compared with the older architectures with existing tutorials, user manuals, libraries of models and example applications?

A model in the PSI architecture has been tested against a set of data taken from a dynamic control task. The model's number of control actions was within the range of human behavior and its predictions of summary scores were outside the range of human behavior -- the model was less competent (Detje, 2000). This model needs to be improved before it matches human emotional data as well as other cognitive models match non-emotional data. It is, however, one of the few models of emotion compared with data.

Several other models of emotions and architectures that use emotions have been created. Reviews of emotional models (Hudlicka & Fellous, 1996; Picard, 1997) typically present models and architectures that have not been compared and validated against human data. There appears to be one other exception, an unpublished PhD thesis by Araujo at Sussex (cited in Picard, 1997). Some of use are attempting to add several simple emotions to ACT-R (Belavkin et al., 1999) and validate the model by comparing the revised model with an existing model and comparable data (G. Jones, Ritter, & Wood, 2000).

6.4.4 Cogent

Cogent is a design environment for creating cognitive models and architectures (Cooper & Fox, 1998). It allows the user to draw box and arrow diagrams to structure and illustrate the high-level organization of the model, and to fill in the details of each box using one or a series of dialogue sheets. The boxes include inputs, outputs, memory buffers, processing steps, and even production systems as components.

Cogent's strengths are that it is easy to teach; the displays provide useful summaries of the model that helps with explanation and development; and the environment is fairly complete. It appears possible to reuse components on the level of boxes. Cogent's weaknesses are that it is fairly unconstrained; for large systems it may be unwieldy; and it might not interface well to external simulations.

Cogent also shows that cognitive modeling environments can at least appear more friendly. The results of its graphic interface routinely appear in talks as model summaries. The interface is also quite encouraging, allowing users to feel that they can start working immediately.

6.4.5 Hybrid architectures

Hybrid architectures are architectures that typically include symbolic and non-symbolic elements. A more general definition would be architectures that include major components from multiple architectures.

Hybrid architectures are mentioned briefly by Pew and Mavor (1998, pp. 108-110). Work has continued in this area past the amount they were able to review, with some interesting results. LICAI (Kitajima & Polson, 1996; Kitajima, Soto, & Polson, 1998), for example, models how people explore and use interfaces based on a theory of how Kintsch's (1999) schemas receive activation. The US Office of Navy Research (ONR) has sponsored a research program on hybrid architectures (Gigley & Chipman, 1999). This has given rise to some interesting hybrid architectures (e.g., Sun, Merrill, & Peterson, 1998; Wang, Johnson, & Zhang, 1998).

Perhaps the most promising hybrids are melding perception components across cognitive architectures. The Epic (Kieras & Meyer, 1997) architecture's perception and action component has been merged with ACT-R (Byrne & Anderson, 1998). This has lead to direct reuse and unification. Similar results have been found with the Nottingham functional interaction architecture being used by Soar and ACT-R models (Bass et al., 1995; Baxter & Ritter, 1996; Ritter et al. 2000; G. Jones et al., 2000).

6.5 Knowledge-based systems and agent architectures

The field of agency has enjoyed a recent surge of interest as a potential solution to some of the problems posed by global information networks. Most principled agent architectures have historical roots in distributed artificial intelligence (DAI). For several decades, DAI has been tackling essentially the same problem as knowledge-based systems (KBS) research, namely how to produce efficient problem-solving behavior in software. The main concept that brings agency and KBS together is the idea of operation at the knowledge level as described by Newell (1982).

The behavioral law used by an observer to understand the agent at the knowledge level is the principle of maximum rationality (Newell, 1982), which states: "If an agent has knowledge that one of its actions will lead to one of its goals, then the agent will select that action". The modeling of intelligent artificial systems at the knowledge level, that is, with no reference to details of implementation, is a key principle in KBS construction. It is also at the heart of many assumptions in the tradition of explaining human behavior.

Nwana (1996) claims that an important difference between agent-based applications and other distributed computing applications is that agent-based applications operate typically at the knowledge level, whereas distributed computing applications operate at the symbol level. At the symbol level, the entity is seen simply as a mechanism acting over symbols, and its behavior is described in these terms.

The theoretical links between the motivations behind KBS and agent research can be seen in the main approaches taken to the definition of software agency. Ascriptional agency attempts to create convincing human-like behaviors in software in the belief that this will produce programs that are easy to interact with. This can be seen as paralleling the expert behavioral modeling approach that is currently widely espoused in the KBS community. BDI (Belief-Desire-Intention) agents focus on the concept of intentionality -- the mental attitudes of the agent. BDI models have been successfully implemented in systems such as the DESIRE framework (Brazier, Dunin-Keplicz, Treur, & Verbrugge, 1999) and the JACK component system (Busetta, Howden, Rönnquist, & Hodgson, 1999a; Busetta, Rönnquist, Hodgson, & Lucas, 1999b) .

Jack is an extension to JAVA. It includes a JAVA library and a compiler that takes a JAVA program with embedded Jack statements. A JAVA compiler expands/incorporates the Jack statements to create a runnable JAVA program. These statements implement a BDI architecture, while allowing JAVA statements to extend and implement them. The statements include commands like @achieve(*condition, event*), which subgoal on *event* if *condition* is not found to be true.

The resulting program instantiates a BDI agent. Its BDI architecture is made up of beliefs represented with a database; desires that are represented as events that can trigger plans; and intentions that are represented through these plans. For example, a fact may come in from perception and match a desire, that of putting new facts into the database. This may result in further desires being matched and intentions (plans) leading to behaviors. Further information is available at the Jack developer's web site (www.agent-software.com.au).

Reviews of the agent literature (Etzioni & Weld, 1995; Franklin & Graesser, 1997; Wooldridge & Jennings, 1995)[2] reveal that, when attempting to define agency as dependent on the possession of a set of cardinal attributes, many of the attributes suggested could also be seen as characteristic of behavior that is best explained at the knowledge level. These include abstraction and delegation, flexibility and opportunism, task orientation, adaptivity, reactivity, autonomy, goal-directedness, flexibility, collaborative and self-starting behavior, temporal continuity, knowledge-level communication ability, social ability, and co-operation.

Both agent systems and KBSs are moving in the direction of modular components of expertise as a response to the problems of knowledge use and reuse to promote intelligent behavior in software. Domain ontologies form a significant subset of these KBS components. Increasingly, multi-agent systems are being produced that use such domain ontologies to facilitate agent communication at the knowledge level, for example, the agent network created as part of the Infosleuth architecture (Jacobs & Shea, 1996). Some agent systems also draw explicitly on models of problem-solving expert behavior developed in KBS research. The IMPS (Internet-based Multi-agent Problem Solving) architecture (Crow & Shadbolt, 1998) uses software agency as a medium for applying model-driven knowledge engineering techniques to the Web. It involves software agents that can conduct structured online knowledge acquisition using distributed knowledge sources. Agent-generated domain ontologies are used to guide a flexible system of autonomous agents driven by problem-solving models.

Even in pure KBS applications the notion of agency is being used as a design principle to facilitate the construction of the KBS itself. Thus in the DERA Future Organic Airborne Early Warning demonstrator (DERA final FOAEW report) the KBS is implemented as sets of computational agents each of which implements a task of an AEW operator. Using agents as task components in a KBS is a very natural design approach.

Agent architectures will be important within synthetic environments for modeling autonomous vehicles, and for exploring the doctrine of autonomous vehicles.

6.6 Architectural ideas behind the Sim_Agent Toolkit[3]

Since the early 1970s Sloman and his colleagues have been attempting to develop requirements and designs for an architecture capable of explaining a wide variety of facts about human beings. Sloman's ideas about cognitive architectures and the agent architecture toolkit (Sim_Agent) provide useful lessons about architectural toolkits and about process models of emotions. Further information is available at the CogAff web site <www.cs.bham.ac.uk/~axs/cogaff.html>.

---

[2] For online information about related US programs, see <www.darpa.mil/tto/mav.html>, <www.darpa.mil/haeuav/>, and <www.nosc.mil/robots/air/amgss/mssmp.html>.

[3] This section was drafted by Aaron Sloman and was revised by the authors.

6.6.1 Cognition and Affect

A human-like information processing architecture includes many components performing different functions all of which operate in parallel, asynchronously. This is not the kind of low-level parallelism found in neural nets (although such neural mechanisms are part of the infrastructure). Rather there seem to be many functionally distinct modules performing different sorts of tasks concurrently, a significant proportion of them are concerned with the monitoring and control of bodily mechanisms, for example, posture, saccades, grasping, temperature control, daily rhythms, and so on.

The very oldest mechanisms in the human architecture are probably all reactive in the sense described in various recent papers (e.g., Sloman, 2000). The key feature of reactivity is the lack of "what-if" reasoning capabilities, with all that entails, including the lack of temporary workspaces for representations of hypothesized futures (or past episodes), the lack of mechanisms for stored factual knowledge (generalizations and facts about individuals) to support generation of possible futures, possible actions, and likely consequences of possible actions, and the lack of mechanisms for manipulating explicit representations.

Both reactive and deliberative mechanisms require perceptual input and can generate motor signals. However, in order to function effectively both perceptual and action subsystems may have evolved new layers of abstraction to support the newer deliberative processes, for example, by categorizing both observed objects and events at a higher level of abstraction, and allowing higher-level action instructions to generate behavior in a hierarchically organized manner. More generally, different subsystems use information for different purposes so that a number of different processes of analysis and interpretation of sensory input occur in parallel, extracting different affordances from raw data extracted from the optic array. Recent work by brain scientists on ventral and dorsal visual pathways are but one manifestation of this phenomenon.

The interactions between reactive and deliberative layers are complex and subtle, especially as neither is in charge of the other, though at times either can dominate. Moreover, the division is not absolute: information in the deliberative system can sometimes be transferred to the reactive system (e.g., via drill and practice learning), and information in the reactive system can sometimes be decompiled and made available to deliberative mechanisms (though this is often highly error-prone).

For reasons explained in various papers available in the CogAff FTP site it is possible to conjecture that at a much later evolutionary stage a third class of mechanism developed, again using and redeploying mechanisms that had existed previously. The new type of mechanism, which has been provisionally labeled "meta-management", provides the ability to do for internal processes what the previous mechanisms did for external processes: namely it supports monitoring, evaluation, and control of other internal processes, including, for instance, thinking about how to plan, or planning better ways of thinking. For example, a deliberative system partly driven by an independent reactive system and sensory mechanisms can unexpectedly acquire inconsistent goals. A system with meta-management can notice and categorize such a situation, evaluate it, and perhaps through deliberation or observation over an extended period develop a strategy for dealing with such conflicts.

Similarly, meta-management can be used to detect features of thinking strategies, and perhaps in some cases notice flaws or opportunities for improvement. Such a mechanism (especially in conjunction with an external language) also provides a route for absorption of new internal processes from a culture, thereby allowing transmission between generations of newly acquired information without having to wait for new genetic encodings of that information to evolve. Through internal monitoring of sensory buffers the extra layer adds a kind of self-awareness that has been the focus of discussions of consciousness, subjective experience, qualia, etc. As with external processes, the monitoring, evaluation, and redirection of internal processes is neither perfect nor total, and as a result mistakes can be made about what is going on, inappropriate evaluations of internal states can occur, and attempts to control processing may fail, for example, when there are lapses of attention despite firm intentions.

Another feature of meta-management is its ability to be driven by different collections of beliefs, attitudes, strategies, and preferences, in different contexts, explaining how in humans a personality may look different at home, driving a car, in the office, etc. Besides the three main concurrent processing layers (reactive, deliberative, and meta-management) identified above that others have found evidence for, a number of additional specialized mechanisms are needed, including: mechanisms for managing short- and long-term goals, a variety of long- and short-term memory stores, and one or more global alarm systems capable of detecting a need for rapid global reorganization of activity (freezing, fleeing, attacking, becoming highly attentive, etc.), and also producing that reorganization.

For instance, whereas many people have distinguished primary and secondary emotions (e.g., Damasio, 1994) Sloman and his colleagues have proposed a third type, tertiary emotions, also sometimes referred to as *perturbances* (Sloman, 1998a; Sloman & Logan, 1999). Primary emotions rely only on the reactive levels in the architecture. Secondary emotions require deliberative mechanisms. Tertiary emotions are grounded in the activities of meta-management, including unsuccessful meta-management. There are other affective states concerned with global control, such as moods, which also have different relationships to the different layers of processing. Many specific states that are often discussed but very unclearly defined, such as arousal, can be given much clearer definitions within the framework of an architecture that supports them.

It looks as if various subsets of the capabilities described here arising out of the three layers and their interactions can be modeled in the architectures developed so far, for example, in Soar, ACT-R/PM, the Moffatt and Frijda architecture, and the various logic-based models that dominate the ATAL series of workshops and books like Wooldridge and Rao (1999).

However, only small subsets of these capabilities can be modeled at present. That is fine as far as ongoing scientific research is concerned: it is not fine when such models are offered as solutions to hard practical problems requiring modeling of complete human beings rather than some restricted human cognitive capability. Any realistic model of human processing needs to be able to cope with contexts including rich bombardment with multi-modal sensory and linguistic information, where complex goals and standards of evaluation are constantly interacting, where things often happen too fast for fully rational deliberation to be used, where everything that occurs falls does not always fall into a previously learnt category for which a standard appropriate response is already known, where decisions have to be taken on the basis of incomplete or uncertain information, and where the activity of solving one problem or carrying out one intricate task can be subverted by the arrival of new factual information, new orders, or new goals generated internally as a side effect of other processes.

Where the individual is also driving a fast moving vehicle or is under fire then it is very likely that a huge amount of the processing going on will involve the older reactive mechanisms, including many concerned with bodily control and visual attention. It may be some time before we fully understand the implications of such total physical immersion in stressful situations, including the effects on deliberative and meta-management processes. (For example, fixing attention on a hard planning problem can be difficult if bombs are exploding all around you. Can our models explain why?)

6.6.2 Sim_Agent and CogAff

Sloman and his colleagues general architectural toolkit Sim_Agent allows them to explore a variety of new ideas about complex architectures. It is not an architecture but a steadily developing toolkit for exploring architectures.

At present Sloman does not propose a specific overarching architecture as a rival to systems like Soar or ACT-R. He feels that not enough is yet known about how human minds work, and consequently any theory proposing *the* architecture is premature. Instead, he and his group have been exploring and continually refining a collection of ideas about possibly relevant architectures and mechanisms. Although the ideas have been steadily developing they do not believe that they are near the end of this process. So although one could use a label like *CogAff* to refer to the general sort of architecture they are currently talk about, it is not a label for a fixed design. Rather CogAff should be taken to refer to a high-level overview of a class of architectures in which many details still remain unclear. The CogAff ideas are likely to change in dramatic ways as more is learned about how brains work, about ways in which they can go wrong (e.g., as a result of disease, aging, brain damage, addictions, stress, abuse in childhood, etc.), and how brains differ from one species to another, or one person to another, or even within one person over a lifetime.

Sloman and his colleagues also wanted a toolkit that supports exploration of a number of interacting agents (and physical objects, etc.) where within each agent a variety of very different mechanisms might be running concurrently and asynchronously yet influencing one another. They also wanted to be able to very easily change the architecture within an agent, change the degree and kind of interaction between components of an agent, could speed up or slow down the processing of one or more submechanisms relative to others (Sloman, 1998b). In particular they wanted to be able to combine fairly easily different types of symbolic mechanisms and also subsymbolic mechanisms within one agent. The toolkit was also required to support rapid prototyping and interactive development with close connections between internal processes and graphical displays.

Because other toolkits did not appear to have the required flexibility, as the other toolkits tended to be committed to a particular type of architecture, Sloman and his colleagues built their own, which has been used for some time at Birmingham and DERA Malvern. Their toolkit is described briefly in Sloman and Logan (1999) and in more detail in the online documentation at the Birmingham Poplog FTP site (<ftp.cs.bham.ac.uk/pub/dist/poplog/>). The code and documentation are freely available online. It runs in Pop-11 in the Poplog system (which is inherently a multi-language AI system, so that code in Prolog, Lisp, or ML can also be included in the same process). Poplog has become freely available (<http://www.cs.bham.ac.uk/research/poplog/freepoplog.html>).

The toolkit is still being enhanced. In the short term they expect to make it easier to explore architectures including meta-management. Later work will include better support for subsymbolic spreading activation mechanisms, and the development of more reusable libraries, preferably in a language-independent form.

6.6.3 Summary

The Sim_Agent toolkit and the goals its developers have for it have some commonalties with other approaches. The need for a library of components is acknowledged. They emphasize that reactive behaviors are necessary and desirable, and that the emotional aspects arise out of the reactive mechanisms. It provides a broad range of support for testing and creating architectures. The toolkit provides support for reflection as a type of meta-learning. Other architectures will need to support this as well, particularly where the world is too fast paced for learning to occur during the task (John, Vera, & Newell, 1994; Nielsen & Kirsner, 1994).

The features that the toolkit supports helps define a description of architectural types. The capabilities that can be provided, from perception through to action and from knowledge to emotions, provide a way of describing architectures.

