Chapter 2

Related Research

2.1 Introduction

The goal of this study is to contribute to the automation of military tactical simulator training, so that in the future trainees can train at any time and by themselves. In Chapter 1 we stated that for this type of training it is minimally required to validly represent human behavior in the simulated environment, and to provide feedback to the behavior of the trainee. As a possible solution to automate such training we introduced the notion of software agents: we proposed that instead of humans, agents can deliver these required behaviors. In addition, we put forward that software agents that are to display human-like behavior should incorporate a cognitive model. The aspects of cognition that this model needs to represent will depend on the task of the agent in the simulated environment and the goal of the training. Moreover, the software agents that are to generate feedback on the behavior of the trainee should be able to reason about the trainee's cognitive process, and can do this by means of cognitive models. Once more, the goal and task of the training will influence the processes the agent has to reason about, and thus the cognitive models required.

The study of cognition, the determination of its distinct capabilities, the investigation of these separate capabilities as well as the entire system by performing experiments and building models, is labeled *Cognitive Science*. The research conducted and described in this dissertation is closely related to this field of research. When developing cognitive models it has to be established which capabilities are required, how these should operate together, and how all this can be modeled. The research field of *Artificial Intelligence* also studies (aspects of) cognition, and determines and models its distinct capabilities as well as the entire system. However, its experiments do not investigate whether the models validly represent human cognition, but if they display behavior at the level of intelligence required for the task they are developed for.

Taking into account the two goals of modeling cognition described in Section 1.2.1, Cognitive Science can be considered a natural science, while Artificial Intelligence is its design science counterpart. Our research draws inspiration from Cognitive Science, but ultimately belongs to Artificial Intelligence. We are concerned with the modeling of cognition, because we want to generate human-like behavior and feedback on human behavior, and not because we want to penetrate the processes underlying cognition.

Chapter Overview

In this chapter, we discuss research that is related to our research questions. The function of this chapter is to provide background knowledge on a wide variety of topics, to increase the readability of the following chapters for those that are not familiar with the topics addressed. Not only this chapter, but also the introduction sections of the coming chapters and of the papers they embed discuss related research. However, especially in the papers, these discussions already focus on addressing specific design issues, while here we aim to provide a generic overview of several research topics.

We start this chapter with discussing global aspects of cognition: in Section 2.2.1 we introduce various categorizations of capabilities that presumably constitute cognition. Next, in Section 2.2.2, we elaborate on the origin of suboptimal task performance by humans. This is an aspect of human behavior that our software agents should be able to display, and provide feedback about.

Then, we discuss how cognition could be modeled. In Section 2.3.1 we shortly discuss various approaches toward the modeling of cognition, i.e., using qualitative and/or quantitative terms, and introduce our approach to the modeling of cognition. In addition, in Section 2.3.2, we discuss the notion of integrated architectures. We address two cognitive architectures in detail: Soar (Laird et al., 1987) and ACT-R (Anderson and Lebiere, 1998; Anderson et al., 2004), and elaborate on their relevancy for our research objectives.

Next, we discuss possible application areas for cognitive models. In Section 2.4.1 we elaborate on a variety of application domains for which software agents have been developed with the purpose to generate human-like behavior. For each domain we discuss how the requirements of the behavior of those agents match the requirements of the behavior of an agent for our goal: a tactical training simulation. In Section 2.4.2, we elaborate on various methods to generate feedback, and discuss software models capable of generating feedback.

Finally, in Section 2.5, we conclude this chapter and look forward to the next.

2.2 Aspects of Cognition

What does it actually entail ‘to be cognitive’, i.e., what kind of processes does cognition embed? This is a hard question, since *cognition* is an umbrella notion. The Oxford English Dictionary (1994) defines cognition as:

> “The action or faculty of knowing taken in its widest sense, including sensation, perception, conception, etc., as distinguished from feeling and volition.”

For the studying and modeling of cognition it is useful to divide this umbrella term in clear conceptual parts. In this section we start with the introduction of several proposals for the division of cognition. Next, we elaborate on the generic aspect of cognition that it is not always rational. We introduce several examples of cognitive biases, and discuss their (presumable) origin.

2.2.1 Cognitive Capabilities

When thinking about cognition it is easy to come up with specific cognitive processes that humans posses, ranging from selecting a filling for their sandwich, to determining the best time to call their grand-mother. Such processes can be categorized as being instances of specific cognitive capabilities: decision-making and scheduling, respectively. Many of these distinct capabilities have been identified and extensively researched within Cognitive Science and Artificial Intelligence, with as respective goals to understand and to model them.

In this section we will not elaborate on specific research projects that focused on the studying and modeling of a specific capability. The introduction sections of Chapters 3, 4, and 5 will, since these chapters focus on specific capabilities. Instead, we present three research projects that have attempted to provide an overview of cognitive capabilities. These projects serve as examples; it is not our goal to subscribe to one of these views. They are merely provided as illustrative for the number and level of capabilities that can be distinguished within human cognition.

Gordon (2005) states that there are sixteen functional requirements (capabilities) that a model of human cognition should possess. In Table 2.1 we present the taxonomy of the sixteen functional classes of cognitive models that Gordon proposes. Gordon bases this taxonomy on formal theories of ‘commonsense psychology’ and states that each of these functions must be encoded within an agent for it to be able to perform commonsense reasoning. Although not all these capabilities are required for the current task, it provides a good overview of the processes an agent should eventually be capable of in order to mimic human behavior in all its facets.

Table 2.1: The 16 Functional Classes of Cognitive Models according to Gordon (2005)

Functional Class	**Summary**
1. Knowledge and inference	Models of how people maintain and update their beliefs in the face of new information
2. Similarity judgment	Models of how people judge things to be similar, different, or analogous
3. Memory	Models of memory storage and retrieval
4. Emotion	Models of emotional appraisal and coping strategies
5. Envisionment	Models of how people reason about causality, possibility, and intervention in real and imagined worlds
6. Explanation	Models of the process of generating explanations for events and states with unknown causes
7. Expectation	Models of people come to expect that certain events and states will occur in the future, and how they handle expectation violations
8. Theory of Mind reasoning	Models of how people reason about the mental states and processes of other people and themselves
9. Threat detection	Models of how people identify threats and opportunities that may impact the achievement of their goals
10. Goal management	Models of how people prioritize and reconsider the goals that they choose to pursue
11. Planning	Models of the process of selecting a course of action that will achieve one's goals
12. Design	Models of how people develop plans for the creation or configuration of an artifact, process or information
13. Scheduling	Models of how people reason about time and select when they will do the plans that they intend to do
14. Decision making	Models of how people identify choices and make decisions
15. Monitoring	Models of how people divide their attention in ways that enable them to wait for, check for, and react to events in the world and in their minds
16. Plan execution	Models of the way that people put their plans into action and control their own behavior

Not all the aspects mentioned by Gordon are required for our task, and neither are they for many other tasks for which cognitive agents have been developed. In contrast with Gordon who started his listing of cognitive capabilities from a natural science, namely psychology, Langley et al. (2006) present a list of cognitive capabilities taking a design science approach. Langley et al. discuss the capabilities and functionalities that a cognitive agent could embed by studying architectures developed for the goal of intelligent behavior generation. They divide these capabilities into nine main areas, see Table 2.2. For clarity of presentation, their last capability ‘Remembering, Reflection and Learning’ is split into the two capabilities ‘Remembering and Reflection’ and ‘Learning’.

Table 2.2: The 9 Capabilities of Cognitive Agents according to Langley et al. (2006)

Capability	**Summary**
1. Recognition and Categorization	The ability to recognize situations or events as instances of known or familiar patterns
2. Decision Making and Choice	The ability to make decisions and select among alternatives
3. Perception and Situation Assessment	The ability to sense, perceive, and interpret some external environment
4. Prediction and Monitoring	The ability to predict future situations and events accurately
5. Problem Solving and Planning	The ability to generate plans and solve problems
6. Reasoning and Belief Maintenance	The ability that lets an agent augment its knowledge state
7. Execution and Action	The ability to execute skills and actions in the environment
8. Interaction and Communication	The ability to communicate and transfer knowledge from one agent to the other
9.a. Remembering and Reflection	The ability to encode and store the results of cognitive processing in memory and to retrieve or access them later
9.b. Learning	The ability to generalize beyond specific beliefs and events

We appreciate the pragmatics of the work of Langley et al. compared to that of Gordon, and decided to take their capability classification as the basis for our Capability Description Framework, see Chapter 6. Even more pragmatic is the division followed by Pew and Mavor (1998), who discuss architectures for modeling individual human behavior with the specific focus of their application to military simulations. Because of its military focus, this piece of work functioned as one of the starting points of our research. Pew and Mavor analyze existing architectures on six key areas, see Table 2.3.