The major drawback is that none of the models or libraries created in Sim_Agent have been compared with human data directly. In defence of this, Sloman claims that, the more complex and realistic an architecture becomes, the less sense it makes to test it directly. Instead he claims that the architecture has to be tested by the depth and variety of the phenomena it can explain, like advanced theories in physics, which also cannot be tested directly.

6.7 Engineering-based architectures and models

There is a history of studying process control in and near industrial engineering that includes studying human operators. This approach is not (yet) part of mainstream psychology, and Pew and Mavor (1998) on the whole do not make many references to work in this field.

If tank operators and ship captains can be viewed as running a process, and we believe they can, there are a lot of behavioral regularities referenced and modeled in engineering psychology that can be generalized and applied to other domains. Major contributions in this area include

Reason's (1990) book on errors, Rasmussen's skill hierarchy (1983), the CREAM methodology for analyzing human performance (Hollnagel, 1998), and numerous studies characterizing the strengths and weaknesses of human operator behavior (de Keyser & Woods, 1990; Sanderson, McNeese, & Zaff, 1994).

Engineers have also created intelligent architectures. These architectures have almost exclusively been used to create models of users of complex machinery, ranging from nuclear power plants to airplanes. The models are often but not always tied to simulations of those domains. Their approach is generally more practical and interested in approximate timing and the overt behavior than in detailed mechanisms. These developers appear to be less interested in the internal mechanisms giving rise to behavior as long as the model is usable and approximately correct.

These models of operators include models of nuclear power plant operators, the Cognitive Simulation Model (COSIMO: Cacciabue et al., 1992), and the Cognitive Environment Simulation (CES: Woods, Roth, & Pople, 1987). AIDE (Amalberti & Deblon, 1992) is a model of fighter pilot behavior; the Step Ladder Model or Skill-based, Rule-based, Knowledge-based model is a generally applicable framework, which was originally formulated in electronics troubleshooting (e.g., Rasmussen, 1983).

We will look at a few operator models in more detail.

6.7.1 APEX

Apex (Freed & Remington, 2000; Freed et al., 1998) is a model of an air traffic controller. It has been based on psychology and perceptual regularities, such as provided by Boff and Lincoln (1986) and Sekuler and Blake (1994). It is probably best described as an engineering model because it has not been tested directly against human data. APEX is interesting because it models the whole operator, from perception to action, and the model interacts with a fairly complete simulation.

6.7.2 SMoC and CoCoM

The Simplified Model of Cognition (SMoC) (Hollnagel & Cacciabue, 1991) is an extension of Neisser's (1976) perceptual cycle, and describes cognition in terms of four essential elements: (a) observation/identification, (b) interpretation, (c) planning/selection, and (d) action/execution. Although these are normally linked in a serial path, there are other links possible between the various elements. The small number of cognitive functions in SMoC reflects the general consensus of opinion on the characteristics of human cognition as they have been developed since the 1950s. The fundamental features of SMoC are the distinction between observation and inference (overt vs. covert behavior), and the cyclical nature of cognition (cf. Neisser, 1976).

SMoC was formulated as part of the System Response Generator (SRG) project (Hollnagel & Cacciabue, 1991). SRG was a software tool developed to study the effect of human cognition (specifically actions and decision making) on the evolution of incidents in complex systems.

The Contextual Control Model (CoCoM: Hollnagel, 1993) is an extension of the SMoC, and addresses the issues of modeling both competence and control. In most models the issue of competence is supported by a set of procedures of routines that can be employed to perform a particular task when a particular set of predefined conditions obtains. CoCoM further proposes that there are four overlapping modes of control -- influenced by knowledge and skill levels -- that also influence behavior:

1. Scrambled control: where the selection of the next action is unpredictable. This is the lowest level of control.
2. Opportunistic control: where the selection of the next action is based on the current context without reference to the current goal of the task being performed.
3. Tactical control: where performance is based on some form of planning.
4. Strategic control: where performance takes full account of higher-level goals. This is the highest level of control.

The transition between control modes depends on a number of factors, particularly the amount of subjectively available time, and the outcome of the previous action. These two factors are interdependent, however, and also depend on aspects such as the task complexity and the current control mode.

CoCoM has been used in the development of the Cognitive Reliability and Error Analysis Method (CREAM: Hollnagel, 1998) within the field of Human Reliability Analysis. CREAM is a method for analyzing human performance when working with complex systems. It can be used in both the retrospective analysis of accidents and events, and in predicting performance for human reliability assessment. Extending CREAM is presented below as a useful project.

6.7.3 Summary

These architectures suggest that engineering models can provide useful behavior even when the internal mechanisms are not fully tested or perhaps even plausible. These architectures show that some of the difficulty in creating the architectures is due to the implicit and explicit knowledge that psychologists bring with them regarding plausibility. We believe this leads to more accurate models but slower development.

6.8 Summary of recent developments for modeling behavior

This section has reviewed several architectures. They show that it is becoming increasingly possible to create plausible and useful architectures, based on a variety of approaches.

An agreed, formal scheme for classifying architectures would be useful. This ideal system classification would note the sorts of tasks that each architecture is best at, supporting users to choose an architecture for a particular task. The best that we have found is Table 3.1 in Pew and Mavor (1998, pp. 98-105). Our Table 2 provides a summary of the architectures presented here in that format as a supplement to their table. In most cases the developers of the architectures have helped complete this table. We have included all relevant information of which we are aware for each architecture. Another approach is available from Logan (1998).

Developments in AI continue to be useful. The general AI methods discussed are not included in this table because they are not broad enough to be considered a cognitive architecture, but they are likely to be useful additions to architectures, either directly or indirectly. For example, genetic algorithms have been included in a proposed architecture (Holland, Holyoak, Nisbett, & Thagard, 1986), and planning algorithms have been included as adjuncts to Soar (Gratch, 1998). These developments will help extend architectures, providing algorithms for inclusion within architectures, particularly hybrid architectures.

There are several interesting trends to note. One is that the diversity of architectures is not decreasing. New, fundamental ideas on which to base architectures has widened from simply problem solving. For example, EPAM is based on pattern recognition, and PSI and architectures created in the Sim_Agent Toolkit are based on ideas about emotions.

Another interesting trend is that some aspects of the architectures are starting to merge and be reused. The interaction aspects of EPIC have been reused by Soar and by ACT-R. The Nottingham Interaction Architecture is similar in some ways and getting similar reuse (e.g., Jones et al., 2000). These strands are becoming to quite similar to each other (Byrne, Chong, Freed, Ritter, & Gray, 1999), and are quite likely to merge in the future.

The importance of model usability is becoming more recognized. Cogent provides an example of how easy a modeling tool should be to pickup and use. Similar developments with Soar and ACT-R are starting to emphasize reusable code, better documentation, and better tutorial materials. Other architectures will have to follow suit to attract users and to train and support their existing users. Newell (1990) wrote about the entry level (the bar) being raised as architectures develop through competition. It is interesting that usability is perhaps the first clear comparison level.

Table 2. Comparison of architectures covered in this report.

		Submodels
Architecture	Original purpose	Sensing and perception
1 EPAM	Model high-level perception, learning, and memory.	Visual, auditory perceptual discrimination is in real-time (assuming feature-based description of objects).
2 SDM	Simulation of cerebellum as a content-addressable memory.	Can be used to recall the nearest stored memory to any encoded perceptual input.
3 PSI	Exploring the interaction of cognition, motivation, and emotion to build an integrated model of human action regulation.	Optical perception by "Hypercept"-process. This process scans the (simulated) environment for basic features, raises hypotheses about the sensory schemas to which the features may belong and tests these hypotheses by subsequent scanning of the environment (comparable to saccadic eye-movements). If a pattern is not recognizable, a new schema will be generated.
4 Cogent	Design environment for modeling cognitive processes.	Input buffers that can be modified to represent vision and hearing.
5 Jack as an example of BDI architectures	To constitute an industrial-strength framework for agent applications.	JAVA methods + inter-agent messaging.
6 Sim_Agent toolkit	Exploring architectures using rapid prototyping.	Defined by methods for each agent class.
7 Engineering-based models, e.g., APEX	Provide models of humans in control loops.	Varies, but exists for most models.

Table 2 (continued).

	Submodels		
	Working/ Short-Term memory	Long-Term Memory	Motor Outputs
1	4-7 slot STM; in some versions (e.g., EPAM-IV), there is a more detailed implementation of a auditory (Baddeley-like) STM & visual STM.	Discrimination net. In recent versions, nodes of the discrimination net are used to create a semantic net and productions.	Eye movements, simple drawing behavior.
2	Not modeled.	Sparse Distributed Memory models are related to PDP and neural net memory models.	Motor sequences can be learned. Nearest match memories can be sequences that could be behaviors.
3	The *head* of a protocol memory that permanently makes a log of the actions and perceptions.	The remnants of the logs decay with time. Strings of the log that are associated with need satisfaction or with pain will be reinforced and therefore have a greater chance to survive and to form a part of the long-term memory than neutral sequences of events.	Basic motor patterns (actions) are combined to form complex sensory-motor-programs by learning (i.e., by reinforcement of the successful sensory-motor-patterns in the logs).
4	Various types supported.	Various types supported.	Simple buffer representation of commands.
5	Object oriented structures (JAVA), plus relational modeling support (JACK).	All JAVA support including database interfaces etc. Support for data modeling in JAVA and C++ using the JACOB (JACK Object Builder) component.	JAVA methods.
6	List structures.	List structures, rules, and arbitrary Pop-11 data structures, but can also use neural nets if required.	Defined by methods for each agent class.
7	Usually simple but extant.	Usually simple but extant.	Usually extant but usually not complex.

Table 2. Comparison of architectures covered in this report (continued).

	Knowledge Representation	
Architecture	Declarative	Procedural
1 EPAM	Chunks, schemas (templates); using nodes in discrimination net.	Productions using nodes in a discrimination net.
2 SDM	A sparse set of memory addresses, where the data *is* the address.	Memories naturally form sequences that could be considered as procedures.
3 PSI	Sensory and sensory-motor patterns consisting of pointer structures forming schemas. A schema includes information about the more basal elements and the relations of these elements in space and time, including language patterns pointing to sensory and sensory-motor patterns (implementation in progress).	Sensory-motor-patterns forming automatisms.
4 Cogent	Numbers, strings, lists, tuples, connectionist networks.	Production rules, connectionist networks, Prolog.
5 Jack and other BDI architectures	Object oriented structures (JAVA), plus relational modeling support.	JACK plans and JAVA methods.
6 Sim_Agent toolkit	List structures, and arbitrary Pop-11 data structures (e.g., could be constrained to express logical assertions, but need not be). Could use neural nets or other mechanisms.	Rule sets and arbitrary Pop-11 procedures that can also invoke Prolog or external functions.
7 Engineering models, e.g., APEX.	Varies, but usually simple.	Varies, but usually simple. Many use some form of schemas.

Table 2 (continued).

	Higher-Level Cognitive Functions			
	Learning	Planning	Decision Making	Situation Assessment
1	Chunking, creation of schemas and production learning is online (incremental) and stable against erroneous data.	Connections between templates are used in planning.	Knowledge based.	Overt and inferred.
2	By incrementing weights across a probability distribution.	Does not plan, but can remember plans.	Iterative memory recall process.	Can learn a set of assessments and generalize these.
3	Associative and perceptual learning; operant conditioning: sensory-motor learning, learning goals (situations that allow need satisfaction) and aversions (situations or objects that cause needs).	Built-in hill-climbing procedure: action schemata (i.e., sensory-motor-patterns) are recombined to form new plans. If planning is unsuccessful, or impossible due to a lack of information, trial-and-error procedures used to collect environmental information.	Expectancy-value-principle.	Built in as part of problem solving.
4	Common methods within connectionist modules.	Could be implemented in rule modules.	Specific to module type. Can vary.	None built in (users can specify).
5	None built in (users can specify as required by their architecture).	None built in (users can specify as required by their architecture).	Includes BDI computation model.	Includes BDI computation model.
6	None built in. Wright, for example, included simple forms of deliberative mechanisms and meta-management in his PhD work.	None built in (users can specify as required by their architecture). Logan's A* with bounded constraints available, among others.	None built in (users can specify as required by their architecture).	None built in (users can specify as required by their architecture).
7	Usually not extant.	Varies, some models do well.	Usually good; this is the domain of these models.	Varies, often implicit.

Table 2. Comparison of architectures covered in this report (continued).

	Multitasking	
Architecture	Serial/Parallel	Resource Representation
1 EPAM	Serial processing; learning done in parallel.	Limited STM capacity, limited perceptual and motor resources (uses time parameters).
2 SDM	Fully parallel recall process, serial recall of sequences.	Too low-level for this to be explicit.
3 PSI	System tries to fulfill different needs (i.e., water, energy, pain-avoidance, etc.); interrupts goal directed behavior to profit from unexpected opportunities.	Allocation of time to *run* an intention according to strength of underlying need and according to expectancy of success.
4 Cogent	Modules can work in parallel, but information is passed between them serially.	Would vary with the knowledge included in modules.
5 Jack and other BDI architectures	Supports multiple computational threads that are handled safely within the JACK Kernel -- achieving atomic reasoning steps.	Agents have time perception. Time can be real-time or simulated time (dilated, externally synchronized, etc.).
6 Sim_Agent toolkit	Discrete event simulation technique, with rule sets within each agent time-sliced, as well as different agents being time-sliced.	Allocation of cycles per time-slice can be made for each rule set, or for each agent. There are no built in memory resource limits. That will differ for each architecture type created.
7 Engineering models, e.g., APEX	Varies, sometimes explicit models.	Varies. Those that interact with simulations are more advanced.

Table 2 (continued).

	Goal/Task Management	Multiple Human Modeling	Implementation Platform
1	Bottom up + 1 main goal per task simulated.	Potential through multiple EPAM modules.	Mac, PC (any system supporting Common Lisp). Graphical environment supported only for Macintosh.
2	None.	None.	UNIX (easily ported).
3	There is a steady competition of the different needs/motives to rule. The strongest will win and will inhibit the others taking command.	Potential through multiple PSI models.	Windows 95, 98, NT.
4	None built in. Users can specify through module selection and programming.	None.	UNIX (X windows). Microsoft Windows soon.
5	Built in. JACK Language includes: wait_for(condition), maintenance conditions, meta-level reasoning, etc.	Allows multiple agents running together or distributed, to interact and communicate as a team or as adversaries. JACK also allows extensions to the basic model, e.g., team models, etc.	Runs on all platforms that support JAVA 1.1.3 or later.
6	None built in (users can specify as required by the architecture).	The toolkit allows multiple agents to sense one another, act on one another, and communicate with one another.	Runs on any system supporting Poplog (and for graphics the X window system). Tested on Sun/Solaris, PC/Linux, Dec Alpha/UNIX (Compaq). Should also run on HP UNIX, SGI UNIX and VAX VMS. Should work without graphics on Windows NT Poplog.
7	Varies. Some advanced.	Some have none; some work in teams.	Varies. Not usually designed for dissemination.

Table 2 Comparison of architectures covered in this report (continued).

Architecture	Language	Support Environment
1 EPAM	Common Lisp.	Lisp programming + editing tools. Some graphical utilities for displaying eye movements, structure of discrimination tree, and task. Customized code used for each task modeled.
2 SDM	C (Elliman is currently recoding in JAVA).	None.
3 PSI	Pascal (Delphi 4).	Delphi 4 features.
4 Cogent	Prolog.	Graphic and textual editors.
5 Jack and other BDI architectures	JAVA. JACK is written in and compiles into pure JAVA.	JACK Make utilities, and all available JAVA tools. GUIs for editing and debugging agent structures will be available soon.
6 Sim_Agent toolkit	Pop-11 (but allows invocation of other Poplog languages (Prolog, Common Lisp, Standard ML, & external functions, e.g., C).	Poplog environment, including VED/XVED, libraries, incremental compiler, etc.
7 Engineering models, e.g., APEX	Varies.	Often simple.

Table 2 (continued).

	Validation	Comments
1	Extensive at many levels.	EPAM models focus on a single, specific information processing task at a time. Not yet scaled up to multitasking situations. Used in high-knowledge domains (e.g., chess, with about 300,000 chunks).
2	None.	This is a system component. A good way of representing long-term memory for patterns and motor behaviors in a larger system.
3	Achievement data and parameters of behavior have been compared between subjects and models in two different scenarios (BioLab and Island). Different human subjects can be modeled by varying parameters.	
4	Would be by architecture. Some have been done by modeling previously validated models.	
5	Would be by architecture. None known.	
6	Would be by architecture. None known.	
7	By model. Usually validated with expert opinion. Some may be compared with data.	There is a wide range of models here.