The first five areas can be considered cognitive capabilities; the latter however is an odd one out. Nevertheless, behavior moderators are important for modeling cognition and human behavior, since human behavior is known to vary due to internal aspects as

Table 2.3: The key areas of architectures according to Pew and Mavor (1998)

Key Area
1. Attention and Multi-Tasking
2. Memory and Learning
3. Planning
4. Decision Making
5. Situation Awareness
6. Behavior Moderators

stress and emotion. Behavior moderators enable the modeling of this variety of behavior, making it more human-like. In the subsequent sections we elaborate on such behavior moderators in respect to the modeling of human suboptimal task performance, with a specific focus on the modeling of cognitive biases.

This section illustrates that there is no clear consensus on which capabilities constitute cognition. Cognitive Science is occupied with clarifying this fundamental question; Artificial Intelligence on the other hand simply models those aspects that are sufficient for generating intelligent, human-like behavior for a given task. For our purpose it holds that the cognitive capabilities that need to be modeled will vary between tasks and training objectives, e.g., an opposing naval force does not need to be able to communicate.

2.2.2 Cognitive Biases

Previously, we explained that the modeling of cognitive biases is relevant for the training of military personnel. On the one hand it is important to train them in recognizing and dealing with their own biases, on the other hand they need to be trained on dealing with the biased behavior of their team mates.

This section elaborates on research concerning cognitive biases. It starts with several examples of cognitive biases, after which the general limitations of human cognition are discussed that by themselves might lead to suboptimal behavior. Although the previous section only mentioned capabilities as the parts that make up cognition, cognitive processes are in addition influenced by other processes, e.g., emotion and stress. At the end of this section we elaborate on the role these so-called behavior or performance moderators play in the functioning of cognition.

When the cognitive mechanisms underlying biases are known, it will be possible to formalize these mechanisms in a cognitive agent model. This model can then be embedded in a software agent that will therefore be able to display more human-like, possibly biased, behavior. In addition it can be used to compare the behavior of the trainee with, so that possible occurring biases can be detected.

Cognitive Bias Examples

Dozens of cognitive biases, operating on various levels and influencing a variety of cognitive processes, have been identified. For extensive discussions on human biases see Baron (2000), Reason (1990), who refers to them as error forms, or Pohl (2004), who calls them cognitive illusions. To facilitate the discussion of cognitive biases we divide them in 4 classes that differ in the level at which they operate: social (attributional), memory, judgment, and decision-making biases. This is no clear-cut division, many biases or their underlying mechanisms overlap categories, e.g., a mechanism that leads to a judgment bias might a the same time lead to a decision-making bias.

A *social or attributional bias* is a cognitive bias that affects the way it is determined who or what was responsible for an event or action (attribution). A well-known example is the *correspondence bias*, which is 'the tendency to draw inferences about a person's unique and enduring dispositions from behaviors that can be entirely explained by the situations in which they occur' (Gilbert and Malone, 1995). In other words, people have an unjustified tendency to assume that a person's actions are not the result of that person's social and environmental situation, but of the 'kind' of person that person is.

A *memory bias* is a cognitive bias that either enhances or impairs the recall of a memory, or that alters the content of memory that is claimed to be recalled. Many memory biases are simply referred to as effects. Two well-known memory effects are primacy and recency: the fact that respectively the first and the last items on a list show an advantage in memory, i.e., are easier recalled than the items in the middle of the list. An example of a memory bias is the *hindsight bias*, which is 'the tendency for people with outcome knowledge to believe falsely that they would have predicted the reported outcome of an event' (Hawkins and Hastie, 1990). In other words, people have an unjustified tendency to claim after events have happened that they 'knew it all along'.

A *judgment bias* is a cognitive bias that affects estimated probabilities or beliefs, and can therefore have impact on the quality of human performance. Two examples of judgment biases are also referred to as primacy and recency, but are now defined as the tendency to weigh initial events more than subsequent events, and the tendency to weigh recent events more than earlier events, respectively. Multiple researchers (Adelman et al., 1996; Wang et al., 2000) have shown that these effects occur in tactical decision-making. For example, when participants were presented the same friendly/neutral/hostile evidence concerning yet unidentified tracks, their order influenced the participants' final judgment concerning the contact's identity. Another well-know judgment bias is the conjunction fallacy, which is the tendency to 'regard a conjunctive event as more probable than one of its components' (Tversky and Kahneman, 1982). For example, Tversky and Kahneman (1982) show that people judge the probability of 'Linda is a bank teller and is active in the feminist movement' as being higher than the probability of 'Linda is a bank teller'. Several other judgment biases will be discussed in the next section.

A *decision-making bias* is a cognitive bias that affects the way a decision is made. This can happen in a multitude of ways, among others by a false probability estimation, i.e., through a judgment bias. A wide-spread decision-making bias is the confirmation bias, which is the tendency to search for or interpret new information in a way that confirms one's preconceptions and opinions, and to ignore, undervalue, or not look for information which contradicts these prior beliefs. This bias has been shown to occur frequently in military tactical decision-making (Fewell and Hazen, 2005).

Mechanisms underlying Cognitive Biases

The study of cognitive biases progressed greatly by the work of Tversky and Kahneman (1974) who investigated in laboratory settings human judgment and decision-making. They compared the judgments and decisions actually made by humans with those that should be made if they would follow the rules of rational choice theory. Rational choice theory assumes that humans select the best option given the current circumstances, and that that value judgment is determined by a stable preference function. However, Tversky and Kahneman found that human choices differ in systematic and predictable ways from the rational choice. These ways are referred to as cognitive biases.

Twenty years earlier Simon (1956) was among the first to oppose the view of humans as rational entities that can calculate the best option; a view attributing humans with unlimited computational powers and perfect knowledge. Norman and Bobrow (1975) stress that human processes are limited exactly by these two factors: a process can be limited in its performance by limits in the amount of available processing resources (such as memory or processing effort) or by limits in the quality of the data available to it. In his work Simon proposes that humans are rational but under the constraints of limited time, knowledge and computational capacities. He states that humans should not aim for an optimal solution, but search until a solution is found that is *satisficing*.

It is often claimed that the origin of cognitive biases lies in the limitations of human information-processing capacity. It is generally acknowledged that human cognition is limited in the amount of information it can attend to and process at a single time. The classical paper by Miller (1956) stresses that the capacity of short-term memory for chunks of information is limited to 'the magical number seven, plus or minus two'. Others have researched and identified limits in cognitive processing capacity, which influences the ability to perform multiple tasks simultaneously (e.g., Kahnemann, 1973). The limited capacity of human cognition gives rise to specific processes, which are, or might lead to, cognitive biases.

Heuristics Tversky and Kahneman (1974) claim that the biases they identified are, at least partly, the result of decision-making using heuristics. Heuristics, also referred to as rules of thumb, are mental shortcuts that humans use to speed up and simplify their decision-making process. It is assumed that heuristics have evolved due to, and are applied in reaction to, the limitations of cognition. Tversky and Kahneman identify three such heuristics:

- **Representativeness** - the tendency to evaluate the probability of object or event A to belong to, or originate from, class or process B by the degree to which A resembles, or is representative for B.
- **Availability** - the tendency to assess the frequency of a class or the probability of an event by the ease with which instances or occurrences can be brought to mind.
- **Adjustment and anchoring** - the tendency to make an estimate by starting from an initial value and adjusting that to yield the final answer, which unfortunately leads to an answer that is biased toward the initial value.

Heuristics speed up the decision-making process and often succeed, but sometimes fail. The Kahneman and Tversky's heuristics-and-biases program (Kahnemann et al., 1982) focused on these failures by constructing decision problems that led to structural decision errors (cognitive biases) caused by the use of heuristics. As a result heuristics received a negative connotation.

Gigerenzer and colleagues (Gigerenzer et al., 1999) oppose this view and stress above all the usefulness of heuristics. Their research explores 'fast and frugal heuristics - simple rules in the mind's adaptive toolbox for making decisions with realistic mental resources'. Todd and Gigerenzer (2000) state that heuristics can enable both living organisms and artificial systems to make smart choices quickly and with a minimum of information by exploiting the way that information is structured in particular environments. The latter aspect is also referred to as a heuristic's ecological rationality; the degree to which it is adapted to the structure of an environment.

Hertwig and Todd (2003) go even further in questioning the negative status of cognitive limitations and forward the thesis that they actually enable, instead of disable, important adaptive functions. Along the same line, Schooler and Hertwig (2005) introduce a model that suggests that forgetting facilitates human inference performance by strengthening the chain of correlations, linking the target criteria, environmental frequencies, and fundamental memory-retrieval processes. Still, the argument that heuristics are adaptive given their ecological rationality implies that when they are applied in a differently structured environment, they might lead to wrong behavior.

Error Phenotype versus Error Genotype A useful distinction when talking about biases is made by Hollnagel (1993). Hollnagel introduces the notion of error phenotype for the observable manifestation of an error, and the notion of error genotype as the mental activity that supposedly underlies and produces that observable manifestation. Many cognitive biases are the phenotypes of heuristics. Tversky and Kahneman (1974) describe the three heuristics mentioned above, but the number of biases they describe that these heuristics lead to is significant larger. One example of a cognitive bias following from a heuristic is the previously described conjunction fallacy. Tversky and Kahneman (1982) note about this:

> "A conjunction can be more representative than one of its constituents, and instances of a specific category can be easier to imagine or to retrieve than instances of a more inclusive category. The representativeness and availability heuristics therefore can make a conjunction appear more probable than one of its constituents."

However, by no means does the application of a heuristic lead in all situations to the emergence of a bias. On the other hand, for memory biases and effects it is generally the case that they are the observed 'error' phenotypes from the 'error' genotypes that are the basic mechanisms of memory. Therefore, memory biases and effects are a constant finding within and between subjects.

Cognitive Biases in Real Life Most studies into cognitive biases have taken place in laboratory settings. However, they do occur in real life, and influence human judgment and decision-making: various famous accidents in military warfare have been attributed to false judgments or decision-making under influence of cognitive biases. Perrin et al. (1993) examine in an empirical study human judgment biases under conditions of uncertainty and time pressure in surface Anti-Air Warfare (AAW). To be precise, they studied whether the judgments of naval tactical action officers in a realistic task simulation exhibit characteristics of the heuristics and biases of availability, representativeness, anchoring-contrast, and confirmation. This is what they found:

> "Our subjects ignored baseline trends when other case-specific information was available (representativeness and availability). They were significantly influenced by the order they received evidence, showing a recency effect characteristic of contrast. Additionally, as is characteristic of confirmation bias, they recalled much more of the information that was consistent with their final hypothesis and evaluated it as more informative than the inconsistent data, regardless of which hypothesis they had adopted."

Manifestations of Cognitive Biases

The manifestation of cognitive biases can be discussed from two viewpoints. First, its timing, a dynamical aspect, can be discussed. In other words; it can be investigated which external and internal circumstances might inflict the occurrence of a cognitive bias. Second, its positioning, a static aspect, can be investigated: which part of the cognitive process it influences.

Positioning of Biases When we divided cognitive biases in several classes in Section 2.2.2, we touched upon the issue of their position, namely as influencing social, memory, judging or decision-making processes. Another way to position them is by means of the Skills, Rules, Knowledge (SRK) framework developed by Rasmussen (1983) to define three types of cognitive processes present in operator information processing. Reason (1990) classifies human errors by means of these three process types as follows:

- **Skill-based level:** At the skill-based level, human performance is governed by stored patterns of preprogrammed instructions represented as analogue structures in a time-space domain. Errors at this level are related to the intrinsic variability of force, space, or time coordination.
- **Rule-based level:** The rule-based level is applicable to tackling familiar problems in which solutions are governed by stored rules (productions) of the type if (state) then (diagnosis), or if (state) then (remedial action). Here, errors are typically associated with the misclassification of situations leading to the application of the wrong rule, or with the incorrect recall of procedures.
- **Knowledge-based level:** The knowledge-based level comes into play in novel situations for which actions must be planned on-line, using conscious analytical processes and stored knowledge. Errors at this level arise from resource limitations ('bounded rationality') and incomplete or incorrect knowledge. With increasing expertise, the primary focus of control moves from the knowledge-based towards the skill-based levels; but all three levels can co-exist at any one time.

Another classification that can be made is on the level at which cognitive biases have impact: *within* a process, or *between* processes. For example, most memory biases influence the recollection *process* of memories, i.e., which memory is retrieved given a certain query. Other biases might influence *which* recollection process executes, i.e., which query is executed. The confirmation bias is a good example to clarify this distinction further, as it operates within and between cognitive processes. Within a specific cognitive process the confirmation bias influences the weight given to (dis)confirming information. On the process level it influences the next information-searching action in the world, and biases this to confirming instead of conflicting information.

Although it might be useful to train military personnel in recognizing and dealing with all types of cognitive biases, we focus on a subset that is relevant to the cognitive models we develop. In Chapters 3 and 4 we focus on the development of a belief and memory framework for cognitive agents, and model biases that occur within processes, like in the retrieving and reasoning upon information. In Chapter 5 we focus on the development of two control frameworks for cognitive agents, for which we model biases and heuristics that occur between processes, so on the level of control.

Timing of Biases We have not elaborated extensively on *when* heuristics and biases occur. We do have mentioned that they are generally considered to be a consequence of the limitations of human cognition. When there is too much information to take into account, or too little time to do so, humans apply heuristics and biases might occur.

A second generally held opinion is that they occur especially under conditions of stress and fatigue, e.g., Cohen et al. (1986) mention that everybody finds that slips of actions increase with tiredness and stress. The general idea behind this is that these circumstances shrink the cognitive capacity of humans, and make the limitations of the cognitive system more prevalent. For example, the results of a study by Harris et al. (2005) on the influence of extended stress on human performance support the model that stress decreases resource reserves. They state that the extended stress circumstances decrease the participants' ability to continue to mobilize the resources required to perform complex tasks. For a more detailed account of the influence of stress, workload, and fatigue on cognition, see, e.g., Reason (1988) and Hancock and Desmond (2001).

Studying the underlying mechanisms of heuristics and cognitive biases is useful, not only to understand them, but also to model them. In the next section we introduce various approaches toward the modeling of cognition, as well as integrated architectures, which are implemented theories of the mechanisms underlying human behavior. In addition, we elaborate on the modeling and appearance of cognitive biases in two of these integrated architectures.

2.3 Models of Cognition

The previous section elaborated on the fact that cognition comprises a wide variety of cognitive capabilities. In addition, it discussed cognitive biases and how these, to a smaller or larger extent, influence these capabilities. Dividing cognition in clear conceptual parts, like capabilities, facilitates the modeling of such parts. In particular, because each of these parts can be modeled by a modeling approach most suited. However, the definition and modeling of ‘parts of cognition’ is not enough for implementing a cognitive agent model. In addition, it has to be determined how these parts interact and in which order they operate. In the introduction of Chapter 5, Control Framework, we will elaborate on this topic.

We start this section with a discussion of a variety of modeling approaches that have been used to model (parts of) cognition. Next, we elaborate on integrated architectures that stress the importance of studying the integrated architecture underlying cognition, instead of only focusing on the modeling of its parts.

2.3.1 Modeling Approaches

Nowadays, most researchers agree that hybrid modeling is the appropriate way to model cognition. Hybrid modeling is used as a term to express the fact that a cognitive model embeds qualitative (symbolic) as well as quantitative (numeric) elements. In the models developed in this study both qualitative and quantitative elements play a role; the former to clearly label knowledge, the latter to handle this knowledge in subtle ways.