7. Review of recent developments and objectives: Specific projects

We now examine specific projects within the general application areas noted in Chapter 6, broadly grouped into projects that support the objectives in the previous chapters, that is, of providing more complete performance, supporting integration of models, and improving model usability. The format of the projects follows the general format used in Pew and Mavor (1998). Where appropriate, this summary also comments on the feasibility and concerns that may arise if the projects are implemented in Soar, a current common approach for computer generated forces. The estimates are optimistic, but uniformly optimistic to allow comparisons. They are in terms of programmer or analyst time, and assume adequate supervision and cooperation with other organizations.

7.1 Projects providing more complete performance

The projects presented here roughly address the issues raised in Section 4. They are grouped in to three main categories. We also note some additional uses that models of behavior can be or are being put to in synthetic environments before concluding.

7.1.1 Gathering data from simulations

It is very clear and consonant with Pew and Mavor (1998, Ch. 12) that data needs to be gathered to validate models of human and organizational behavior. An approach at which they hint is to instrument synthetic environments. Synthetic environments should be instrumented not only for playback, but in a way to provide data for developing and testing models. While the data is not directly equivalent to real world behavior, as the environment becomes more realistic, the data should become more realistic as well.

A uniform representation for this data should be created. This representation should be human readable, at least in some formats.

Creating summary measures will also be necessary. Otherwise the sheer volume of data may preclude its analysis. The individual actions of control are not likely to be useful on their own (e.g., pressing an accelerator), but will be required to build higher-level summaries. Creating these summaries is likely to represent an additional research agenda item requiring AI, domain knowledge, and some feeling for behavioral data.

The payback could be quite large for developing models. Analysis of this data might also provide insights into the quality of the simulation (e.g., how quickly could someone act and whether they were limited by the simulation's ability to display information), and provide insights about the implementation of doctrine (e.g., how often tanks actually follow doctrine). When done in cooperation with a simulator's developers, the resources required for this task could be quite modest. Otherwise, it could take some time. Developing initial automated summaries is a six to twelve month effort.

7.1.2 Models that learn in synthetic environments

Work on creating agents in synthetic environments has been successful, but one aspect that has not been modeled that would be particularly useful is learning. A worthwhile project would be to take a learning algorithm and put it to use within a synthetic environment, either as (part of) a problem solver or as an observer. There are a variety of learning algorithms and models that would be appropriate, for example, connectionism, one of the hybrid learning architectures developed within the ONR program (Gigley & Chipman, 1999), Programmable User Models (Young, Green, & Simon, 1989b), Soar with learning turned on, ACT-R, EPAM, or any of a wide variety of machine learning algorithms.

This work is likely to be difficult. This task is large and would allow multiple subprojects to be attempted. It could be supported by a wide range of resources. Including learning with problem solving has been difficult in the past, but it is likely to lead to more accurate agents and ones that may be useful for testing and developing tactics.

Soar models exist that function fairly well in a synthetic environment. If these could be used, a small project of a programmer-year or two should be able to create an initial model that learns in a synthetic environment. Attaching a learning component to find regularities in behavior is likely to take at least that much time. Both projects would provide potential PhD topics and are broad enough to be supported by a wide range of resources.

7.1.3 Using tabu search to model behavior

The internal architecture of a combatant might be constructed from a perceptual module that is closely coupled to the synthetic environment, and can be modified by plug-in items that alter the incoming data to be processed (night vision aids, etc.). The results of perception are crudely classified using a learning system such as a multi-layer perceptron, which triggers a rapid emotional response and consequent reactive behavior. This behavior might be generated using an SDM that finds the nearest match to previous scenarios and is capable of producing a sequence of outputs rather than a single state result. Both perception and emotional response are calibrated by a perceptual and personality model that may be unique to individual entities, albeit assigned from a known distribution.

The cognitive processing would be rule-based using an established cognitive model, for example, ACT-R (possibly reprogrammed in JAVA), with planning activities augmented by a tabu search approach. There would be interactions between the state of the entity (including its emotional state) and the cognitive processing based on psychological data on human performance under stress. This approach is similar and perhaps a generalization of Sloman's meta-architecture, and the Soar and PSI architectures.

7.1.4 Unified theory of emotions

There are three specific projects related to modeling emotions that we can propose: (a) adding general emotional effects, (b) adding reactive emotions, and (c) testing emotional models with performance data. While work is ongoing implementing models like this in Soar (Chong, 1999;

Gratch, 1999) and ACT-R (Belavkin et al., 1999), the domain is large. Projects can range from a few months to implement a simple emotional effect to several years or decades to incorporate a significant amount.

(a) Adding general emotional affects. As noted above, it is possible to start to realize emotions and affective behavior within toolkits like Sim_Agent and general cognitive architectures like ACT-R and Soar. Including emotions will provide a more complete architecture for modeling behavior and a platform for performing future studies of how emotions effect problem solving. Including emotions may also provide a way to duplicate personality and provide another approach to providing appropriate variations in behavior. Hudlicka (1997) provides a list of intrinsic and extrinsic behavior moderators that could be modeled similar to the categories suggested in Ritter (1993b). Boff and Lincoln (1986) provide a list of regularities related to fatigue and other related stressors that might be considered for testing against the model. By making ACT-R's motivation sensitive to local performance, we have fit the Yerkes-Dodson law (Belavkin & Ritter, 2000).

These models should move from applying to a single task to applying to multiple tasks. They would then become modifications to the architecture and thus reusable.

(b) Adding Reactive emotions. It is worthwhile to model reactive processes as well as slower-acting cognitive behaviors. The effect of repeated levels of stress also changes the state of the competence in important ways. A proportion of troops engaged in active combat will become ineffective as a result of fear and stress-fatigue. This will be increased by the number of casualties taken by a given platoon, the length of time without sleep, the weather conditions, the perceived chance of survival, and so on. Modeling these effects at the micro-level of individuals, following known distributions, would advance the realism of simulations in interesting ways.

In production system architectures, these emotions can initially be implemented by changing the decision (rule matching) procedure, adding rules to make parameter changes, and by augmenting working memory to include affective information (e.g., an operator or state looks good or bad). These types of changes are being applied to an existing model, which matches adult behavior well, to better match children's more emotional behavior (Belavkin et al., 1999). These emotional effects should improve the match to the children's performance by (a) slowing down performance in general, (b) slowing down initial performance as the child explores the puzzle driven by curiosity, and (c) abandoning the task if performance is not successful. This work should be extended and applied more widely.

(c) Testing emotional models with performance data. Many of the theories of emotions proposed have not been compared with detailed data. Partly this may be because there is not always a lot of data available on how behavior changes with emotions. It is no doubt a difficult factor to manipulate safely and reliably. But the models must not just be based on intuitions.

The use of simulators may provide a way to obtain further data, with some validity. Better instrumentation of some primary features of emotions (e.g., heart rate, blood pressure) is providing new insights (Picard, 1997; Stern, Ray, & Quigley, 2001) and will be necessary for testing models of emotions.

Some argue that emotions are necessary for problem solving. Examples of brain damaged patients are put forward (e.g., Damasio's (1994) Elliot) who have impaired problem solving and impaired emotions. It is not clear that emotions per se are required, or if multiple aspects of behavior were impaired as well as emotions by the trauma. Others agree from first principles that emotions can improve problem solving (Belavkin, 2001). Data and a model may help answer whether this is true. Clearly, AI models of scheduling do not have the same troubles scheduling an appointment despite their lack of emotion.

7.1.5 Modeling errors

There are two premises that underpin the modeling of erroneous behavior. The first premise is that the attribution of the label error to a particular action is a judgment made in hindsight. The identification of the erroneous action forms the starting point for further investigation to identify the underlying reasons why a particular person executed that particular action in that particular situation. In other words, the erroneous action arises as the result of a combination of factors from a triad of sources: the person (psychological and physiological factors), the system (in the most general sense of the term), and the environment (including the organization in which the system is deployed).

The second premise acknowledges that an error is simply another aspect of behavior. In other words, any theory of behavior should naturally encompass erroneous behavior. The behavior can be adjudged as erroneous only with respect to a description of what constitutes correct behavior.

Once these two premises are accepted, it becomes apparent that modeling erroneous behavior is actually an inherent and important part of modeling behavior. If the psychological and physiological limitations of human behavior are incorporated into a model of human behavior, then particular types of erroneous behavior should naturally occur in certain specific situations. The corollary of this argument is that an understanding of erroneous behavior can be used as the basis for evaluating models of behavior. So, if a human performs a task correctly in a given situation, the model should also be able to perform the task correctly in the same situation. If the situation is changed, however, and the human generates erroneous behavior as a result, the model should also generate the same erroneous behavior as the human in the new situation, without any modifications being required to the model.

Modeling error therefore depends on understanding the concept of error -- its nature, origins, and causes -- and central to this is the need for an accepted means of describing the phenomenon (Senders & Moray, 1991). In other words, a taxonomy of human error is required.

The utility of the taxonomic approach, however, depends on the understanding that the taxonomy is generated with a particular purpose in mind. In other words, the taxonomy has to reflect:

1. A particular notion of what constitutes an error.
2. A particular level of abstraction at which behavior is judged to be erroneous.
3. A particular task or domain.

There is a need to be very clear about the classes of errors and their origin in the models so that the appropriate ones can be included. In the military context, for example, a major source of error is communication breakdown. One approach to developing an appropriate taxonomy of errors for the military domain is to use the scheme that lies at the heart of the Cognitive Reliability and Error Analysis Method (CREAM: Hollnagel, 1998). The CREAM purports to be a general purpose way of analyzing human behavior in both a retrospective and a predictive manner. Although the method was developed on the basis of several years of research into human performance, mainly in the process industries, it is intended to be applicable to any domain.

The CREAM uses a domain-independent definition of what constitutes an erroneous action (also called error modes or phenotypes). One of the goals of the CREAM is to be able to identify the chain of precursors for the various error modes. This is achieved by means of a set of tables that define categories of actions or events. At the highest level there are three types of tables:

1. Personal (or operator),
2. Technological (or system), and
3. Organizational (or environment).

Within these categories there are sub category tables. So, for example, the personal tables include observation, interpretation, planning, and so on.

The individual actions or events are paired together across tables on the basis of causality or to use the more neutral terms, in a consequent-antecedent relationship. When CREAM is used to analyze a particular accident or incident retrospectively, the aim is to build up the list of possible chains of events and actions which led to the accident or incident.

The contents of the tables are domain specific, so the first step in developing the taxonomy for agents in synthetic environments depends on identifying the appropriate categories of events and actions for the military domain. These categories and the links between individual actions or events will be generated from a combination of knowledge elicited from domain experts and a review of the appropriate literature.

The second step is to generate the possible chains of actions and events that precede the various error modes, based on information available from reports of real accidents or incidents. This process will involve access to desensitized accident or incident reports -- like those used in the Confidential Human Factors Incident Reporting Programme (CHIRP: Green, 1990) originally operated by the RAF Institute of Aviation Medicine -- which can be analyzed and coded using the domain-specific CREAM tables generated in Step 1. Where omissions from the tables are detected, or links between actions do not already exist, these should be added to the tables.

The possible causal chains of events or actions generated by the second step will provide the basis for a specification of behavior in a particular situation. Models of behavior should yield the same sequences of actions and events in the same circumstances. The specification of behavior can therefore be used to test the models of behavior for compliance, during development, with the model being modified as appropriate to match the specification.

In addition, the results of the analysis of the incident behavior provide a basis for evaluating the veracity of synthetic environments. Performance in the real world (as described in the incident reports) can be compared with the way people behave when performing in the synthetic environment. There should be a high degree of correspondence between the two. If there is a mismatch, this suggests that there is a difference between the real world and the synthetic environment, which may be worth further investigation to identify the source of the difference.

One other beneficent side effect of the CREAM analysis is that the resultant chains of actions and events can be used in training personnel how to manage error. If common chains of actions or events can be identified, it may be possible to train personnel to recognize these chains, and take appropriate remedial action before the erroneous action that gives rise to the incident is generated.

Initial models that include errorful behavior can best be created with an existing model. One to three years of work should lead to a model that includes errors and has been validated against human behavior.

7.1.6 Unified theory of personality

It would be useful to identify features that lead to modeling personality, problem-solving styles, and operator traits. While models that choose between strategies have been created, there are few models that exhibit personality by choosing between similar strategies (although see Nerb, Spada, & Ernst, 1997 for an example used to put subjects in a veridical but artificial social environment). Personality will be an important aspect of variation in behavior between agents.

Including personality requires a task (and the model) to include multiple approaches and multiple successful styles. It is these choices that can thus appear as a personality. If the task requires a specific, single approach, it is not possible to express a strategy. Psychology, or at least cognitive psychology, has typically not studied tasks that allow or particularly highlight multiple strategies. Looking for multiple strategies has also been difficult because it requires additional subjects and additional data analysis that before has not represented real differences in task performance. Differences in strategies, however, do lead to variance in behavior (Delaney, Reder, Staszewski, & Ritter, 1998; Siegler, 1987).

There appear to be at least the following ways to realize variance in behavior that might appear like personality: learning, differences in knowledge, differences in utility theory and initial weighting, and differences in emotional effects. Such a model would fulfill a need for a source of regular, repeatable differences between agents in a situation.

All of the current cognitive architectures can support models of personality. These types of changes should be straightforward, as long as there are multiple strategies. In Soar, personality can be expressed as differences in task knowledge, as well as differences in knowledge about strategy preferences either absolutely or based on a different sets of state and strategy features. ACT-R appears to learn better and faster which strategy to use compared with a simple Soar model, but requires additional state (Ritter & Wallach, 1998). Models in both architectures can, however, modify their choice of strategies. The role of (multiple) strategies has been investigated within the EPAM architecture in several tasks, such as concept formation (Gobet et al., 1997); and expert memory (Gobet & Simon, 2000; Richman, Gobet, Staszewski, & Simon, 1996; Richman et al., 1995).

These models could also be crossed with emotional and other non-cognitive effects to see how personality types respond differently in different circumstances (broadly defined). This could even be extended to look at how teams with different mixes of personalities work together under stress.

The amount of work to realize a model in this area will depend on the number of factors taken account of by the model. Providing a full model of personality and how it interacts with tasks and with other models is a fantasy at this point. However, a minimal piece of work would take an existing model and give it more of a personality. A more extensive project over a year or two would apply several of these techniques and see how it starts to match human data.

7.1.7 Understanding expectations of behavior

In order to understand what it means to provide realistic behavior, it will be necessary to understand what people expect from other people, and to understand what aspect of an adversary is necessary for training. (These two are different.) In one sense, this means understanding the Turing test: what is necessary to appear human? More important, however, is knowing what is necessary to train people. A model that passed the Turing test and appeared human might be weaker or unusual in some way, such that training with the model did not transfer, or incorrect behavior was learned.

It would be a useful exercise to study which characteristics of behavior make a model appear human (so-called believable agents). The model must start with competencies; it must be able to perform tasks. It is likely that it must also include errors, hesitations, and variation in behavior.

Work with the Soar Quake-bot examining how firing accuracy and movement speed make agents believe is an example of what is required (Laird & Duchi, 2000) to understand what people think is human. The Soar Quake-bot has been tested at several settings of such things are firing accuracy, and then observers are asked to rate its human-ness. The measure of humanness, however, does not tell you about how good the Quake-bot is to train against, and what aspects of the Quake-bot should be made more (or less) human in order to improve training. The current belief is that more human makes a better opponent to train against, but we do not know of any evidence to support this belief.

A useful six-month to one-year study would be to examine a range of models and humans in a synthetic environment, noting observers' comments and behavior towards a range of behavior. It might be these aspects above that make an agent appear human, but it might also include implicit effects, such as second-order (or lagged) dependencies in behavior. The results would be important for training, but also useful for models used in analysis. This project is similar to, but conceived of separately from, a similar call proposed by Chandrasakaren and Josephson (1999) to develop a better understanding of how and how far to develop models.

The results are also essential for understanding how to help modelers. The results will point out the most likely mismatches to be left in models because modelers do not consider such behavior abnormal. This will provide suggestions about where comparisons with data are particularly needed by models. As this is basically experimental work less resources are needed, but the time to run the experiment and analyze the results will take up to a year for preliminary studies.

7.1.8 A model of situation awareness and Rapid Decision Making

Novices have to do problem solving, and experts can do problem solving but save effort (or improve their problem solving performance) by recognizing solutions based on the problem. Viewed broadly, a model that does this transition starts to provide an explanation of situation awareness and rapid decision making (RDM) as a result of expertise and recognition.

Able (Larkin, 1981) and its recent reimplementations (Ritter et al., 1998b) provide a simulation explaining the path of development from novice to expert in formal domains (i.e., those where behavior is based on manipulated equations such as physics or math). The early (or barely) Able model works with a backward chaining approach, that is, it starts with what is desired and chooses domain principles to apply based on what will provide the desired output. This approach is applied recursively until initial conditions are found. The chunking mechanism in Soar gives rise to new rules that allow the model to use a forward chaining method that is faster. That is, from the initial conditions new results are proposed. The rules are applied until the desired result is found. Students at Nottingham have applied the Able mechanism to several new domains. Their examples are available at <www.nottingham.ac.uk/pub/soar/nottingham/>.