Hybrid models acknowledge the intuitive feeling that humans embed two approaches toward the processing of cognitive tasks (Smolensky, 1988; Sloman, 1996). First, they can reason consciously following certain rules and algorithms, and these can be captured well by qualitative systems. But second, humans execute cognitive processes on a more unconscious, automatic level formed by associations, which are better captured by quantitative means. Qualitative systems are well-suited to generate rational behavior in certain tasks, but they usually lack the means to generate typical aspects of human behavior, like the ability to cope with uncertain or incomplete information. In order to model human behavior in all its aspects, a hybrid modeling approach is required (Minsky, 1992). Sun (2002b), and more recently Bader and Hitzler (2005), provide comprehensive surveys of various types of hybrid systems.

Qualitative Modeling Approaches

A few decades ago it was thought that ‘a physical symbol system has the necessary and sufficient means of general intelligent action’, as expressed by the Physical Symbol System Hypothesis of Newell and Simon (1976). This view gave rise to many qualitative systems of cognition that solely consist of symbols and mechanisms that operate on these symbols independent of their content.

In general, a qualitative system capable of displaying cognitive behavior embeds a symbolic model of its environment and a symbolic specification of actions that it can perform. In addition, it should embed an inference mechanism. Qualitative systems frequently use condition-action rules, in which case they are referred to as rule-based systems, or a logic engine. Other examples of qualitative modeling approaches are causal networks whose connections denote causal relations in a binary fashion, and semantic networks, in which the connections between knowledge elements are labeled with meaningful symbols, e.g., isa, or has.

A specific subset of systems embedding qualitative elements are formed by systems whose symbols are labeled with the mentalistic notions *belief*, *desire* and *intention*. The idea that these three concepts suffice to generate intelligent action dates back to Aristotle (B.C. 350) and was revived by the work of Bratman (1987) on rational agents. For so-called BDI agents it holds, globally stated, that beliefs represent the agent's knowledge (its information state), while desires represent its goals (the agent's motivational state). What intentions exactly represent varies between approaches, but in general the agent's deliberation state. In the formal BDI-framework provided by Rao and Georgeff (1991) intentions are a separate notion which cannot be reduced to the concepts of beliefs and goals (desires), while Cohen and Levesque (1990) only take beliefs and goals as primitives and consider intentions to be a subset of the goals. The main stream of agent modelers uses intentions to express the agent's commitments to certain plans that will lead to achievement of the goals formed by its desires.

Using mentalist notions in agent systems has several advantages. First, the specific representation of beliefs and goals facilitates autonomous goal-directed behavior tailored to the current state of the world. Second, agents that are based on mentalist notions are intuitively understandable. Notions like belief, desire, and intention stem from folk psychology: they map easily to the language people use to describe their reasoning and actions in everyday conversation (Norling, 2004). For a panel discussion on the BDI model of agency, in specific on how it stands in relation to other qualitative models of agency and on its limitations to capture certain (human-like) types of behaviors, see Georgeff et al. (1999).

Quantitative Modeling Approaches

Previously, we stressed the benefit of hybrid cognitive models because although qualitative models are well-suited to model specific aspects of human behavior, they are less suited for others. In this section we elaborate on a subset of the quantitative models that have been proposed as alternatives for the qualitative models of cognition. For extensive discussions on many other mechanisms to implement reasoning on quantitative data, e.g., fuzzy logic, frequency probability, and utility theory, see Russell and Norvig (2003).

Connectionist Systems Connectionism is inspired by the structure of the brain, and views cognition as the emergence of global states in a network of simple components. Connectionism tries to model cognitive behaviors by interconnecting many simple processing elements, referred to as nodes, that can have a small local memory and form together with their connections a so-called Artificial Neural Network (ANN). When the network is active its nodes operate in parallel on their local data and the numerical (instead of symbolic) information that they receive through their connections. Within a connectionist model knowledge is distributed and resides in the weights of the connections between the simple units.

Various types of artificial neural networks can be distinguished based on their overall structure and way in which they are formed, see e.g. Maes (1994) and Browne and Sun (2001). One important aspect that ANNs can vary in is on whether they incorporate distributed or local representations. Local representation networks resemble the qualitative approach in that they use single nodes to represent symbols. However, the rules operating on these symbols are not represented symbolically like in causal or semantic networks, but formed by the numerical properties of the nodes and connections: activation thresholds and strengths respectively. Neural networks incorporating distributed representations do not capture meaning in one single node, but let it emerge from the interaction of multiple neurons. This makes those neural networks more biological plausible, but also more complex and harder to analyze than the local representation versions. Another drawback of such neural networks is that for forming ('training') them a large amount of training data, generally in the form of thousands of examples, is required.

In general, connectionist systems are formed by training them on a large amount of data using a variety of techniques (e.g., supervised, unsupervised, or reinforcement learning). However, this data might not always be available. An alternative approach to ANNs to deal with the lack or ambiguity of information as often encountered in the real world are Bayesian systems.

Bayesian systems A Bayesian system consists of a network whose nodes represent (any kind of) variables, and whose connections encode the causal dependencies among the variables; missing connections encode causal independencies between variables.The dependencies are quantified by conditional probabilities for each node given its parents in the network. Bayesian networks can be used to model a large number of cognitive processes: reasoning (using the Bayesian inference algorithm), learning (using the expectation maximization algorithm), planning (using decision networks) and perception (using dynamic Bayesian networks), see Russell and Norvig (2003).

Dynamical Systems Although connectionist and Bayesian systems are more biologically plausible than symbolic systems, they still comply to the view of cognition as an information processing system that converts a set of inputs into an set of outputs. In contrast, dynamical systems view cognition as a situated activity and model it as part of a system that also encompasses an agent's body and world (Beer, 2000).

Dynamical system theory states that 'Natural cognitive systems are certain kinds of dynamical systems, and are best understood from the perspective of dynamics' (Gelder, 1995). Dynamical Systems consider cognition to be a multiple-dimension space of thoughts and behaviors that is explored when thinking. For modeling cognition as a dynamical system it has to be described how this space of thoughts and behavior is explored under influence of external and internal pressures. This is usually done by differential equations. Knowledge is distributed within a dynamical system, namely over many different kinds of processes, each represented by a numerical equation.

Selected Modeling Approach

Most of the research on modeling synthetic entities for training simulations embed an approach that is mainly qualitative in nature, see Section 2.4.1. Reason for this is that qualitative models offer conceptual and practical benefits. First, 'mental states' are transparent using this approach, and it is easy to backtrack which part of the model is responsible for a certain behavior. Second, because the model is made up of discrete parts, it is easy to access and alter one part of the model without affecting the rest. Third, discrete parts facilitate reuse of parts of the model. However, qualitative models are not well suited for modeling some fundamental characteristics of cognition, like the ability to learn and to deal with uncertain or incomplete information[1]. To model such aspects, frequently quantitative elements are embedded within qualitative models. This endangers the benefits of the latter: its functioning becomes less transparent, and the knowledge acquired during execution and stored in quantitative terms is difficult to reuse.

Sloman (1997) argues that selecting the 'best' approach for modeling cognition for a specific task is a matter of analyzing trade-offs. We know that our goal is the development of cognitive models that enable agents to fulfill tasks within simulator training of military tasks currently fulfilled by humans. In Section 1.1.5, we denoted several requirements for these agents. Among others, they need to reduce costs of training, which makes it undesired that a new agent has to be developed from scratch for every scenario. Furthermore, training requires that the behavior of such agents is varied but tunable,

[1]Qualitative approaches towards these aspects do exist. For example, inductive logic programming is a type of machine-learning for logic programs, and uncertainty can qualitatively be denoted by, e.g., comparative probability, linguistic labels, or partial preference orderings (Parsons, 2001).

which requires a clear understanding of the source of the behavior. These requirements motivate us to model cognition based on qualitative representations. Another reason for this is that we want to base our models on the mentalistic notions of beliefs and goals, because of the benefits that mentalist notions bring to the modeling and understanding of agent behavior. However, we are also interested in the modeling of more human-like behavior, like the ability to deal with information with varying degrees of certainty, and the fact that humans almost never behave in the same way. We want to model variety in behavior due to the occurrence of cognitive biases that can influence cognitive processes to a smaller or larger extent. These aspects are hard to model following an approach that is purely qualitative in nature. Therefore, we also embed quantitative aspects in our models.

As an outlook: in Chapter 3 we attach a quantitative certainty value to beliefs of agents, and reason about the classification of a radar contact using a naive Bayesian classifier; in Chapter 4 we determine and use availability values of beliefs; and in Chapter 5 we reason about the relevancy (utility) of reasoning components, among others based on the agent's cognitive exhaustion level, a quantitative value.

2.3.2 Integrated Architectures

In the introduction chapter we quoted Lewis (2001) who states that 'cognitive architectures constitute a fixed set of processes, memories and control structures that define its underlying theory about human cognition'. This definition of cognitive architectures supports our previous remark that for implementing a cognitive agent model not only 'parts of cognition' need to be modeled, but also the way in which these interact, i.e., their control. Newell (1990) was among the first to stress the importance of studying the integrated architecture underlying cognition. He phrased this as the need for a unified theory of cognition: a single set of mechanisms that together account for all aspects of cognition, similar to the mind. A few years later Sloman (1997) expresses the same need, i.e., the need to study architectures to research how diverse but interrelated capabilities can be put together.

Fulfilling their call, many integrated architectures have been developed in the last two decades that facilitate the building of software agents. An agent is formed when task-specific declarative and procedural knowledge is added to the architecture that takes care of everything else, like the functioning of the agent's memory and how it decides on its actions. How this is done greatly varies between architectures. The work of Pew and Mavor (1998), later extended by Ritter et al. (2003), provides the most extensive and comprehensive overview and comparison of existing architectures for modeling human behavior to date. For another overview, see Morrison (2003).

This section is about architectures, to investigate whether existing architectures can be used to build software agents for our research objectives. We start with a general discussion on the suitability of various architectures to model cognition and human behavior. To increase the readability of that section, we do not elaborate on the architectures and software packages mentioned. Instead, Appendix A lists and shortly discusses the various architectures and languages referred to throughout this dissertation. Next, we elaborate in more detail on two cognitive architectures frequently used for the development of cognitive agents, Soar (Laird et al., 1987) and ACT-R (Anderson and Lebiere, 1998; Anderson et al., 2004). We discuss their applicability in general, as well as their relevance for our research objective. Hereby specific attention is paid to their ability to model cognitive biases.

Suitability of Architectures

In the beginning of this study we investigated several integrated architectures to gain insight in which cognitive architectures would be (most) relevant to the research questions posed. This turned out to be a hard question, because architectures are very diverse and comparing them is like comparing apples to pears. Simon (1997) notes about existing integrated architectures that they can be seen to be built around a core cognitive activity, which is then extended to handle other cognitive tasks, e.g., the core cognitive activity of ACT-R is semantic memory, in Soar the core is problem-solving.

From reviews on the modeling of human behavior for military simulations (Pew and Mavor, 1998; Banks and Stytz, 2003) it follows that there exists no consensus on the way, or architecture, that is best suited for this task. Van Lent et al. (2004) subscribe to this, as in their research three separate human behavior models are used to support three different roles for synthetic entities in a single simulated scenario. They select for each role an architecture suited to model the requirements posed for each role:

> "The Soar architecture, focusing on knowledge-based, goal-directed behavior, supports a fire team of U.S. Army Rangers. PMFServ, focusing on a physiologically/stress constrained model of decision-making based on emotional utility, supports civilians that may become hostile. Finally, AI.Implant, focusing on individual and crowd navigation, supports a small group of opposing militia."

Morgan et al. (2005) try to get insight into the strengths and weaknesses of architectures by the introduction of dTank. dTank is a competitive environment which can be used for architectural comparisons of competitive agents, and for comparisons of human and agent behavior. They used dTank to compare models built in Java, Jess and Soar. The development of dTank is continued (Ritter et al., 2007a) and additional models are developed, most recently a CoJACK model (Evertsz et al., 2008b). Morgan et al. (2005)

state that only few modelers have used the same environment to examine the behavior of humans as well as that of agents built in different architectures.

A noticeable exception is the Agent-based Modeling and Behavior Representation project (AMBR), initiated in 1999 by the Air Force Research laboratory (AFRL). AMBR focused on comparing models of complex human behavior. Gluck and Pew (2005) cover its methods and results. In short, four research groups were asked to develop models for the same two tasks. Each group used a different architecture, which resulted in a comparison between ACT-R, COGNET/iGEN, D-COG, and EPIC-Soar. The focus of the first experiment was modeling multi-tasking; the second was modeling category learning. The domain for both experiments was a simplified version of Air Traffic Control.

The conclusion of this project was that 'the quality/acceptability/appropriateness of a model, and any effort to rank order it relative to models developed with other architectures, depends on what one values in a model and an architecture.' Gluck and Pew mention several factors on which models can be evaluated and compared: goodness of fit of the model data with the human data; the degrees of freedom available in implementing the model; how much of the model was reused from previously implemented models; the interpretability of the model's behavior during run-time; and the generalization of the model. These factors are likely to vary between the implemented models, between the architectures used for this implementation, and even more general between the various approaches toward the modeling of cognition implemented in these architectures.

Therefore, it is difficult to select an architecture which is best suited for modeling cognition and thus human behavior. Fortunately, our goal is not to model cognition in all its facets, but to develop cognitive models for tactical training simulations that will enable software agents to play a role in a human-like way, and that aid in providing cognitive feedback. Model requirements for these objectives are listed in Section 1.1.5, which made us decide to model cognition using mentalistic notions and following a hybrid approach (Section 2.3.1). Below we introduce the cognitive architectures Soar and ACT-R, and discuss their applicability and relevancy for our research objectives.

Soar

Description The Soar architecture implements a qualitative approach toward the modeling of cognition (Laird et al., 1987), and is proposed by Newell (1990) as 'a candidate unified theory'.

Soar's basic assumption is that human behavior can be modeled as a set of parallel operating and interacting processors, to be precise by a cognitive, perceptual and motor processor. Soar's perceptual and motor processors provide the means of perceiving and acting upon an external world. In addition, Soar incorporates an unbounded work-

```
Soar
  while (HALT not true) Cycle;

Cycle
  InputPhase;
  ProposalPhase;
  DecisionPhase;
  ApplicationPhase;
  OutputPhase;


ProposalPhase
  while (some I-supported productions are waiting to fire or retract)
    FireNewlyMatchedProductions;
    RetractNewlyUnmatchedProductions;

DecisionPhase
  for (each state in the stack,
       starting with the top-level state)
  until (a new decision is reached)
    EvaluateOperatorPreferences; /* for the state being considered */
    if (one operator preferred after preference evaluation)
      SelectNewOperator;
    else                    /* could be no operator available or */
      CreateNewSubstate;    /* unable to decide between more than one */

ApplicationPhase
  while (some productions are waiting to fire or retract)
    FireNewlyMatchedProductions;
    RetractNewlyUnmatchedProductions;
```

Figure 2.1: A simplified version of the Soar Algorithm (Laird et al., 2006))

ing memory through which the three processors communicate. This working memory symbolically represents the current state of knowledge, which comprises the inputs of perception, the output parameters for the motor modules, and purely internal structure. These symbolic representations are called *working-memory elements*.

Soar assumes human behavior to be goal-oriented. It implements this as a search through interacting problem spaces, where each problem space contains a set of symbolic production rules called *operators* that are applied to states to produce new states. While searching, the model learns the results of its problem solving. A task, or goal, is modeled by the specification of an initial state and one or more desired states.

When the model is run, symbolic production rules called *productions* match the contents of working memory and can either directly retrieve additional information from long-term memory (automatic association; an unconscious reasoning step) or propose to apply a certain operator (embedding a conscious reasoning step). These productions match and fire in parallel, and therefore multiple operators might be proposed at the same time. However, only one operator can execute each cycle, but the rules embedded in an operator can fire simultaneously. Which operator is applied is determined by weighing the operators' *preferences* (qualitative labels denoting its priority), which can be added to operators by productions.