Work could be done to translate this mechanism, which has worked in Lisp and in several versions of Soar, into other architectures, and extend it from a simulation to a full process model. This would require a rather modest amount of effort, less than a programmer-year to get started if the programmer was familiar with Soar. Applying it in a realistic domain would take longer.

7.2 Projects supporting integration

Integration is approached in two ways here: integrating model components and integrating the model with simulations in more psychologically plausible ways. Several projects described in this subsection could be equally at home in the set of projects for making modeling routine, because the two areas are related.

7.2.1 Models of higher-level vision

It has been argued that an understanding of higher-level vision is necessary for continued development of models in synthetic environments (Laird, Coulter, Jones, Kenny, Koss, & Nielsen, 1997). We agree (Ritter et al., 2000). Neisser's (1976) perceptual cycle is just starting to be explored.

There are several areas of higher-level vision (HLV) that are of particular interest for military modeling. These areas include:

1. How information from long-term memory indicates incoming danger or serious change in the environment.

2. How HLV directs attention.
3. How HLV integrates various aspects of information, or integrates information occurring at different times.
4. How HLV can be used to facilitate learning.
5. How HLV can be used in planning and problem solving.

To put it simply, HLV is at the interface between lower-level vision (LLV) and postulated memory entities such as productions, schemas, concepts, and so on. At the present time, this interface is poorly understood, perhaps because LLV and long-term memory are not understood in a sufficiently stable way. (However, see Kosslyn & Koenig, 1992 for neuropsychological hypotheses about HLV.)

Most models of cognition such as Soar and ACT (actually, most architectures reviewed by Pew and Mavor, 1998) use modeller-coded information, which avoids dealing with the interface between LLV and long-term memory constructs. Neural nets for vision have been used to go from pixel-like information to features or even higher, but have not been incorporated into higher-cognition models. CAMERA (Tabachneck-Schijf, Leonardo, & Simon, 1997) and to a certain extent EPAM (Feigenbaum & Simon, 1984; Richman & Simon, 1989) explore ways in which features may be extracted from low-level representation, and may be combined into long-term memory constructs.

The relationship of HLV and problem solving is undoubtedly an area where more research should be carried out. For example, modeling instruction and training requires a theory of how low-level acoustic input merges with low-level visual input and connects to long-term memory knowledge. In some cases vehicles and gunfire will be heard rather than seen, and this will direct visual attention in the appropriate direction. Perceptual models of hearing are also well developed and exploited with dramatic success in, for example, the MPEG-2 compression standard that is likely to form the basis for much broadcast and recorded sound in the future. The variance amongst individuals is large for both auditory and visual perception, and both processes are degraded temporarily or permanently by intense overload as is likely in a military environment.

Work extending this approach to create integrated architectures (Byrne et al., 1999; Hill, 1999; Ritter & Young, in press) is ongoing. Significant progress will require at least a year-long project, and a longer period would be more appropriate.

7.2.2 Tying models to tasks

It should be easier to tie cognitive models to synthetic environments in psychologically plausible ways. There are two approaches that seem particularly useful and plausible that we can ground with particular suggestions for work. They are consistent with Pew and Mavor's (1998, p. 200) short-term goal for perceptual front ends.

The first approach is to provide a system for cognitive models to access ModSAF's display and pass commands to it. This approach has the advantage that it hides changes in ModSAF from the programmer/analyst and from the model. The disadvantage is the need for ModSAF experts, programmers, users, time, and money to make it work. There has been such a system created for Soar models to use ModSAF (Schwamb, Koss, & Keirsey, 1994), but it is our impression that this system, although it was quite useful, needs further development and dissemination.

The second approach is to create a reusable functional model of interaction based on a particular graphics system or interface tool (as does the Nottingham Interaction Architecture and ACT-R/PM). A functional rather than a complete model may be more appropriate here as a first step. This functional approach has been already created in Tcl/Tk (Lonsdale & Ritter, 2000), Garnet and Common Lisp (Ritter et al., 2000), Visual Basic (Ritter, 2000), Windows bitmaps (St. Amant & Riedl, in press), Windows 98 objects (Misker, Taatgen, & Aasman, 2001), and most recently in Java. They could be created in Amulet, X-windows, Delphi, or a variety of similar systems, each of which allows models to interact with synthetic environments through a better programming interface. A functional model would then provide the necessary basis for improving the accuracy and psychological plausibility of interaction.

This approach could also support creating cognitive models in general, such as for problem solving, working memory, and the effect of visual interaction. These could be later assimilated back into models and architectures in the synthetic environments.

A good programmer can create an initial system in about two weeks. Integrating and applying these models takes several months to a year.

7.2.3 Review of existing simulations

In order to provide for reuse and to understand the current situation, a review of simulation systems used (for as broad a geographic region as possible, working with allied nations if possible) should be created that is similar to the listing in Pew and Mavor (1998, Ch. 2 Annex). This listing, for example, could initially be created by an intercalated year (co-op) student and then maintained as part of standing infrastructure. This listing could provide an initial basis for understanding what the total needs were and the totality of current simulation efforts.

7.2.4 Focus on a flagship task

Supporting all the uses of synthetic forces as shown in Table 1 with a single model of behavior is probably impossible in the short term. The uses of simulations in operations research, in training individual group behavior, and in examining new materials or doctrine are too disparate to be met by a single approach. While the various levels and uses of simulations mentioned here are related by the real world they all represent, it does not appear to be possible in the next five to ten years to integrate them to the extent to which the real world is integrated.

While there may be some systems that allow multiple use, and there will certainly be some reuse between these areas, a focus for work must be selected. Therefore, a more narrow focus on the most important uses for DERA and its customers should be adopted. A selective focus on the most approachable or most natural set of uses is more likely to be successful in the short term, and may provide a better foundation upon which to build in the long term. Discussion of these issues should be grounded, if possible, with a set of potential uses with possible systems and domains that will be used in the next five to ten years. Complete unification is not likely in that time period, but significant reuse should be strived for.

Having a focus would also support the choice of a specific application. Applications can be then chosen with a user audience in mind. Having a specific audience will help the application to be useful and to be seen as useful by a well-defined user community.

Work that attempts to serve too many needs will serve all of them poorly. Projects and research programmes will have to pick a domain and an application (or two), and work with them. This application could be an existing use or application, such as the Purple Link exercise in 1997, or it could be a new use. Work with simulations for training often have high payoffs. Augmenting existing training would be a natural place to consider starting.

The students being trained could also be coopted to help test the simulation. Apocryphal tales from MIT suggest that building computer-based tutors to deliver instruction is as useful for learning as using the resulting tutors. Creating and validating these models would be good training for such students as well.

7.2.5 A framework for integrating models

Perhaps the most significant current requirement is a way to integrate multiple cognitive and behavior architectures into synthetic environments. Currently, it takes a large amount of effort to introduce new models of behavior and connect them directly to simulations via the DIS protocol. Coupling cognitive architectures to a simulation via ModSAF is probably marginally easier because ModSAF, while difficult to use, provides physical models and an interface to the network. The left-hand side of Figure 1 shows the organization of systems like Tac-Air Soar that interact with ModSAF to generate behavior.

A worthwhile medium- to long-range goal would be to develop utilities to support making a tool like ModSAF even more modular. The core activities of supporting communication across the network for simulation and supporting the physical model need to be provided, but are not of particular interest for modeling behavior.

Efforts have attempted to provide similar interfaces for Soar; however, they have never been fully successful. They have made hooking Soar up easier, but have not yet made it easy (e.g., Ong, 1995; Ong & Ritter, 1995; but also see the Soar home page for later utilities <bigfoot.eecs.umich.edu/~soar/>). Work on the Tank-Soar simulator (provided as a demo in the latest release of Soar, Soar 8.3) might provide a path for this.

The right-hand side of Figure 1 shows how future systems might interact with ModSAF using the same interface that users see through a simulated eye & hand designed to allow models to interact with synthetic environments (Ritter, Jones, Baxter, & Young, 1998a). The interface to the physical simulation could no doubt be made more regular and easier to use so that other architectures, such as Sim_Agent, could be hooked up to it. We suspect this project might take a good programmer familiar with ModSAF about half-time over a year because we have had a similar system built in two weeks by someone who was an expert in their graphical programming language. A much longer time should be allowed, however, because this system requires knowing ModSAF very well because it will make use of all of ModSAF and may require extending ModSAF.

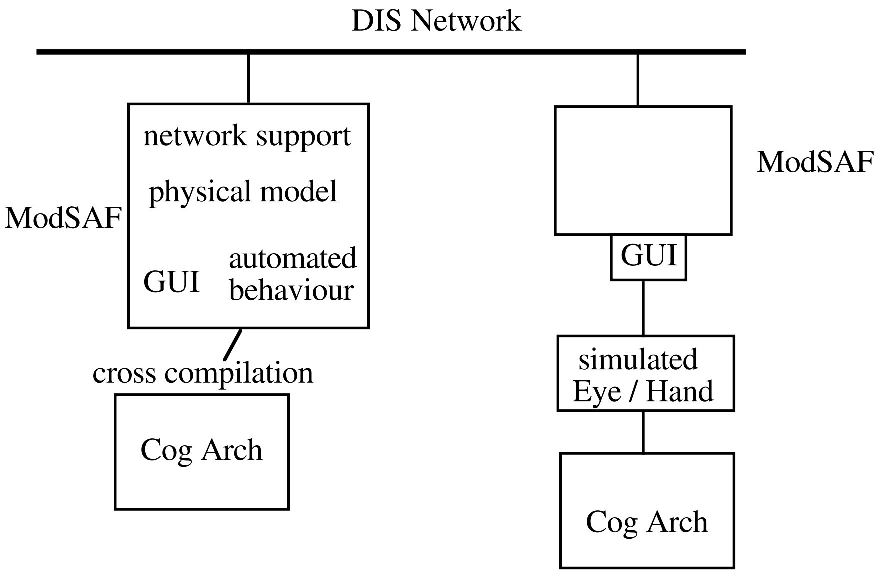

Figure 1. A functional description of Tac-Air Soar and how it uses ModSAF on the left and a perceptual interface to ModSAF on the right.

7.2.6 A framework for integrating knowledge

Currently, there are multiple knowledge sets in different simulations that exist in multiple formats. It would be useful to create a framework for integrating multiple knowledge sets, allowing the knowledge to be reused in different simulations.

One way to do this is to create a task editor that could take a knowledge set and compile it for different architectures. The editor would have to be based on a high-level description of knowledge, such as generic tasks (Wielinga, Schreiber, & Breuker, 1992). These generic tasks would then be compiled into things such as an ACT-R or Soar rule set.

There are potentially huge payoffs from this very high risk project. First, it would provide a way to reuse knowledge in multiple simulations. This reuse would help validate models, and might provide a way forward for validating architectures. Second, it would provide another way of documenting behavior models. The (presumably) graphic representation would allow others to browse and understand the model on a high level. Third, it would assist in writing models. In most cases, there are a lot of low-level details in creating these models that are not of theoretical interest, but require attention, such as using the same attribute name consistently. A high-level compiler for knowledge like this would bring with it all the advantages traditionally associated with high-level languages. When done with Soar models could be built two to three times faster (Yost, 1992).

PC-Pack (<www.epistemics.co.uk>) is a potential tool to start building upon. Implementing an initial, demonstration version of this approach would take a good programmer six to twelve months. Putting it to use would take longer.

7.2.7 Methods for comparing modeling approaches

We find ourselves in a position where a number of different approaches to simulating human behavior are on offer. Some of these approaches, at least, see themselves as rivals, and make competing claims about their suitability and quality. How can we set about assessing and comparing them?

There can, of course, be no one method that answers such a question. Earlier sections of this report have discussed how practical considerations such as usability and communicability come into play as well as scientific qualities such as agreement with data. Thus, almost any defensible comment about a model or architecture can be relevant to considering a choice between them.

However, there are some methods available that are too loose and varied to constitute a "technique" but are useful nonetheless for comparing and contrasting such differing approaches. They take the form of *matrix exercises*, in which a range of modeling approaches are pitted against a battery of concrete scenarios to be modeled. Young and Barnard (1987) provide the basic rationale for such a method, and explain how it can be used to judge the fit and scope of a modeling approach. They argue, first, that the modeling approaches need to be applied to concrete scenarios. It is not sufficient to try comparing approaches on the basis of their "features" or "characteristics". Second, it is important to use a *range* of scenarios. Taking just a single case will inevitably introduce a bias towards or against certain approaches, and will fail to provide an indication of their scope. Young, Barnard, Simon and Whittington (1989a) provide a short example of such a matrix exercise, and show how the entries in the matrix can be interpreted.

This kind of matrix exercise derives from the idea of a *bake-off* between rival approaches, but also differs in important respects. It must be recognized that there is unlikely to be a *winner*, one approach that is regarded as the best in all respects. Moreover, the matrix exercise is fundamentally co-operative rather than competitive. Instead of finding the *best* approach, it provides a tool for probing the scope of applicability of the different approaches, and investigating their relative strengths and weaknesses, advantages and disadvantages.

Some exercises of this kind have been performed in public. At the Research Symposia associated with the CHI conferences in 1993 and 1994, Young (in 1993) and Young and C. Lewis (in 1994) organized such matrix exercises on the design of an undo facility for a shared editor (1993), and on the analysis of the persistent "unselected window" error and of the design of an automated bank teller machine as a walk-up-and-use device (in 1994). Furthermore, there are precedents for such an exercise in a military research context. In 1993, NASA funded a comparative study of models of pilot checklist completion. ONR has funded, on a longer timescale, multiple analyses and modeling of several interactive tasks using hybrid architectures (Gigley & Chipman, 1999). The speech recognition community in the United States uses this approach in a quite competitive way as well. The US Air Force (contract officer Michael Young) has recently started a similar programme to explore models of complex behavior. Pew and Mavor appear to call for this kind of activity (1998, pp. 336-339), but without specifying so concrete a methodology.

A final but important point about such an exercise is that it cannot be done successfully on the cheap. The exercise requires earmarked and realistic funding in order to provide useful results. A considerable amount of work is required: first in negotiating, agreeing, and then specifying a set of concrete and clearly described scenarios, ideally with associated empirical data; and then subsequently for applying the modeling approaches to the scenarios, performing the comparisons, and drawing conclusions. Multiple research groups are used and the funding is leveraged by their existing work.

7.2.8 (Re)Implementing the battlefield simulation

There are strong arguments for implementing communicating agents and intra-agent processes in JAVA. These are discussed in Bigus and Bigus (1997), and in the context of JACK Intelligent Agents by Busetta et al. at <www.agentlink.org>. In fact, there are powerful arguments for building the entire synthetic agent simulation in JAVA as described below. This is possible within the Higher Level Architecture (HLA) framework. Implementing Soar in JAVA has also been mooted (Schwamb, 1998).

A core system implementation is needed that can then be accessed through Application Programming Interfaces (APIs). Supporting software is available for this, but any software could be developed for the purpose, provided it conformed with the standard. The core system might be written in JAVA or any other language provided only that the API is implemented. Similarly, entities may be written in any language, or several, provided that they set up calls to the API specification. There are a number of arguments for using JAVA as the basis for both individual entity simulation and for building a core system to the HLA specification. These are described below.

The single most attractive advantage of developing a synthetic battlefield simulation within a JAVA environment lies in the capabilities available within RMI that forms part of the JAVA run-time environment. This is a distributed object model with some similarities to Microsoft's DCOM, but with the advantage that it is effective on any platform that supports a JAVA run-time environment. It goes well beyond traditional RPC in being entirely object based, even allowing objects to be passed as parameters. This means that object behavior as well as data can be passed to a remote object in a seamless and transparent way. A mortar weapon being passed as an argument to an individual infantry man entity and arriving complete with its complement of munitions and ability to be fired gives a picture of this capability. The JAVA run-time environment also supports a naming and directory service API (JAVA JNDI) that allows the objects of RMI calls to be found. (For more details of this see <www.javasoft.com/products/jndi/index.html>.)

To show how such a service might be used, suppose that a simulation of an individual paratrooper has been developed. This simulation is a uniquely named JAVA object that can be invoked on any machine on the network used for the simulation. The JNDI service will inform a process of which machines have a suitable simulation available. To take this an important stage further, we use a class-factory object to produce the individual paratrooper objects. This class factory might use randomized parameters to make each entity distinct to fit a known distribution (like cabbage-patch dolls). To introduce these entities into the simulation a process would ask the naming service for a suitable class-factory object. This might be on one of any number of machines and is therefore extremely robust against damage to the network. The class factory can then be asked to produce any number of paratrooper entities each of which (in JAVA) is capable of serializing itself to any other machine on the network, and running there. Indeed the simulation can be moved from machine to machine at will, perhaps in response to a condition such as imminent power failure.

This approach would also support testing new platforms. A manufacturer might develop an improved simulation of a Tornado fighter-bomber. They then could introduce a new machine with a suitably registered class-factory object. Once this was connected to the network, the new simulation would be immediately available even if this were done while a simulation was running. No relinking, recompilation, or even pause in the simulation would be needed. The objects could be defined in conformity with the HLA standard.

JAVA also supports secure communications and has well-developed APIs for database connectivity and for driving graphics devices. An attractive user interface is very much easier to develop using the JAVA Foundation Classed (JFC) than, for example, using X-Motif. In addition, if a JIT (Just In Time) compiler is available to the RTE, programs developed in JAVA show little performance degradation in comparison with C++.

A synthetic environment could be developed using facilities offered by the JAVA run-time environment and existing APIs that would come much closer than existing simulations in meeting the design goals of maintainability, versatility, and robustness. This approach would have to be agreed with multiple communities, and would require a large amount of resources if it were applied uniformly.

7.3 Projects improving usability

This section reviews several possible projects for making model building more routine. For practical reasons, it is useful to make the process more routine. It is also important for theoretical reasons. If the models cannot be created within a time commensurate with gathering data, the majority of work will continue to be data gathering because theory development will be seen as too difficult.

7.3.1 Defining the modeling methodology

There is not yet a definitive approach or handbook for building models that can also be used for teaching and practicing modeling cognitive behavior. Newell and Simon's (1972) book is too long and mostly teaches by example. Ericsson and Simon's (1993) book on verbal protocol analysis has comments on how to create models. Although useful, the comments are short. VanSomeren, Barnard, and Sandberg (1994) provide a useful text, but it is slightly short and some of the details of going from model to data are not specified (if indeed they can be).

Baxter's (1997) report and Yost and Newell's (1989) article examples of the process are useful, but tied to a single architecture and not widely available. There are other useful papers worth noting, but they are short and do not provide the full story (Brazier et al., 1999; Ritter & Larkin, 1994; Sun & Ling, 1998).

Rouse (1980) has also made an attempt at describing the modeling process. He identifies the following steps as forming an important part of the modeling process: (a) definition, (b) representation, (c) calculation, (d) experimentation, (e) comparison, and (f) iteration. Rouse mainly focuses on the representation and calculation aspects of modeling, particularly from an engineering point of view. He describes several methodologies, including control theory, queuing theory, and rule-based production systems; he also provides a short tutorial on several of these modeling methods together with practical examples of systems engineering models. The examples are taken from a wide variety of domains including aviation, air traffic control, and industrial process control. It is not a complete treatise on human behavior, but does provide suggestions for methods that may be useful in modeling certain aspects of human behavior.

Similar tutorials and methodological summaries should be created until they converge. The results will be useful to practitioners and to those who are learning to model, which will be an important audience as this field grows. The output is most likely to require a textbook. A year to several years of support would significantly help create this.

7.3.2 Improvements to ModSAF

A major problem with ModSAF is usability. ModSAF is large and has a complicated syntax. Users report problems learning and using it. One way to improve its usability might be a better interface; better manuals and training aids might also be useful.

The approach used by models of behavior to interact with basic simulation capabilities such as ModSAF needs to be regularized. A fundamentally better approach might be possible. There exists an interface between ModSAF and Soar that partly provides a model eye and hand. This eye/hand could be improved to provide a more abstract interface to ModSAF, one that might be easier to use.

One thing we have repeatedly noted is that getting models to interact with simulations is more bearable when both are implemented within the same development environment. When they are not, work proceeds much more slowly (Ritter et al., 2000; Ritter & Major, 1995), requiring a mastery of both environments. The situation is exacerbated because the development and use of any communication facility tends to be an ill-defined problem with numerous wild subproblems (i.e., problems where the time to solution can be high and with a large variance, that is, not easily predicted). So, for example, although the ModSAF Tac-Air system (Tambe, Johnson, Jones, Koss, Laird, Rosenbloom, et al., 1995) appears as if it was developed using joint compilation techniques, it was probably difficult to use because it implements communication between ModSAF and the Tac-Air model using sockets. Although informal communication with researchers in the Soar and robotics communities suggest that the use of sockets may be becoming more routine, this has not always been the case.

7.3.3 Environments for model building and reuse

There remains a need for better environments for creating models. Few modeling interfaces provide much support for the user to program at the problem space level or even the knowledge level, although the Cogent interface is interesting as an example of usability, the Soar interface is providing increasing amounts of support at the symbol level. TAQL (Yost & Newell, 1989) and Able (Ritter et al., 1998b) have been moderately successful but modest attempts to create high level tools in Soar, for example.

Soar, in particular, needs a better interface. While there is now a modest interface, even the latest versions of the Soar interface (Kalus & Hirst, 1999; Laird, 1999; Ritter et al., 1998b) are not as advanced as many expert system shells and the previous, Lisp-based version (Ritter & Larkin, 1994). Gratch's (1998) planning level interface should be expanded and disseminated as a modeling interface. Knowledge acquisition tools and techniques (e.g., Cottam & Shadbolt, 1998; O'Hara & Shadbolt, 1998) might be particularly useful bases upon which to build.

Associated with this project would be general support for programming. This includes lists of frequently asked questions, tutorials, and generating models or model libraries designed for reuse. These libraries should either exist in each architecture, or in the general task language developed in the previous task. These would serve as a type of default knowledge for use in other applications. We can already envision libraries of interaction knowledge (about how to push buttons and search menus), arithmetic, and simple optimization like the default knowledge in Soar.

Work on improving the modeling interfaces for each architecture should be incorporated as part of another modeling project so that the developers of the interface have a ready-made audience. There are multiple additions that would be useful and multiple approaches that could be explored, so it could take almost any amount of resources, ranging from a month to several years.

7.3.4 Individual Data Modeling (IDM): An approach for validating models

What is the best way to make theoretical progress in the study of behavior in general and of cognition in particular? To develop micro-theories explaining a small domain or to aim at a higher goal, and develop an overarching theory covering a large number of domains -- a unified theory? Modern psychology, as a field, has tended to prefer micro-theories. It is true that unified theories have regularly appeared in psychology -- think of Piaget's (1954) or Skinner's (1957) theories -- but it is generally admitted that such unified theories have failed to offer a rigorous and testable picture of the human mind. Given this relatively unsuccessful history, it was with interest that cognitive science has observed Newell's (1990; see also Newell, 1973) call for a revival of unified theories in psychology.

One of the reasons for the limited success of Newell's own brand of UTC is that the methodology commonly used in psychology, based on controlling potentially confounding variables by using group data, is not the best way forward for developing UTCs. Instead, Gobet and Ritter (2000) propose an approach, which they call individual data modeling (IDM), where (a) the problems related to group averages are alleviated by analyzing subjects individually on a large set of tasks; (b) there is a close interaction between theory building and experimentation; and (c) computer technology is used to routinely test versions of the theory on a wide range of data. They claim that there are significant advantages here, that this approach will also help traditional approaches progress, and that the main potential disadvantage -- lack of generality -- may be taken care of by adequate testing procedures.

IDM offers several particular advantages in this area. It does not require as much data because the data will not be averaged but compared on a fine-grained level. This is attractive when the data is detailed or expensive to acquire, or where the model makes detailed predictions. The other advantage it offers is that it provides a model that produces more accurate behavior on a detailed level. It is this detailed level of behavior that will be necessary for not only appearing human in a Turing test, but leading to accurate training results because it performs like a comparable colleague or foe.

Work using this approach is ongoing at Nottingham and at Penn State. A full test would require one to two years of work to gather data and compare it with a model. This project could be combined with other projects, however, because it is a methodology and not a feature of behavior to include in a model.

7.3.5 Automatic model building

Most process models induced from protocols are created by hand. There has been some work to do this automatically or semi-automatically with machine learning techniques. Semi-automatic generation is done in the event structure modeling domain (a sociological level of social events) by a program called Ethno (Heise, 1989; Heise & Lewis, 1991). It iterates though a database of known events finding those without known precursors. It presents these to the modeller, querying for their precursors. As it runs it asks the modeller to create simple qualitative, non-variablized token matching rules representing the event's causal relationships based on social and scientific processes. The result at the end of an analysis is a rule set of ten to twenty rules that shape sociological behavior in that area. In a sense, the modeller is doing impasse driven programming (i.e., what is the next precursor for an uncovered event not provided by an already existing rule?). After this step, or in place of it, the modeller can compare the model's predictions with a series of actions on a sociological level (a protocol in the formal sense of the word). The tool notes which actions could follow, and queries the modeller based on this. Where mismatches occur, Ethno can present several possible fixes for configuration. Incorporating the model with the analysis tool in an integrated environment makes it more powerful. It would be a short extension to see the social events as cognitive events in a protocol.

Stronger methods for building models from a protocol are also available. Cirrus (VanLehn & Garlick, 1987) and ACM (Langley & Ohlsson, 1984) will induce decision trees for transitions between states that could be turned into production rules given a description of the problem space, including its elements, and the coded actions in the protocol. Cirrus and ACM use a variant of the ID3 learning algorithm (Quinlan, 1983). (ID3 induces rules that describe relationships in data.)

These tools look like a useful way to refine process models. Why is automatic creation of process models not done more often? Perhaps it is because these tools do not create complete process models. They take a generalized version of an operator that must be specified as part of a process model. It could also be that finding the conditions of operators is not the difficult problem, but that creating the initial process model and operators is. It could also be that it is harder to write process models that can be used by these machine learning algorithms. In any case, these methods should be explored further.

Diligent (Angros, 1998), Instructo-Soar (Huffman & Laird, 1995), and Observo-Soar (van Lent, 1999) are approaches to create models in Soar that learn how to perform new tasks by observing behavior and inferring problem solving steps to duplicate it. Related models have been used in synthetic environments (Assanie & Laird, 1999; van Lent & Laird, 1999). They have had limited use, but suggest that learning through observation may be a quite important way to create models as an important way that humans learn. Their lack of use could simply be due to that they are novel software systems. As novel systems they are probably difficult for people other than their developers to use, and they will have to go through several iterations of improvement (like most pieces of software) before they are ready for outsiders. With a small user base (so far), the need has not forced the software development, which has further decreased their potential audience.

Automatic modeling tools need to be developed. Machine learning algorithms and theories of cognition are developed enough that this could be a very fruitful approach. A several-year effort here could yield large benefits of more routine modeling.

7.3.6 Using genetic algorithms to fit data

There are two potential uses of genetic algorithms worth highlighting. The first is for generating behavior as described above in Section 5.2.

The second use of genetic algorithms is for optimizing model fits by adjusting their parameters (Davis & Ritter, 1987; Ritter, 1991). Most model fits have been optimized by hand, which leads to absolute and relative performance problems. In absolute terms, researchers may not be getting optimal performance from their models. In relative terms, comparisons of or hand- optimized models may not be fair. (Sometimes even one model is optimized and the other not.) In the case of models with multiple parameters (with submodels to include), this job is not tractable by hand.

The results obtained by optimizing models with genetic algorithms suggest that optimizations done by hand are likely to be inferior to those done by genetic algorithms (Ritter, 1991) or by other machine learning techniques (Butler, 2000). Use of genetic algorithms (or similar techniques) would improve performance in absolute terms, provide fairer comparisons between models, and encourage the inclusion of parameter set behavior in model comparisons. Several years of a PhD student working within a project with a model to optimize is probably a good way to progress work in this area.

This optimization should initially be done with an existing model so that the developers of the interface have a ready-made model and audience. The basic approach is simple and robust, and should be straightforward to demonstrate. Making it routine and portable are separate steps and more advanced steps, so it could take almost any amount of resources, ranging from a month to several years.

7.4 Other applications of behavioral models in synthetic environments

The are numerous ways that behavioral models could be applied outside the military domain. We will examine just four types of them here.

The most obvious additional application of the models arising from approaches proposed in this report is in the provision of automated support for system operators. This support can take the form of intelligent decision support systems or embedded assistants that guide operator behavior. There are some existing applications, most notably the Pilot's Associate (Geddes, 1989), its derivative Hazard Monitor (Greenberg, Small, Zenyah, & Skidmore, 1995), and CASSY (Wittig & Onken, 1992), all from the aviation domain. In the UK, the Future Organic Airborne early warning system is attempting to insert a knowledge-based system into the Osprey aircraft and radar simulation to assist users.

These assistants, because they have a model of what the user is likely to do next, should be able to assist the user: if not by performing the task, then by preparing materials or information, or by modifying the display to help distinguish between alternatives or make performing actions easier. In the past, such assistants have had only a limited ability to model users. With increased validity and accuracy, these models may become truly useful.

The second application is in education and training. The uses in education have been fairly well illustrated by Anderson's work with cognitive model-based tutors (Anderson, Corbett, Koedinger, & Pelletier, 1995). In training, behavioral models can be used to provide experts to emulate (Ritter & Feurzeig, 1988). The same knowledge can also be used to debrief students' performances. The knowledge can also be used to populate adversaries and colleagues in the same environment (Bloedorn & Downes-Martin, 1985).

Training needs exist outside the military in several domains where dynamic models are necessary. Mining, for example, is starting to use virtual reality to train simple tasks (Hollands, Denby, & Brooks, in press). Virtual reality is already being used to train hazard spotting, avoiding mine machinery as a pedestrian, and driving vehicles underground (Schofield & Denby, 1995).

The third application is in entertainment. This has been proposed for some time as an application. The recent report by the (US) National Research Council on this topic (Computer Science and Telecommunications Board, 1997) suggests that not only is it possible to use synthetic environments and the behavioral models in them for entertainment, but it is currently being done by the Center for Interactive Technologies at the University of Southern California.

The fourth application is in system analysis. The behavioral models can be used to examine different system designs to measure errors, processing rates, or emergent strategies. To return to mining again, truck models in a simulation can be used to examine road layouts in mines (Williams, Schofield, & Denby, 1998).

7.5 Summary of projects

This report has laid out important objectives for models of behavior in synthetic environments in the important areas of providing more complete performance, increased integration of the models, and improved usability of the models, It should be noted that funding bodies may be interested in supporting some of this work because most of these projects will not simply improve engineering models of human behavior, but will also improve our understanding of behavior and our general scientific ability to predict and model human behavior in general.

These proposals, taken as a whole, call for several general research programmes. They suggest several moderating variables that affect cognition, including emotions, personality, and interactions with the environment, which should be included in cognitive architectures. They argue for creating or moving towards a more uniform format for data and models and a more clearly defined approach for modeling. There are also several concrete suggestions for making this approach easier and more routine, including providing more usable modeling environments and supporting automatic model generation. Finally, we were able to suggest some further applications of models of behavior in synthetic environments.

References

Aasman, J. (1995). *Modelling driver behaviour in Soar*. Leidschendam, The Netherlands: KPN Research.

Aasman, J., & Michon, J. A. (1992). Multitasking in driving. In J. A. Michon & A. Akyürek (Eds.), *Soar: A cognitive architecture in perspective.* Dordrecht, The Netherlands: Kluwer.

Agre, P. E., & Shrager, J. (1990). Routine evolution as the microgenetic basis of skill acquisition. In *Proceedings of the 12th Annual Conference of the Cognitive Science Society*. 694-701. Hillsdale, NJ: Lawrence Erlbaum.

Albus, J. S. (1971). A theory of cerebella function. *Mathematical BioScience, 10*, 25-61.

Alechina, N., & Logan, B. (1998). State space search with prioritised soft constraints. In *Proceedings of the {ECAI-98} Workshop 'Decision theory meets artificial intelligence: qualitative and quantitative approaches'*. 33-42. ECCAI.

Amalberti, R., & Deblon, F. (1992). Cognitive modelling of fighter aircraft's control process: A step towards intelligent onboard assistance system. *Int. Journal of Man-Machine Studies, 36*, 639-671.

Anderson, J. R. (1983). *The architecture of cognition*. Cambridge, MA: Harvard University Press.

Anderson, J. R. (1990). *The adaptive character of thought*. Hillsdale, NJ: Lawrence Erlbaum.

Anderson, J. R. (1993). *Rules of the mind.* Hillsdale, NJ: Lawrence Erlbaum.

Anderson, J. R., Corbett, A. T., Koedinger, K. R., & Pelletier, R. (1995). Cognitive tutors: Lessons learned. *Journal of the Learning Sciences, 4*(2), 167-207.

Anderson, J. R., Farrell, R., & Sauers, R. (1984). Learning to program in LISP. *Cognitive Science, 8*, 87-129.

Anderson, J. R., & Lebiere, C. (1998). *The atomic components of thought*. Mahwah, NJ: Lawrence Erlbaum.

Anderson, J. R., Matessa, M., & Lebiere, C. (1998). ACT-R: A theory of higher level cognition and its relation to visual attention. *Human-Computer Interaction, 12*, 439-462.

Angros, R. (1998). Learning what to instruct. In *Proceedings of Soar Workshop 18*. 58-64. Vienna, VA: Explore Reasoning Systems.

Anzai, Y., & Simon, H. A. (1979). The theory of learning by doing. *Psychological Review, 86*, 124-140.

Assanie, M., & Laird, J. (1999). Learning by instruction using a constrained natural language interface. In *Proceedings of Soar Workshop 19*. 144-149. U. of Michigan Soar Group.

Baddeley, A. D. (1986). *Working memory*. Oxford, UK: Oxford University Press.