When sufficient knowledge is available in the problem space for a single operator to be selected and applied to the current state, the behavior of a Soar model is directed and smooth, as in skilled human behavior. Whenever knowledge is lacking and the active preferences are not sufficient to allow a unique choice, the architecture responds by setting a subtask to resolve this state called *impasse*, and the entire process recurs. For a simplified version of the Soar algorithm, see Figure 2.1.

If the recursive processing adds new preferences to operators that are active in the original task, Soar's learning mechanism can create a new association between those working-memory structures in the original task that led, through a chain of associations in the subtask, to the new preferences in the original task. This learning mechanism is called *chunking* and can be turned on or off.

The initial knowledge of a Soar model has to be encoded by the human modeler in symbolic production rules. The initial state of the model is often coded manually, but a model can also start with no knowledge of the problem state and just acquire that knowledge from the external environment.

Applicability Soar has been evaluated for its suitability to model human behavior for a wide variety of tasks, among others military. It is frequently used to develop agents that act as (large) expert systems. For this, knowledge off-line encoded in Soar is used on-line by the agent to decide on its actions and to solve problems.

A well-known example of such an effort is TacAir-Soar (Jones et al., 1999), which is a fully autonomous Soar system that embeds behaviors from the tactical air domain. TacAir-Soar's knowledge base consists of over 7500 rules, which provide agents with behaviors for a wide range of simulated aircraft and missions. The agents, once briefed, can autonomously plan and execute their missions using US doctrine and tactics. TacAir-Soar can be used to model friendly, opponent, as neutral forces in simulated environments.

Soar has also been used to model opponents (Wray et al., 2002, 2004) and subordinate team members (Van Lent et al., 2004) as part of the Virtual Training & Environments (VIRTE) program of the American Office of Naval Research (ONR). This program focuses on developing virtual trainers for Military Operations on Urbanized Terrain (MOUT). For this program synthetic entities were modeled as either opponents or members of a fire team consisting of four U.S. Marines. This fire team is situated in a virtual urban environment and has the task to clear a building that possibly contains enemy soldiers. The environment and synthetic entities were developed using the Unreal Tournament game environment. Wray et al. (2002) list five high level requirements for the intelligent opponents of the fire team in the domain described: Observational fidelity; Faster than real-time performance; Behavior variability; Transparency; and Rapid, low-cost development. Later on Wray et al. (2004) extend this list with the requirements of 'Competence' and 'Taskability', and transform 'Faster than real-time performance' to 'Minimal computational footprint'. Most of these requirements have also been listed in Section 1.1.5 as required for the cognitive software agents that are to play a role within a simulated training environment. Wray et al. (2004) conclude their paper with the statement: 'Achieving intelligent opponents that are fully autonomous, believable, and interactive, while exhibiting a variety of behaviors in similar circumstances, will require much additional research and development.' More specific, they mention that 'less' competent behaviors (possibly representing poorly trained guerrilla's or novices), and additional errors in judgment and perception could be modeled.

The team members modeled by Van Lent et al. (2004) are of less interest to our research objectives than the opponents, since these team members were not autonomous; a human player directed their mission-specific behavior by issuing specific commands.

Relevancy The Soar architecture is interesting for our goal of modeling cognitive agents for training simulations, since it is shown to be sufficiently powerful to generate believable individual human behavior in military situations. In addition, it is sufficiently robust to run large-scale models connected to real-world systems in real time. However, most Soar models are developed to display only one type of behavior, namely expert behavior.

Randolph M. Jones (SoarTech, private conversation at ICCM 2007) explained that it is hard to have the TacAir-Soar agents display wrong behavior. In one project the goal was to train oil rig operators in handling pilots that fly in too steep. This training should take place in a simulated environment with TacAir-Soar agents controlling these pilots. Because TacAir-Soar agents normally display correct behavior, their procedure on approaching a platform had to be altered so they would fly in too steep. Unfortunately, this did not work, because as soon as the behavior was obviously false, e.g., they would fly too quick for their height, other behavior rules would take over and correct the behavior.

So, although the Soar architecture looks promising for generating human behavior, it remains to be seen how suited it is for displaying behavior at a variety of performance levels. One reason why this may be hard is that Soar does not embed the cognitive limitations found in humans, which possibly give rise to cognitive biases. Most obviously, it embeds an unbounded working memory while it is uncontentious that human working memory is limited.

Interestingly, research has been done on embedding performance moderators in Soar. Chong (1999) models fear in Soar. Unfortunately, this is not a dynamical model and thus does not allow for changes in this moderator value: the models start and stay fearful. Gratch and Marsella (2001) extend Soar with a model of emotional appraisal called Émile. Émile embeds mechanisms for evaluating how environmental events relate to an agent's plans and goals, and how these evaluations give rise to emotions. However, the effects of these emotions on behaviors are left to be determined by the modeler. In more recent work, Marinier and Laird (2007) incorporated a computational model of emotion, mood, and feeling in Soar, and demonstrate that such a model can help direct problem solving, and speeds reinforcement learning.

Although all these models are interesting for the purpose of developing software agents that show more human-like behavior, none of them is tailored to model the occurrence of cognitive biases. The modeling of more human-like behavior due to the incorporation of emotions is not one of the topics of this dissertation. This study attempts to model more human-like behavior by incorporating cognitive models that show varied behavior, with (part of) this variety stemming from the occurrence of biases.

ACT-R

Description The ACT-R architecture implements a hybrid approach toward the modeling of cognition (Anderson and Lebiere, 1998). ACT-R constitutes a unified theory of cognition and is based on detailed findings concerning the functioning of human memory and of learning and problem-solving processes.

ACT-R incorporates a declarative and a procedural memory module, as well as a perceptual (visual) and motor (manual) module that provide the means for perceiving and interacting with an external world. All modules, except the procedural memory module, are accessed through a specific buffer. Buffers represent a strictly limited working memory: each buffer can hold one element at maximum. Working memory is not separated from long term memory, but is the portion of declarative knowledge that is currently active, and represents the current state. Declarative knowledge is represented in ACT-R as *chunks* of information; procedural knowledge is represented by if-then rules called *productions*. For a visual overview of the ACT-R architecture, see Figure 2.2.

ACT-R's qualitative nature is represented by the production system it embeds, its quantitative nature by a set of massively parallel processes that can be summarized by a number of mathematical equations. These subsymbolic equations control many of the symbolic processes.

Models developed in ACT-R are goal-directed. Its processing consists of moving from an initial state to a specified goal state, which is encoded as a special type of declar-

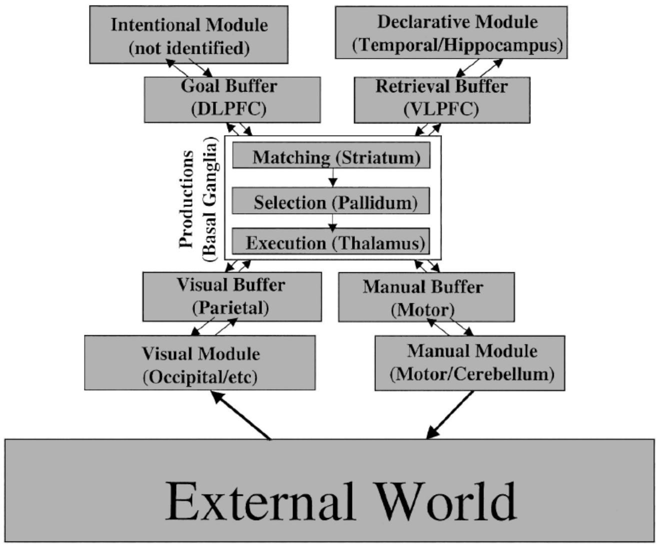

Figure 2.2: The organization of information in ACT-R 5.0 (Anderson et al., 2004))

ative memory element. This movement takes place when productions fire. A production responds to some goal, can retrieve information from declarative memory, and possibly takes some action or sets a sub-goal. Created sub-goals represent intermediate steps toward the end goal, form a hierarchy, and have to be satisfied in a bottom-up manner.