Baddeley, A. D. (1997). *Human memory: Theory and practice*. Hove, UK: Psychology Press.

Bai, L., Burke, E. K., Elliman, D. G., & Ford, P. H. (1993). Constraint propagation over ordered domains. In *Proceedings of the 8th IEEE Conference on Artificial Intelligence for Applications*. 35-43. New York, NY: IEEE Press.

Bartl, C., & Dörner, D. (1998). PSI: A theory of the integration of cognition, emotion and motivation. In F. E. Ritter & R. M. Young (Eds.), *Proceedings of the 2nd European Conference on Cognitive Modelling*. 66-73. Thrumpton, Nottingham, UK: Nottingham University Press.

Bass, E. J., Baxter, G. D., & Ritter, F. E. (1995). Creating cognitive models to control simulations of complex systems. *AISB Quarterly, 93*, 18-25.

Baxter, G. D. (1997). From soup to nuts: Developing a Soar cognitive model of the electronic warfare officer's task (Working paper No. WP/R3BAIA005/014). Cognitive Modelling unit. Psychology Department, U. of Nottingham.

Baxter, G. D., & Ritter, F. E. (1996). Designing abstract visual perceptual and motor action capabilities for use by cognitive models (Tech. Report No. 36). ESRC CREDIT, Psychology, U. of Nottingham.

Baylor, G. W., & Simon, H. A. (1966). A chess mating combinations program. In *Proceedings of the 1966 Spring Joint Computer Conference. 28*. 431-447. Boston. Reprinted in H. A. Simon (Ed.), (1979) Models of thought. New Haven: Yale University Press.

Belavkin, R. (2001). The role of emotions in problem solving. In *AISB'01 Symposium on Emotion, Cognition and Affective Computing*. 49-57. Falmer, UK: The Society for the Study of AI and Simulation of Behaviour.

Belavkin, R., & Ritter, F. E. (2000). Adding a theory of motivation to ACT-R. In *Proceedings of the Seventh Annual ACT-R Workshop*. 133-139. Department of Psychology, Carnegie-Mellon University: Pittsburgh, PA.

Belavkin, R. V., Ritter, F. E., & Elliman, D. G. (1999). Towards including simple emotions in a cognitive architecture in order to fit children's behaviour better. In *Proceedings of the 1999 Conference of the Cognitive Science Society*. 784. Mahwah, NJ: Lawrence Erlbaum.

Bigus, J. P., & Bigus, J. (1997). *Constructing intelligent agents in Java*. New York, NY: John Wiley & Sons.

Bloedorn, G. W., & Downes-Martin, S. G. (1985). Tank tactics grandmaster (Report No. 5938). BBN Laboratories, Cambridge, MA.

Boff, K. R., & Lincoln, J. E. (1986). *Engineering data compendium: Human perception and performance*. Wright-Patterson Air Force Base, OH: John Wiley & Sons.

Brazier, F., Dunin-Keplicz, B., Treur, J., & Verbrugge, R. (1999). Modelling internal dynamic behaviour of BDI agents. In A. Cesto & P. Y. Schobbes (Eds.), *Proceedings of the Third International Workshop on Formal Methods of Agents, MODELAGE '97*. 21. Lecture notes in AI. Berlin: Springer Verlag.

Brooks, R. (1992). Intelligence without representation. In D. Kirsh (Ed.), *Foundations of artificial intelligence*. Cambridge, MA: MIT Press.

Burke, E. K., Elliman, D. G., & Weare, R. F. (1995). A hybrid genetic algorithm for highly constrained timetabling problems. In *Proceedings of the 6th International Conference on Genetic Algorithms*. 605-610. Morgan Kaufman.

Busetta, P., Howden, N., Rönnquist, R., & Hodgson, A. (1999a). Structuring BDI agents in functional clusters. In *Proceedings of the Sixth International Workshop on Agent Technologies, Architectures and Languages*. 149-161.

Busetta, P., Rönnquist, R., Hodgson, A., & Lucas, A. (1999b). JACK intelligent agents - Components for intelligent agents in JAVA. *AgentLink News Letter, 2*(Jan.), <http://www.agent-software.com/white-paper.pdf>.

Butler, E. (2000).

Byrne, M. (in press). ACT-R/PM and menu selection: Applying a cognitive architecture to HCI. *International Journal of Human-Computer Studies*.

Byrne, M. D. (1997). ACT-R Perceptual-Motor (ACT-R/PM) version 1.0b1: A users manual. Pittsburgh, PA: Psychology Department, Carnegie-Mellon University. Available at <act.psy.cmu.edu>.

Byrne, M. D., & Anderson, J. R. (1998). Perception and action. In J. R. Anderson & C. Lebière (Eds.), *The atomic components of thought.* Mahwah, NJ: Lawrence Erlbaum.

Byrne, M. D., & Bovair, S. (1997). A working memory model of a common procedural error. *Cognitive Science, 21*(1), 31-61.

Byrne, M. D., Chong, R., Freed, M., Ritter, F. E., & Gray, W. (1999). Symposium on Integrated models of perception, cognition, and action. In *Proceedings of the 1999 Conference of the Cognitive Science Society*. 1. Mahwah, NJ: Lawrence Erlbaum.

Cacciabue, P. C., Decortis, F., Drozdowicz, B., Masson, M., & Nordvik, J. (1992). COSIMO: A cognitive simulation model of human decision making and behavior in accident management of complex plants. *IEEE Transactions on Systems, Man, and Cybernetics, 22*(5), 1058-1074.

Campbell, C., Hull, R., Root, E., & Jackson, L. (1995). Route Planning in CCTT. In *Proceedings of the Fifth Conference on Computer Generated Forces and Behavioural Representation*. 233-244. Institute for Simulation and Training.

Ceranowicz, A. (1998). STOW 97 - 99. In *Proceedings of the 7th Conference on Computer Generated Forces and Behavioural Representation*. 3-17.

Chandrasakaren, B., & Josephson, J. R. (1999). Cognitive modeling for simulation goals: A research strategy for computer-generated forces. In *Proceedings of the 7th Computer Generated Forces and Behavioural Representation Conference*. 239-250.

Charness, N. (1991). Expertise in chess: The balance between knowledge and search. In K. A. Ericsson & J. Smith (Eds.), *Studies of expertise : Prospects and limits.* Cambridge (UK): Cambridge University Press.

Charness, N. (1992). The impact of chess research on cognitive science. *Psychological Research, 54*, 4-9.

Chase, W. G., & Simon, H. A. (1973). Perception in chess. *Cognitive Psychology, 4*, 55-81.

Chipman, S., & Meyrowitz, A. L. (Eds.). (1993). *Foundations of knowledge acquisition: Cognitive models of complex learning*. Boston, MA: Kluwer.

Chong, R. (1999). Towards a model of fear in Soar. In *Proceedings of Soar Workshop 19*. 6-9. U. of Michigan Soar Group.

Chong, R. S., & Laird, J. E. (1997). Identifying dual-task executive process knowledge using EPIC-Soar. In *Proceedings of the 19th Annual Conference of the Cognitive Science Society*. 107-112. Mahwah, NJ: Lawrence Erlbaum.

Computer Science and Telecommunications Board, N. R. C. (1997). *Modeling and simulation: Linking entertainment and defense*. Washington, DC: National Academy Press.

Cooper, R., & Fox, J. (1998). COGENT: A visual design environment for cognitive modelling. *Behavior Research Methods, Instruments and Computers, 30*, 553-564.

Cottam, H., & Shadbolt, N. R. (1998). Knowledge acquisition for search and rescue planning. *International Journal of Human-Computer Studies, 48*, 449-473.

Crow, L., & Shadbolt, N. R. (1998). IMPS - Internet agents for knowledge engineering. In B. R. Gaines & M. Musen (Eds.), *Knowledge Acquisition Workshop KAW'98*. 1-19. University of Calgary, Banff, Alberta: SRAG Publications.

Crowder, R. G. (1976). *Principles of learning and memory*. Hillsdale, NJ: Lawrence Erlbaum.

Damasio, A. R. (1994). *Descartes' error: Emotion, reason, and the human brain*. New York, NY: Gosset/Putnam Press.

Daneman, M., & Carpenter, P. A. (1980). Individual differences in working memory and reading. *Journal of Verbal Learning and Verbal Behavior, 19*, 450-466.

Davis, L. W., & Ritter, F. (1987). Schedule optimization with probabilistic search. In *The Third IEEE Computer Society Conference on Artificial Intelligence Applications*. 231-236. Washington, DC: IEEE Press.

Dawes, R. M. (1988). *Rational choice in an uncertain world*. Orlando, FL: Harcourt Brace Jovanovich.

Dawes, R. M. (1994). *House of cards: Psychology and psychotherapy built on myth*. New York, NY: The Free Press.

de Groot, A. D. (1946). *Het denken van den schaker (Thought and choice in chess).* Amsterdam: Noord Hollandsche.

de Groot, A. D. (1978). *Thought and choice in chess*. The Hague: Mouton Publishers.

de Groot, A. D., & Gobet, F. (1996). *Perception and memory in chess*. Assen, NL: Van Gorcum.

de Keyser, V., & Woods, D. D. (1990). Fixation errors: Failures to revise situation assessment in dynamic and risky systems. In A. G. Colombo & A. S. de Bustamante (Eds.), *Systems reliability assessment*. 231-251. Dordrecht, The Netherlands: Kluwer Academic.

Delaney, P. F., Reder, L. M., Staszewski, J. J., & Ritter, F. E. (1998). The strategy specific nature of improvement: The power law applies by strategy within task. *Psychological Science, 9*(1), 1-8.

Detje, F. (2000). Comparison of the PSI-theory with human behaviour in a complex task. In N. Taatgen & J. Aasman (Eds.), *Proceedings of the Third International Conference on Cognitive Modelling*. 86-93. KS Veenendaal: Universal Press.

Elkind, J. I., Card, S. K., Hochberg, J., & Huey, B. M. (Eds.). (1989). *Human performance models for computer-aided engineering*. Washington, DC: National Academy Press.

Elliman, D. G. (1989). Machine recognition of engineering drawings. In G. Bryan (Ed.), *Exploitable UK research for manufacturing industry.* London: Peter Peregrinus Press.

Elman, J. L. (1991). Distributed representations, simple recurrent networks, and grammatical structure. *Machine Learning, 7*, 194-220.

Ericsson, K. A., & Kintsch, W. (1995). Long-term working memory. *Psychological Review, 102*, 211-245.

Ericsson, K. A., & Simon, H. A. (1993). *Protocol analysis: Verbal reports as data*. Cambridge, MA: The MIT Press.

Etzioni, O., & Weld, D. S. (1995). Intelligent agents on the Internet - Fact, fiction, and forecast. *IEEE Expert - Intelligent Systems & Their Applications, 10*(4), 44-49.

Falasconi, S., Lazola, G., & Stefanelli, M. (1999). Using ontologies in multi-agent systems. In A. Cesto & P. Y. Schobbes (Eds.), *Proceedings of the Third International Workshop on Formal Methods of Agents, MODELAGE '97.* 21. Lecture notes in AI. Berlin: Springer Verlag.

Feigenbaum, E. A., & Simon, H. A. (1962). A theory of the serial position effect. *British Journal of Psychology, 53*, 307-320.

Feigenbaum, E. A., & Simon, H. A. (1984). EPAM-like models of recognition and learning. *Cognitive Science, 8*, 305-336.

Franklin, S., & Graesser, A. (1997). Is it an agent, or just a program?: A taxonomy for autonomous agents. In J. P. Müller, M. J. Wooldridge, & N. R. Jennings (Eds.), *Intelligent Agents III -- Proceedings of the Third International Workshop on Agent Theories, Architectures, and Languages (ATAL-96), Lecture Notes in Artificial Intelligence.* 21-36. Berlin: Springer-Verlag.

Freed, M., & Remington, R. (2000). Making human-machine system simulation a practical engineering tool: An APEX overview. In N. Taatgen & J. Aasman (Eds.), *Proceedings of the 3rd International Conference on Cognitive Modelling.* 110-117. Veenendaal (NL): Universal Press.

Freed, M. A., Shafto, M. G., & Remington, R. W. (1998). Employing simulation to evaluate designs: The APEX approach. In *Proceedings of the 2nd Workshop on Human Error, Safety, and System Development.* 120-132.

Frese, M., & Altmann, A. (1989). The treatment of errors in training and learning. In L. Bainbridge & S. A. R. Quintanilla (Eds.), *Developing skills with information technology.* 65-86. Chichester, UK: Wiley.

Geddes, N. D. (1989). *Understanding human intentions through action interpretation.* Unpublished PhD thesis, Georgia Institute of Technology.

Gigley, H. M., & Chipman, S. F. (1999). Productive interdisciplinarity: The challenge that human learning poses to machine learning. In *Proceedings of the 21st Conference of the Cognitive Science Society.* 2. Mahwah, NJ: Lawrence Erlbaum.

Glover, F., & Laguna, M. (1998). *Tabu search.* Dordrecht, NL: Kluwer Academic Publishers.

Gobet, F. (1997). A pattern-recognition theory of search in expert problem solving. *Thinking and Reasoning, 3*, 291-313.

Gobet, F. (1998). Expert memory: A comparison of four theories. *Cognition, 66*, 115-152.

Gobet, F., & Jansen, P. (1994). Towards a chess program based on a model of human memory. In H. J. van den Herik, I. S. Herschberg, & J. E. Uiterwijk (Eds.), *Advances in computer chess 7.* Maastricht, NL: University of Limburg Press.

Gobet, F., Richman, H., Staszewski, J., & Simon, H. A. (1997). Goals, representations, and strategies in a concept attainment task: The EPAM model. *The Psychology of Learning and Motivation, 37*, 265-290.

Gobet, F., & Ritter, F. E. (2000). Individual Data Analysis and Unified Theories of Cognition: A methodological proposal. In N. Taatgen & J. Aasman (Eds.), *Proceedings of the 3rd International Conference on Cognitive Modelling.* 150-157. Veenendaal (NL): Universal Press.

Gobet, F., & Simon, H. (1996a). Templates in chess memory: A mechanism for recalling several boards. *Cognitive Psychology, 31*, 1-40.

Gobet, F., & Simon, H. A. (1996b). The roles of recognition processes and look-ahead search in time-constrained expert problem solving: Evidence from grandmaster level chess. *Psychological Science, 7*, 52-55.

Gobet, F., & Simon, H. A. (2000). Five seconds or sixty? Presentation time in expert memory. *Cognitive Science, 24*, 651-682.

Gobet, F., & Simon, H. A. (in press). Human learning in game playing. In M. Kubat, M. Fürnkranz, & J. Fürnkranz (Eds.), *Machines that learn to play games. NOVA science, Advances in Computation: Theory and Practice.*

Gratch, J. (1998). Planning in Soar. In *Proceedings of Soar Workshop 18.* 17-25. Explore Reasoning Systems.

Gratch, J. (1999). Why you should buy an emotional planner. In *Proceedings of Soar Workshop 19.* 1-3. U. of Michigan Soar Group.

Green, R. (1990). Human error on the flight deck. *Phil. Trans. Royal Society London, Series B, 327*, 503-512.

Greenberg, A. D., Small, R. L., Zenyah, J. P., & Skidmore, M. D. (1995). Monitoring for hazard in flight managment systems. *European Journal of Operations Research, 84*, 5-24.

Grimes, C., Picton, P. D., & Elliman, D. G. (1996). A neural network position independent multiple pattern recogniser. *Artificial Intelligence in Engineering, 10*, 117-126.

Hart, P. E., Nilsson, N. J., & Raphael, B. (1968). A formal basis for the heuristic determination of minimum cost paths. *IEEE Transactions on Systems Science and Cybernetics, SSC-4*(2), 100-107.

Heise, D. R. (1989). Modeling event structures. *Journal of Mathematical Sociology, 14*(2-3), 139-169.

Heise, D. R., & Lewis, E. (1991). *Introduction to ETHNO*. Dubuque, IA: William. C. Brown.

Hill, R. (1999). Modeling perceptual attention in virtual humans. In *Proceedings of the 7th Conference on Computer Generated Forces and Behavioural Representation.* 3-17.

Hoffman, R., & Shadbolt, N. R. (1995). A review of "Naturalistic Decision-Making" Research on the Critical Decision Method of knowledge elicitation and the Recognition Priming Model of Decision-Making. Report commisioned by Defence Research Agency, Farnborough, Project No ASF/2188/U. 329 pages.

Hoffman, R., & Shadbolt, N. R. (1996). Facilitating the acquisition of expertise in domains involving perceptual skill, mental workload, and situation awareness. Report commissioned by Defence Research Agency, Farnborough, Project No ASF/2819U. 375 pages RPD2.

Hoffman, R. R. (1987). The problem of extracting the knowledge of experts from the perspective of experimental psychology. *AI Magazine, 8*, 53-67.

Hoffman, R. R. (1992). *The psychology of expertise. Cognitive research and empirical AI.* New York: Springer-Verlag.