A maximum of one production can fire at a single time, so when the conditions of more rules are met, a conflict resolution mechanism determines which one fires. It does this using the rules' expected gain value, determined by a quantitative utility equation, and based on the rules' probability of success, their costs, and the current goal's value.

Initial knowledge, both declarative and procedural, is hand-coded by the human modeler. Also many values have to be hand-coded, such as the initial numerical parameters for the strength of productions, the cost and probability of success of these productions with respect to a goal, and the base-level activation of declarative knowledge structures. However, ACT-R assumes several learning mechanisms that tune these values in run-time. For example, the activation level of declarative knowledge elements is raised the more the element is used. Similarly, the expected gain value of productions is heightened the more often the rule is successful, e.g., when its sub-goal is achieved.

Applicability ACT-R models have been developed and validated against human data for a large number of tasks. Most of these efforts concern the modeling of memory tasks or small problem-solving tasks, like the Towers of Hanoi (Gunzelmann and Anderson, 2001). For these tasks it is shown repeatedly that ACT-R is capable of matching human performance data, including cognitive biases displayed by people. For example, Anderson et al. (1998) show that the ACT-R theory accounts for a wide variety of list memory paradigms, including the recency and primacy effect.

In recent years, ACT-R has also been used to model human behavior in military simulations. For example, Best et al. (2002) developed as part of the Virtual Training & Environments (VIRTE) program ACT-R models that control opposing and attacking forces in the MOUT domain. Following this exercise Best and Lebiere (2006) describe a general approach for developing ACT-R agents that have to interact in a three-dimensional space in real-time with humans and each other. Another military task was modeled by Anderson et al. (2004). They describe how the data of an ACT-R model practicing a simplified Anti-Air Warfare Coordinator task strikingly matches the learning and performance curve of humans. Juarez-Espinosa and Gonzalez (2004) discuss an ACT-R model of situational awareness (SA) for military command and control that can reproduce a commander's behavior. Although their model is very simple and supported by multiple information pre-processing agents, they do manage to show a relation between the modeled commander's memory, displaying omission errors, and SA performance.

Concerning the modeling of errors with ACT-R, Byrne and Kirlik (2005) model the taxiing behavior of pilots from commercial airlines, and show that a variety of decision-related errors can be correctly modeled. For this, their model assumes that when time is short, pilots use heuristics. More precise, in these circumstances the pilot models tend to rely on computationally cheaper, but less specific, information gained from experience with a wider class of situations, of which their current situation is an instance. This work resembles in some aspects the work of Schooler and Hertwig (2005) who also focus on the modeling of heuristics within ACT-R. Fotta et al. (2005) used the ACT-R architecture as basis to build their Human Error Modeling Architecture on, and provide a clear overview of the types of errors that the mechanisms embedded in ACT-R can produce. Also Lebiere et al. (1994) focus on error modeling in ACT-R and show that omission and commission errors can satisfactory be modeled.

Besides the modeling of human behavior for synthetic entities, ACT-R has also been used to successfully model intelligent tutors teaching mathematics and computer programming in high schools, see (Carnegie Learning Inc., 2009). These, and other intelligent tutors, are further discussed in Section 2.4.2.

Relevancy The ACT-R architecture is interesting for our goal of modeling cognitive agents for training simulations. First, ACT-R has been shown to be sufficiently powerful to represent individual human behavior in military situations. Although it must be noted that in all the mentioned research projects the agents simply performed doctrine behavior, following a fixed set of rules. For example, the only variability found in the behavior rules of the attacking forces modeled by Best et al. (2002) was whether they, after entering a room, would clear the area to the right or left. Therefore, it remains to be seen how suited ACT-R is for the modeling of varied behavior as required for training (Section 1.1.5).

A second reason why ACT-R is interesting for our research objective, is that it has been extensively demonstrated that the ACT-R mechanisms give rise to many typical cognitive biases and errors. Most of the research mentioned shows that specific (phenotype) errors can be explained by the specific mechanisms (error genotype) embedded in ACT-R. An exception on this is the research on errors stemming from the use of heuristics, which have to be hand-modeled by the agent designer. In the latter case an additional mechanism is required that controls when these heuristics are used. But also in the former case it should be so that under specific circumstances of stress and fatigue the amount of errors increases. To model these kinds of aspects it is required that behavior moderators are incorporated within the ACT-R framework. Ritter et al. (2007b) discuss six theories of stress and show how four of these can be incorporated in ACT-R as overlays. An overlay is defined as 'an adjustment or set of adjustments to the parameters or mechanism that influence all models implemented in the architecture to reflect changes due to an altered mental state or due to long term changes such as development.' The two other theories were not possible to model as overlays, but require a more fundamental change in the architecture. One of the theories, named 'the task as stressor' has been modeled as part of this study, see Section 5.2. The modeling of fatigue within ACT-R is done by Gunzelmann et al. (2007). They incorporate fatigue as an overlay and show that its effects on performance in a sustained attentional task resembles human data.

Because ACT-R is a hybrid architecture, it remains to be seen how transparent the functioning of the models developed in ACT-R is. Related to this is the question how reusable the knowledge embedded in the developed models is.

To get hands-on experience with the two architectures described in this section, and to develop a sense for the kind of tasks they are suited for, we have developed cognitive software agents in ACT-R and Soar, in addition to cognitive agent models implemented in Swi-Prolog (Wielemaker, 2003) and LeadsTo (Bosse et al., 2007).

2.4 Applications of Cognitive Models

In this section we first discuss four application domains in which cognitive models can be applied to model human-like behavior. Next, we elaborate on the generation of feedback on task performance; also in this process cognitive models can play a role.

2.4.1 Human Behavior Models for Simulated Environments

Much of the research discussed so far focuses on the modeling of cognition or human behavior from the perspective of natural science, i.e., to understand it better. In contrast, here we describe four distinct research areas that develop software agents for the purpose of generating human-like behavior for specific tasks, so from a design science perspective. This division in four areas is adopted from Ritter et al. (2003) who describe four main application areas for human behavioral models: *education and training*, *entertainment*, *automated support*, and *systems analysis*. For each area we examine to what extent the general requirements of the behavior of the developed agents match with the requirements of the behavior of an agent for our goal, which is the ability to display human-like behavior at a variety of performance levels: from rational to biased.

Training and Simulation

Nowadays, a large part of military training takes places in simulated environments, for the reasons mentioned in Section 1.1. Tasks that are trained range from low-level tasks such as shooting at targets, to high-level tasks such as commanding a complex military operation. Setting up training for the last type requires many people, in particular if every entity in the training environment has to be controlled by a human. Luckily, this is often not required. In general soldiers do not operate alone, but as a team. Such a team is commanded by a single entity. When this entity is human, the members of such teams can be simulated without losing behavioral validity, since this is preserved by the human commander. Such simulated entities are generally called Computer Generated Forces (CGFs). CGFs that can be controlled by a human player are also referred to as Semi-Automated Forces (SAFs). For an overview of specific programs that are developed to create synthetic forces, see Table 2.2 in Pew and Mavor (1998). Well known are the packages ModSaf and its follow-up OneSaf. In addition, some military training systems, e.g., Virtual BattleSpace 2 (VBS2), also allow the programming of their virtual entities.

Because the human decides about the actions of the simulated entities, CGFs do, in general, not posses much intelligence or reasoning capability. This is fine for most tasks, e.g., walking from point A to point B can be done without much intelligence in most cases. However, their behavior might turn awkward due to their limited intelligence. A case in point is when there lays a minefield between A and B. No human soldier, even when ordered, would keep walking forward when all its team members get blown up before him. Unfortunately, CGFs are likely to do so. The limits in intelligence in CGFs becomes particular prevalent when their behavior is set by a human modeler before the training scenario starts, and is not controlled during training. Unfortunately, this is exactly the situation of much training nowadays due to limited personnel.