Hoffman, R. R., Crandall, B., & Shadbolt, N. R. (1998). Use of the Critical Decision Method to elicit expert knowledge: A case study in the methodology of Cognitive Task Analysis. *Human Factors, 40*, 254-276.

Holland, J. H., Holyoak, K., Nisbett, R., & Thagard, P. (1986). *Induction: Processes of inference, learning, and discovery.* Cambridge, MA: MIT Press.

Hollands, R., Denby, B., & Brooks, G. (in press). SAFE-VR - A virtual reality safety training system. *Proceedings of APCOM '99.*

Hollnagel, E. (1993). *Human reliability analysis: Context and control.* London: Academic Press.

Hollnagel, E. (1998). *Cognitive reliability and error assessment method.* Oxford, UK: Elsevier Science.

Hollnagel, E., & Cacciabue, C. (1991). Cognitive modelling in system simulation. Proceedings of the Third European Conference on Cognitive Science Approaches to Process Control. 1-29. Cardiff, UK, 2-6 September, 1991.

Howes, A. (1993). Recognition-based problem solving. In *Proceedings of the Fifteenth Annual Meeting of the Cognitive Science Society.* 551-556. Hillsdale, NJ: Lawrence Earlbaum.

Howes, A., & Young, R. M. (1996). Learning consistent, interactive, and meaningful task-action mappings: A computational model. *Cognitive Science, 20*(3), 301-356.

Hudlicka, E. (1997). Modeling behavior moderators in military human performance models (Technical Report No. 9716). Psychometrix. Lincoln, MA.

Hudlicka, E., & Fellous, J.-M. (1996). Review of computational models of emotion (Technical Report No. 9612). Psychometrix. Arlington, MA.

Huffman, S. B., & Laird, J. E. (1995). Flexibly instructable agents. *J. of AI Research, 3*, 271-324.

Jacobs, N., & Shea, R. (1996). The role of Java in InfoSleuth: Agent-based exploitation of heterogeneous information resources No. MCC-INSL-018-96. Presented at the IntraNet96 Java Developers Conference. Available online at

<http://www.mcc.com/projects/infosleuth/publications/intranet-java.html>.). MCC Technical Report.

Jansen, P. J. (1992). *Using knowledge about the opponent in game-tree search.* Unpublished Ph. D. thesis CMU-CS-92-192, Carnegie-Mellon University, PA.

John, B. E., & Kieras, D. E. (1996). Using GOMS for user interface design and evaluation: Which technique? *ACM Transactions on Computer-Human Interaction, 3*(4), 287-319.

John, B. E., Vera, A. H., & Newell, A. (1994). Towards real-time GOMS: A model of expert behavior in a highly interactive task. *Behavior and Information Technology, 13*, 255-267.

Jones, G. (1999). *Testing mechanisms of development within a computational framework.* Unpublished Ph. D. thesis, University of Nottingham.

Jones, G., & Ritter, F. E. (1998). Initial explorations of simulating cognitive and perceptual development by modifying architectures. In *Proceedings of the 20th Annual Conference of the Cognitive Science Society*. 543-548. Mahwah, NJ: Lawrence Erlbaum.

Jones, G., Ritter, F. E., & Wood, D. J. (2000). Using a cognitive architecture to examine what develops. *Psychological Science, 11*(2), 93-100.

Jones, R. (1998). Modeling pilot fatigue with a synthetic behavior model. In *Proceedings of the 7th Conference on Computer Generated Forces and Behavioural Representation*. 349-357.

Jones, R. M., Laird, J. E., Nielsen, P. E., Coulter, K. J., Kenny, P., & Koss, F. V. (1999). Automated intelligent pilots for combat flight simulation. *AI Magazine, 20*(1), 27-41.

Just, M. A., & Carpenter, P. A. (1992). A capacity theory of comprehension: Individual differences in working memory. *Psychological Review, 99*, 122-149.

Kalus, T., & Hirst, T. (1999). ViSoar. In *Proceedings of Soar Workshop 19*. 70-73. www.dcs.port.ac.uk/~hirsta/visoarx.htm. U. of Michigan Soar Group.

Kanerva, P. (1988). *Sparse distributed memory*. Cambridge, MA: MIT Press.

Kieras, D. E., & Meyer, D. E. (1997). An overview of the EPIC architecture for cognition and performance with application to human-computer interaction. *Human-Computer Interaction, 12*, 391-438.

Kitajima, M., & Polson, P. G. (1996). A comprehension-based model of exploration. In M. J. Tauber, V. Bellotti, R. Jeffries, J. D. MacKinlay, & J. Nielsen (Eds.), *Proceedings of the CHI '96 Conference on Human Factors in Computer Systems*. 324-331. New York, NY: ACM.

Kitajima, M., Soto, R., & Polson, P. G. (1998). LICAI+: A comprehension-based model of the recall of action sequences. In F. E. Ritter & R. M. Young (Eds.), *Proceedings of the 2nd European Conference on Cognitive Modelling*. 82-89. Thrumpton, Nottingham: Nottingham University Press.

Klein, G. A. (1997). *Recognition-primed decision making: Looking back, looking forward.* Hillsdale, NJ: Lawrence Erlbaum.

Koriat, A., & Lieblich, I. (1974). What does a person in a 'TOT' state know that a person in a 'don't know' state doesn't know? *Memory & Cognition, 2*(4), 647-655.

Kosslyn, S. M., & Koenig, O. (1992). *Wet mind*. New York, NY: The Free Press.

Kuk, G., Arnold, M., & Ritter, F. E. (1999). Using event history analysis to model the impact of workload on an air traffic tactical controller's operations. *Ergonomics, 42*(9), 1133-1148.

Laird, J. (1999). Visual Soar. In *Proceedings of Soar Workshop 19.* 99-102. U. of Michigan Soar Group.

Laird, J., & Duchi, J. C. (2000). Creating human-like synthetic characters with multiple skill levels: A case study using the Soar Quakebot. In *Simulating Human Agents, Papers from the 2000 AAAI Fall Symposium.* 75-79. Menlo Park, CA: AAAI Press.

Laird, J. E., Coulter, K., Jones, R., Kenny, P., Koss, F., & Nielsen, P. (1997). Review of Soar/FWA participation in STOW-97. Published electronically at <http://ai.eecs.umich.edu/ifor/stow-review.html>

Lane, P. C. R., Cheng, P. C.-H., & Gobet, F. (1999). Problem solving with diagrams: Modelling the implicit learning of perceptual information (Tech. Report No. 59). University of Nottingham: Department of Psychology, ESRC Centre for Research in Development, Instruction and Training.

Langley, P., & Ohlsson, S. (1984). Automated cognitive modeling. In *AAAI-84.* 193-197. Los Altos, CA: Morgan Kaufman.

Larkin, J. H. (1981). Enriching formal knowledge: A model for learning to solve textbook physics problems. In J. R. Anderson (Ed.), *Cognitive skills and their acquisition.* 311-334. Hillsdale, NJ: Lawrence Erlbaum.

Larkin, J. H., McDermott, J., Simon, D. P., & Simon, H. A. (1980). Expert and novice performance in solving physics problems. *Science, 208*, 1335-1342.

Lebière, C., Anderson, J. R., & Reder, L. M. (1994). Error modeling in the ACT-R production system. In *Proceedings of the 16th Annual Conference of the Cognitive Science Society.* 555-559.

Lesgold, A. M., Lajoie, S., Bunzon, M., & Eggan, E. (1992). SHERLOCK: A coached practice environment for an electronics troubleshooting job. In J. Larkin, R. Chabay, & C. Scheftic (Eds.), *Computer assisted instruction and intelligent tutoring systems: Establishing communication and collaboration.* Hillsdale, NJ: Lawrence Erlbaum.

Lewis, R. L. (1993). *An architecturally-based theory of human sentence comprehension.* Ph. D. thesis. CMU-CS-93-226, Carnegie-Mellon University, Pittsburgh, PA.

Logan, B. (1998). Classifying agent systems. In J. Baxter & B. Logan (Eds.), *Software Tools for Developing Agents: Papers from the 1998 Workshop*. Technical Report WS-98-10 11-21. Menlo Park, CA: AAAI Press. Also available at

http://www.cs.nott.ac.uk/~bsl/aaai-98/papers/logan.pdf.

Logan, B., & Alechina, N. (1998). A* with bounded costs. In *Proceedings of the Fifteenth National Conference on Artificial Intelligence (AAAI-98).* 444-449. Menlo Park CA & Cambridge MA: AAAI Press/MIT Press.

Logan, B., & Sloman, A. (1998). Qualitative decision support using prioritised soft constraints. Technical Report CSRP-98-14. School of Computer Science, University of Birmingham.

Lonsdale, P. R., & Ritter, F. E. (2000). Soar/Tcl-PM: Extending the Soar architecture to include a widely applicable virtual eye and hand. In N. Taatgen & J. Aasman (Eds.), *Proceedings of the 3rd International Conference on Cognitive Modelling*. 202-209. Veenendaal (NL): Universal Press.

Loral (1995). ModSAF software architecture design and overview document (SADOD) No. Report ADST/WDL/TR--95--W003339B). Prepared for U.S. Army Simulation, Training and Instrumention Command (STRICOM).

Lovett, M. C., Reder, L. M., & Lebiere, C. (1999). Modeling working memory in a unified architecture. In A. Miyake & P. Shah (Eds.), *Models of working memory: Mechanisms of active maintenance and executive control.* 135-182. New York, NY: Cambridge University Press.

Lucas, A., & Goss, S. (1999). The potential for intelligent software agents in defence simulation. In *IDC-99: Information, Decision and Control Adelaide, Australia, 8 - 10 February 1999.* .

Mach, E. (1905). *Knowledge and error*. Dordrecht, Netherlands: (English translation, D. Reidel, 1976).

McClelland, J. L., & Rumelhart, D. E. (1988). *Explorations in parallel distributed processing: A handbook of models, programs, and exercises*. Cambridge, MA: MIT Press.

Miller, C. S., & Laird, J. E. (1996). Accounting for graded performance within a discrete search framework. *Cognitive Science, 20*, 499-537.

Miller, G. A. (1956). The magic number seven, plus or minus two: Some limits on our capacity for processing information. *Psychological Review, 63*, 81-97.

Misker, J., Taatgen, N. A., & Aasman, J. (2001). Validating a tool for

simulating user interaction. In *Proceedings of the 4rth International Conference on Cognitive Modeling*. . Mahwah, NJ: Lawrence Earlbaum Associates.

Miyake, A., & Shah, P. (Eds.). (1999). *Models of working memory: Mechanisms of active maintenance and executive control.* New York, NY: Cambridge University Press.

Monk, A., Sasse, A., & Crerar, A. (1999). Affective computing: The role of emotion in human-computer interaction. British HCI Group one-day meeting in conjunction with University College London (collection of abstracts):

Neisser, U. (1976). *Cognition and reality*. San Francisco: W. H. Freeman.

Nelson, G., Lehman, J. F., & John, B. E. (1994). Integrating cognitive capabilities in a real-time task. In *Proceedings of 16th Annual Conference of the Cognitive Science Society*. 658-663. Hillsdale, NJ: Lawrence Erlbaum.

Nerb, J., Spada, H., & Ernst, A. M. (1997). A cognitive model of agents in a commons dilemma. In *Proceedings of the 19th Annual Conference of the Cognitive Science Society*. 560-565. Mahwah, NJ: Lawrence Erlbaum.

Newell, A. (1973). You can't play 20 questions with nature and win. In W. G. Chase (Ed.), *Visual information processing*. 283-308. New York, NY: Academic Press.

Newell, A. (1982). The knowledge level. *Artificial Intelligence, 18*, 87-127.

Newell, A. (1990). *Unified theories of cognition*. Cambridge, MA: Harvard University Press.

Newell, A., Shaw, J. C., & Simon, H. A. (1958). Chess-playing programs and the problem of complexity. *IBM Journal of Research and Development, 2*, 320-335.

Newell, A., & Simon, H. A. (1972). *Human problem solving*. Englewood Cliffs, NJ: Prentice-Hall.

Nielsen, T. E., & Kirsner, K. (1994). A challenge for Soar: Modeling proactive expertise in a complex dynamic environment. In *Singapore International Conference on Intelligent Systems (SPICIS-94)*. B79-B84.

Nwana, H. S. (1996). Software agents: An overview. *The Knowledge Engineering Review, 11*(3), 205-244.

O'Hara, K., & Shadbolt, N. R. (1998). Generalised directive models: Integrating model development and knowledge acquisition. *International Journal of Human-Computer Studies, 49*, 497-522.

Ohlsson, S. (1992). Artificial instruction: A method for relating learning theory to instructional design. In M. Jones & P. H. Winne (Eds.), *Adaptive learning environments: Foundations and frontiers*. 55-83. Berlin: Springer-Verlag.

Ong, R., & Ritter, F. E. (1995). Mechanisms for routinely tying cognitive models to interactive simulations. In *HCI International '95: Poster sessions abridged proceedings*. 84. Osaka, Japan: Dept. of Industrial Engineering, Musashi Institute of Technology.

Ong, R. L. (1995). Mongsu 2.0: Socket utility for hooking up Soar to simulations with sockets. Available via anonymous ftp from unicorn.ccc.nott.ac.uk as file pub/lpzfr/mongsu2.0.tar.Z.

Payne, S. J. (1991). Display-based action at the user interface. *International Journal of Man-Machine Studies, 35*, 275-289.

Pearl, J. (1982). A* - An algorithm using search effort estimates. *IEEE Transactions on Pattern Analysis and Machine Intelligence, 4*(4), 392-399.

Pew, R. W., & Mavor, A. S. (Eds.). (1998). *Modeling human and organizational behavior: Application to military simulations*. Washington, DC: National Academy Press. http://books.nap.edu/catalog/6173.html.

Piaget, J. (1954). *The construction of reality in the child*. New York, NY: Basic Books.

Picard, R. (1999). Response to Sloman's review. *The AI Magazine, 20*(1), 134-137.

Picard, R. W. (1997). *Affective computing*. Cambridge, MA: The MIT Press.

Pitrat, J. (1977). A chess combinations program which uses plans. *Artificial Intelligence, 8*, 275-321.

Popper, K. R. (1959). *The logic of scientific discovery*. New York, NY: Basic Books.

Pylyshyn, Z. (1999). Is vision continuous with cognition? The case for cognitive impenetrability of visual perception [lead article and responses]. *Behavioural and Brain Sciences, 22*(3), 341-365.

Quinlan, R. (1983). Learning efficient classification procedures and their application to chess end games. In R. S. Michalski, J. G. Carbonell, & T. M. Mitchell (Eds.), *Machine learning: An artificial intelligence approach.* Palo Alto, CA: Tioga.

Rasmussen, J. (1983). Skills, rules, knowledge: signals, signs and symbols and other distinctions in human performance models. *IEEE Transactions: Systems, Man & Cybernetics, SMC-13*, 257-267.

Rauterberg, M. (1993). AMME: An automatic mental model evaluation to analyse user behavior traced in a finite, discrete state space. *Ergonomics, 36*(11), 1369-1380.

Reason, J. (1990). *Human error*. Cambridge, UK: Cambridge University Press.

Richman, H., & Simon, H. A. (1989). Context effects in letter perception: Comparison of two theories. *Psychological Review, 96*, 417-432.

Richman, H. B., Gobet, F., Staszewski, J. J., & Simon, H. A. (1996). Perceptual and memory processes in the acquisition of expert performance: The EPAM model. In K. A. Ericsson (Ed.), *The road to excellence.* Mahwah, NJ: Erlbaum.

Richman, H. B., Staszewski, J. J., & Simon, H. A. (1995). Simulation of expert memory with EPAM IV. *Psychological Review, 102*, 305-330.

Rieman, J., Young, R. M., & Howes, A. (1996). A dual-space model of iteratively deepening exploratory learning. *International Journal of Human-Computer Studies*, 743-775.

Ritter, F., & Feurzeig, W. (1988). Teaching real-time tactical thinking. In J. Psotka, L. D. Massey, & S. A. Mutter (Eds.), *Intelligent tutoring systems: Lessons learned.* 285-301. Hillsdale, NJ: Lawrence Erlbaum.

Ritter, F. E. (1991). Towards fair comparisons of connectionist algorithms through automatically generated parameter sets. In *Proceedings of the 13th Annual Conference of the Cognitive Science Society*. 877-881. Hillsdale, NJ: Lawrence Erlbaum.

Ritter, F. E. (1993a). TBPA: A methodology and software environment for testing process models' sequential predictions with protocols (Technical Report No. CMU-CS-93-101). School of Computer Science, Carnegie Mellon University, Pittsburgh, PA.

Ritter, F. E. (1993b). Three types of emotional effects that will occur in cognitive architectures. In Workshop on architectures underlying motivation and emotion (WAUME93). School of Computer Science and Centre for Research in Cognitive Science, University of Birmingham

Ritter, F. E. (2000). A role for cognitive architectures: Guiding user interface design, contribution to the Applications of Cognitive Architectures panel (slides). In *Proceedings of the Seventh Annual ACT-R Workshop*. 85-91. Department of Psychology, Carnegie-Mellon University: Pittsburgh, PA.

Ritter, F. E., Baxter, G. D., Jones, G., & Young, R. M. (2000). Supporting cognitive models as users. *ACM Transactions on Computer-Human Interaction, 7*(2), 141-173.

Ritter, F. E., & Bibby, P. (2001). Modeling how and when learning happens in a simple fault-finding task. In *Proceedings of the Fourth International Cognitive Modeling Conference*. . Mahwah, NJ: Lawrence Earlbaum Associates.

Ritter, F. E., & Bibby, P. A. (1997). Modelling learning as it happens in a diagramatic reasoning task (Tech. Report No. 45). ESRC CREDIT, Dept. of Psychology, U. of Nottingham.

Ritter, F. E., Jones, G., Baxter, G. D., & Young, R. M. (1998a). Lessons from using models of attention and interaction. In *5th ACT-R Workshop.* 117-123. Psychology Department, Carnegie Mellon University, Pittsburgh, PA.

Ritter, F. E., Jones, R. M., & Baxter, G. D. (1998b). Reusable models and graphical interfaces: Realising the potential of a unified theory of cognition. In U. Schmid, J. Krems, & F. Wysotzki (Eds.), *Mind modeling - A cognitive science approach to reasoning, learning and discovery.* 83-109. Lengerich, Germany: Pabst Scientific Publishing.

Ritter, F. E., & Larkin, J. H. (1994). Using process models to summarize sequences of human actions. *Human-Computer Interaction, 9*(3), 345-383.

Ritter, F. E., & Major, N. P. (1995). Useful mechanisms for developing simulations for cognitive models. *AISB Quarterly, 91*(Spring), 7-18.

Ritter, F. E., & Wallach, D. P. (1998). Models of two-person games in ACT-R and Soar. In *Proceedings of the Second European Conference on Cognitive Modelling.* 202-203. Nottingham: Nottingham University Press.

Ritter, F. E., & Young, R. M. (1999). Moving the Psychological Soar Tutorial to HTML: An example of using the Web to assist learning. In D. Peterson, R. J. Stevenson, & R. M. Young (Eds.), *Proceedings of the AISB '99 Workshop on Issues in Teaching Cognitive Science to Undergraduates.* 23-24. Society for the Study of Artificial Intelligence and Simulation of Behaviour.

Ritter, F. E., & Young, R. M. (in press). Embodied models as simulated users: Introduction to this special issue on using cognitive models to improve interface design. *International Journal of Human-Computer Systems.*

Rosenbloom, P. (1998). Emotion in Soar. In *Proceedings of Soar Workshop 18.* 26-28. Vienna, VA: Explore Reasoning Systems.

Rouse, W. B. (1980). *Systems engineering models of human-machine interaction.* New York, NY: Elsevier.

Saariluoma, P. (1990). Apperception and restructuring in chess player's problem solving. In K. J. Gilhooly, M. T. G. Keane, R. H. Logie, & G. Erdos (Eds.), *Lines of thinking.* John Wiley Sons Ltd.

Salvucci, D. (in press). Predicting the effects of in-car interface use on driver performance: An integrated model approach. *International Journal of Human-Computer Studies.*

Sanderson, P. M., & Fisher, C. A. (1994). Exploratory sequential data analysis: Foundations. *Human-Computer Interaction, 9*(3&4), 251-317.

Sanderson, P. M., McNeese, M. D., & Zaff, B. S. (1994). Handling complex real-world data with two cognitive engineering tools: COGENT and MacSHAPA. *Behavior Research Methods, Instruments, & Computers, 26*(2), 117-124.

Schofield, D., & Denby, B. (1995). Visualizing the risk - Virtual reality training for LHDs. *Engineering and Mining Journal, 196*(2), 39-40.

Schreiber, G., Akkermans, H., Wielinga, B., Anjewierden, A., Shadbolt, N. R., de Hoog, R., & Van de Velde, W. (in press). *Engineering and managing knowledge*. Cambridge, MA: MIT Press.

Schunn, C. D., Reder, L. M., Nhouyvanisvong, A., Richards, D. R., & Stroffolino, P. J. (1997). To calculate or not calculate: A source activation confusion (SAC) model of problem-familiarity's role in strategy selection. *Journal of Experimental Psychology: Learning, Memory, & Cognition, 23*, 1-27.

Schwamb, K. B. (1998). Soar at ERS. In *Proceedings of Soar Workshop 18.* 41-46. Vienna, VA: Explore Reasoning Systems.

Schwamb, K. B., Koss, F. V., & Keirsey, D. (1994). Working with ModSAF: Interfaces for programs and users. In Proceedings of the Fourth Conference on Computer Generated Forces and Behavioral Representation. Technical Report IST-TR-94-12. Institute for Simulation and Training

Sekuler, R., & Blake, R. (1994). *Perception* (3rd ed.). New York, NY: Alfred A. Knopf.

Senders, J. W., & Moray, N. P. (1991). *Human error: Cause, prediction, and reduction*. Hillsdale, NJ: Lawrence Erlbaum.

Shadbolt, N. R., & Burton, A. M. (1995). Knowledge elicitation: A systematic approach. In J. R. Wilson & E. N. Corlett (Eds.), *Evaluation of human work: A practical ergonomics methodology*. 406-440. London: Taylor and Francis.

Shadbolt, N. R., & O'Hara, K. (1997). Model-based expert systems and the explanation of expertise. In P. Feltovich, K. Ford, & R. Hoffman (Eds.), *Expertise in context.* Cambridge, MA: AAAI and MIT Press.

Siegler, R. S. (1987). The perils of averaging data over strategies: An example from children's addition. *Journal of Experimental Psychology, 115*, 250-264.

Simon, H. A. (1955). Prediction and hindsight as confirmatory evidence. *Philosophy of Science, 22*, 227-230.

Simon, H. A. (1974). How big is a chunk? *Science, 183*, 482-488.

Simon, H. A., & Chase, W. G. (1973). Skill in chess. *American Scientist, 61*(393-403).

Simon, H. A., & Gilmartin, K. J. (1973). A simulation of memory for chess positions. *Cognitive Psychology, 5*, 29-46.

Skinner, B. F. (1957). *Verbal behavior*. Englewood Cliffs, NJ: Prentice-Hall.

Sloman, A. (1998a). Damasio, Descartes, alarms and meta-management. In *Proceedings of the International Conference on Systems, Man, and Cybernetics (SMC98).* 2652-2657. IEEE.

Sloman, A. (1998b). What's an AI toolkit for? In B. Logan & J. Baxter (Eds.), *Proceedings of the AAAI-98 Workshop on Software Tools for Developing Agents.* 1-10. Menlo Park, CA: AAAI Press.

Sloman, A. (1999). Review of *Affective Computing* by Rosalind Picard (1997). *The AI Magazine, 20*(1), 127-133 (followed by reply by Picard).

Sloman, A. (2000). Architectural requirements for human-like agents both natural and artificial (What sorts of machines can love?). In K. Dautenhahn (Ed.), *Human cognition and social* *agent technology. Advances in consciousness research.* 163-195. Amsterdam: John Benjamins.

Sloman, A., & Logan, B. (1999). Building cognitively rich agents using the Sim_Agent toolkit. *Communications of the Association of Computing Machinery, 42*(3), 71-77.

St. Amant, R., & Riedl, M. O. (in press). A perception/action substrate for cognitive modeling in HCI. *International Journal of Human-Computer Studies.*

Stern, Ray, R., & Quigley, K. (2001). *Psychophysiological Recording*. Oxford, UK: Oxford University Press.

Sun, R., & Ling, C. X. (1998). Computational cognitive modeling, the source of power, and other related issues. *AI Magazine, 19*(2), 113-120.

Sun, R., Merrill, E., & Peterson, T. (1998). Skill learning using a bottom-up hybrid model. In F. E. Ritter & R. M. Young (Eds.), *Proceedings of the 2nd European Conference on Cognitive Modelling*. 23-29. Thrumpton, Nottingham: Nottingham University Press.

Synthetic Environments Management Board (1998). Synthetic environments and simulation: The report of the Defence and Aerospace Foresight Panel Technology Working Party No. Vickers Defence Systems, Crossgates, Leeds (UK).

Tabachneck-Schijf, H. J. M., Leonardo, A. M., & Simon, H. A. (1997). CaMeRa: A computational model of multiple representations. *Cognitive Science, 21*(3), 305-350.

Tambe, M., Johnson, W. L., Jones, R. M., Koss, F., Laird, J. E., Rosenbloom, P. S., & Schwamb, K. (1995). Intelligent agents for interactive simulation environments. *AI Magazine, 16*(1), 15-40.

van Lent, M. (1999). Learning by observation in complex domains. In *Proceedings of Soar Workshop 19*. 70-73. U. of Michigan Soar Group.

van Lent, M., & Laird, J. (1999). Learning by observation in a tactical air combat domain. In *Proceedings of the 7th Computer Generated Forces and Behavioural Representation Conference*. 239-250.

VanLehn, K., & Garlick, S. (1987). Cirrus: An automated protocol analysis tool. In P. Langley (Ed.), *Fourth Machine Learning Workshop*. 205-217. Los Altos, CA: Morgan-Kaufman.

vanSomeren, M. W., Barnard, Y. F., & Sandberg, J. A. C. (1994). *The Think Aloud Method: A practical guide to modelling cognitive processes*. London and San Diego: Academic Press.

Vicente, K. (1998). *Cognitive work analysis*. Mahwah, NJ: Lawrence Erlbaum.

Wang, H., Johnson, T. R., & Zhang, J. (1998). UEcho: A model of uncertainty management in human abductive reasoning. In M. A. Gernsbacher & S. J. Derry (Eds.), *Proceedings of the Twentieth Annual Conference of the Cognitive Science Society.* 1113-1129. Mahwah, NJ: Lawrence Erlbaum.

Whalen, P. J. (1999). Fear, vigilance, and ambiguity: Initial neuroimaging studies of the human amygdala. *Current Directions, 7*(6), 177-188.

Wickens, C. D. (1992). *Engineering psychology and human performance* (2nd ed.). New York, NY: HarperCollins.

Wielinga, B. J., Schreiber, A. T., & Breuker, J. A. (1992). KADS: A modelling approach to knowledge engineering. *Knowledge Acquisition Journal, 4*(1), 5-53.

Wilkins, D. (1980). Using pattern and plans in chess. *Artificial Intelligence, 14*, 165-203.

Williams, M., Schofield, D., & Denby, B. (1998). The development of an intelligent haulage truck simulator for improving the safety of operation in surface mines. In *Proceedings of Virtual Worlds*. .

Wittig, T., & Onken, R. (1992). Knowledge based cockpit assistant for controlled airspace flight operation. In H. G. Stassen (Ed.), *Analysis, design and evaluation of man-machine systems 1992*. Oxford, UK: Pergamon Press.

Woods, D. D., Patterson, E. S., Roth, E. M., & Christoffersen, K. (1999). Can we ever escape from data overload? A cognitive systems diagnosis. In *Proceedings of the Human Factors and Ergonomics Society 43rd annual meeting*. 174-178. Houston, TX:

Woods, D. D., Roth, E. M., & Pople, H. J. (1987). *Cognitive environment simulation: An artificial intelligence system for human performance assessment*. Washington, DC: Nureg-CR-4862.

Wooldridge, M., & Jennings, N. R. (1995). Intelligent agents: Theory and practice. *Knowledge Engineering Review, 10*(2), 115-152.

Wooldridge, M., & Rao, A. (Eds.). (1999). *Foundations of rational agency*. Dordrecht, The Netherlands: Kluwer Academic.

Yost, G. R. (1992). *TAQL: A Problem Space Tool for Expert System Development*. PhD, School of Computer Science, Carnegie-Mellon University.

Yost, G. R., & Newell, A. (1989). A problem space approach to expert system specification. In *Eleventh International Joint Conference on Artificial Intelligence*. 621-627.

Young, R. M. (1999). A zoo of browsable, runnable cognitive models. In D. Peterson, R. J. Stevenson, & R. M. Young (Eds.), *Proceedings of the AISB '99 Workshop on Issues in Teaching Cognitive Science to Undergraduates*. 25. The Society for the Study of Artificial Intelligence and Simulation of Behaviour.

Young, R. M., & Barnard, P. J. (1987). The use of scenarios in human-computer interaction research: Turbocharging the tortoise of cumulative science. In J. M. Carroll & P. Tanner (Eds.), *Human Factors in Computing Systems and Graphics Interfaces (CHI & GI 87)*. 291-296. New York, NY: ACM.

Young, R. M., Barnard, P. J., Simon, A. J., & Whittington, J. E. (1989a). How would your favourite user model cope with these scenarios? *ACM SIGCHI Bulletin, 20*(4 (April)), 51-55.

Young, R. M., Green, T. R. G., & Simon, T. (1989b). Programmable user models for predictive evaluation of interface designs. In *CHI'89 Conference Proceedings: Human Factors in Computing Systems*. 15-19. New York, NY: ACM Press.

Young, R. M., & Lewis, R. L. (1999). The Soar cognitive architecture and human working memory. In A. Miyake & P. Shah (Eds.), *Models of working memory: Mechanisms of active maintenance and executive control*. 224-256. New York, NY: Cambridge University Press.

Zettlemoyer, L. S., & St. Amant, R. (1999). A visual medium for programmatic control of interactive applications. In *CHI '99, Human Factors in Computer Systems*. 199-206. New York, NY: ACM.

Appendix 0: Author contact details

Prof. David Elliman

School of Computer Science & IT

University of Nottingham

Jubilee Campus

Triumph Road

Nottingham

NG8 1BB

UK

Tel: 44 1159514208

Fax: 44 1159514254

Gordon D. Baxter

Department of Psychology

University of York

Heslington

York YO10 5DD

United Kingdom

Phone: +44 1904 433170

Fax: +44 1904 433181

e-mail: g.baxter@psych.york.ac.uk

web: http://www-users.york.ac.uk/~gdb1

Dr. Fernand Gobet

Allan Standen Reader in Intelligent Systems

ESRC Centre for Research in Development,

Instruction and Training

School of Psychology

University of Nottingham

Nottingham NG7 2RD

England

+ 44 (115) 951 5402 (phone)

+ 44 (115) 951 5324 (fax)

http://www.psychology.nottingham.ac.uk/staff/Fernand.Gobet/

Brian Logan

School of Computer Science & IT,

Jubilee Campus,

University of Nottingham

Nottingham NG8 1BB UK

Email: bsl@cs.nott.ac.uk

Phone: +44 115 846 6509

Fax: +44 115 951 4254

Aaron Sloman

School of Computer Science

The University of Birmingham

Birmingham, B15 2TT

England, UK

EMAIL A.Sloman@cs.bham.ac.uk

WWW: http://www.cs.bham.ac.uk/~axs

Phone numbers:

Office: +44-121-414-4775 Secretary:

+44-121-414-3711

Fax: +44-121-414-4281

Appendix 1: List of participants

Attendees at the workshop on Techniques for Modelling Human Performance in Synthetic Environments, Nottingham, 17th March , 1999.

Belavkin, Roman. University of Nottingham, rvb@cs.nottingham.ac.uk

Croker, Stephen. University of Nottingham, sfc@psychology.nottingham.ac.uk

Elliman, David. University of Nottingham, dge@cs.nottingham.ac.uk

Gobet, Fernand. University of Nottingham, frg@psychology.nottingham.ac.uk

Greig, Ian. DERA, igreig@dera.gov.uk

Howes, Andrew. University of Cardiff, howesa@cardiff.ac.uk

Logan, Brian. University of Birmingham, b.logan@cs.bham.ac.uk

Lonsdale, Peter. University of Nottingham, lpxprl@psychology.nottingham.ac.uk

Page, Ian. DERA, ipage@dera.gov.uk

Ritter, Frank (Chair). University of Nottingham, Frank.Ritter@nottingham.ac.uk

Russell, Simon. DERA, russell@dera.gov.uk

Shadbolt, Nigel. University of Nottingham, nrs@psychology.nottingham.ac.uk

Sheppard, Colin. DERA, csheppard@dera.gov.uk

Sloman, Aaron. University of Birmingham, a.sloman@cs.bham.ac.uk

Widdowson, Marc. VEGA Group, marc.widdowson@lineone.net

Young, Richard. University of Hertfordshire, r.m.young@herts.ac.uk

Appendix 2: Description of ACT-R and Soar

Index

Index terms go here

human error

EPAM

ACT-R

Soar

working memory

attention

Cognitive Reliability and Error Analysis Method

CREAM

intelligent architectures

Cognitive Simulation Model

COSIMO

Cognitive Environment Simulation

CES

AIDE

Step ladder model

Skill-based, rule-based, knowledge based model

APEX

Simplified Model of Cognition

SMoC

Contextual Control Model

CoCoM

erroneous behaviour

erroneous action

modelling error

taxonomy of human error

Confidential Human Factors Incident Reporting Programme

CHIRP

modelling methodology