Anja van der Hulst (TNO Defence, Security and Safety, personal conversation) trains military personnel for foreign peace-keeping missions by running them through a series of tactical scenarios developed in VBS2. This software package can simulate a variety of environments as well as human entities, and offers a scripting language to model behavior for these simulated human entities using basic, pre-defined behaviors. Unfortunately, it is proven time after time that the intelligence incorporated in these basic behaviors is not sufficient to create human-like behavior, and therefore to create a realistic training environment with according experiences. Van der Hulst discusses two cases in point, emerging from the basic behavior incorporated in CGFs to 'take cover when fired at'. The first is that these CGFs will, e.g., when fired at in the middle of the desert, run for miles and miles until this cover is reached. The second that 'taking cover' is also implemented as standing with 20 other CGF's in a market stall consisting of a cotton cloth, given that the latter is an 'object'.

These two examples highlight a critical aspect for the generation of human-like behavior in simulated environments; the correct representation of knowledge within the agent (first example) as well as in the simulated task environment (second example). The second example also touches upon the topic of whether all required knowledge should be formed by the agent on-line, or whether part of it can be put in the environment off-line. In the first case the agent should be able to conclude, e.g., that a cotton cloth is not appropriate to take cover, for which it has to have knowledge about the properties of cotton and the ones required to be a suitable cover. In the second case it is predetermined that a cotton cloth is not suitable for taking cover, and therefore the environment does not tag this object with the property 'cover against fire'.

The finding that CGFs are not able to display human-like behavior is not new. Sandercock (2004) identifies, using a Turing Test, several areas in which CGFs consistently show weaknesses when compared to human players. It is found that current CGFs perform weakly on: environment awareness, human variance, persistence, vengeance, anticipation, learning, and teaming.

Relevancy Current CGFs are in general not yet capable of displaying human-like behavior that is required for realistic tactical training, which includes capabilities such as situational awareness and anticipation. They are unable to display rational behavior, let alone more biased behavior. The military is interested in synthetic entities that do not display the mentioned weaknesses of current CGFs. Our proposal to get there is embedding valid human behavioral or cognitive models in the software agent controlling these entities.

Several other researchers have attempted to do the same. Section 2.3.2 described attempts to develop cognitive models that could control synthetic entities in Soar (Jones et al., 1999; Wray et al., 2002, 2004; Van Lent et al., 2004) and ACT-R (Best et al., 2002; Juarez-Espinosa and Gonzalez, 2004). In addition, Van Lent et al. (2004) developed human behavioral models in PMFServ. The development of JACK was spurred because of the necessity to model teams of BDI intelligent software agents for military training simulations (Lucas and Goss, 1999).

For a more general survey on the technologies and progress made in the construction and use of software agents controlling synthetic entities see (Stytz and Banks, 2003a,b; Banks and Stytz, 2003). In general it can be stated that there does not yet exists a satisficing solution for developing intelligent behavior for training simulations that meets our requirements.

Entertainment

In the last two decades the entertainment industry has put much time and effort in developing intelligent software agents for simulated environments: in particular for First Player Shooters (FPSs). FPSs are video games in which the player's on-screen view of the game world simulates that of a character. FPSs generally involve an avatar, one or more ranged weapons, and a varying number of enemies (Rollings and Adams, 2003). These enemies can be played by other humans, for example in on-line multi-player games, but are usually played by bots. Bots are pieces of software that control simulated characters in a game environment. Bots are programmed using techniques from so-called game Artificial Intelligence, which is a subset of common, academic Artificial Intelligence (AI). Typical examples of game AI are collision detecting techniques, and path finding and path planning algorithms such as A* (Hart et al., 1968). In general, the reasoning of game bots is scripted or limited to a few simple heuristics.

Dependent on the level of the human player, bots are easier or harder to beat. In any case it holds that certain abilities of bots, such as a perfect hit rate, should be toned down to give the human player a feeling of fairness. Laird and Duchi (2000) conclude from their attempt to model a human-like Quake-bot using Soar that *firing accuracy* and *movement speed* are significant determinators for the believability of the bot.

Although the firing accuracy level is constrained to preserve believability, it can still have different values. As such, it is possible to generate behavior at different competence levels. Unfortunately, these variations often do not occur within play, but are set beforehand. When bots do alter their competence level within play, this is not due to human-like aspects as stress or exhaustion. They alter their level to match the level of the human player, to increase the fun of the game (Scott, 2002).

Relevancy Many techniques used in game AI might be useful for our purpose. However, reuse is not without danger, since bots are developed for simulations that have as their primary function entertainment, and not training. For game AI it holds (Tozour, 2002):

> "Our field requires us to design agents that produce appropriate behaviors in a given context, but the adaptability of human-like 'intelligence' is not always necessary to produce the appropriate behaviors, nor is it always desirable".

For example, hacks and cheats are generally accepted as long as they increase game play, and are not obvious (Scott, 2002). An example of a common cheat is that bots know where the player is, although they have not observed that. Such cheats can easily cause unrealistic behavior, which is fine for games as long as it enhances the fun of playing. However, such behaviors might not always be appropriate for training simulations. For example, when a trainee has successfully crept up on an enemy, it is undesired that this behavior goes unrewarded because the enemy knows where the trainee is anyway.

For training simulations the main goal of modeling human-like behavior is not to increase the *fun* of training, but to increase the *realism*. It is doubtful whether knowledge concerning the modeling of human-like behavior in games can aid this increase in realism. For example, game AI has not been been used to model biased behavior. For a survey of commercial game technologies related to behavior modeling and their relevance for military simulations, see Diller et al. (2004).

Automated Support

The increase in system technologies over the last decades has resulted in an increase of information that is available at any time to an operator of such a system. Unfortunately, more is not always better. Too much information can lead to a state of *information overload*. On top of this, information might be incomplete or uncertain, which further complicates the decision-making process. As such, there has been an increasing demand for automatic support of system operators.

Systems capable of generating support are usually referred to as intelligent decision-support systems. The support they provide varies: some take over the task, others prepare materials or information, yet others modify the display to help distinguish between alternatives, or to make performing actions easier. Some of this support can be delivered by a software agent in the form of an intelligent assistant. An example of such an intelligent assistant developed using the COGNET architecture is described by Weiland et al. (1998). Their Naval Surface Fire Assistant helps a human gun commander to utilize a gun system, and is understandable to its human supervisor.

For generating support, it is minimally required that the intelligent decision-support system incorporates an *expert model* of the task to ensure that its recommendations are justified and useful. However, by itself, this is not sufficient. Current decision-support systems are often not used efficiently, because the operators lack confidence in the recommendations of the system (Moulin et al., 2002). To increase confidence, the system should be able to convince its user that its suggestion is justified and useful. For this it is important that the system incorporates a *user model*. A user model can be defined as 'a model that incorporates information about a system user to provide services that supports its demands' (McTear, 1993). Neerincx (2007) states that intelligent decision-support systems for complex, military tasks have to acquire and maintain knowledge of the cognitive and affective load of the tasks and situation, and the capacities of the user to cope with this load. By means of cognitive and affective load models adequate decision-support can be given that supports shared situation awareness, trust and scrutability.

Relevancy Expert and user modeling is relevant for training systems: feedback that is provided to the user of a training system (a student) should be based on an expert model, and tailored to the specific user. More specific, for adapting instructions and feedback to the individual student, a *student-model* that incorporates the knowledge of the student is required.

Therefore, knowledge developed within the field of intelligent decision-support systems concerning the modeling of expert and user models is relevant to our research objective. However, this knowledge might not be directly applicable and reusable. Supporting decisions in minimal time might require a different type of expert and user model than task training. In addition, the development of users models for decision-support is a younger research area than the development of user models for training, which makes it more likely that the former can learn from the latter. In Section 2.4.2 we elaborate on methods with which expert and student models can be formed that are developed within the research field of intelligent tutoring systems.

Systems Analysis

A last field that realistically models human task performance in simulated environments is engineering. The field of engineering has created multiple models of users of their systems, often on a complex operator level. These simulated operators are used to evaluate proposed human-machine designs and operator procedures.

Several general engineering-based architectures for the modeling of human operators have been developed, see Pew and Mavor (1998) for an overview. In general the main focus of these engineering-based architectures is the modeling of the human-machine interaction part. Therefore, they are more concerned with modeling the psychomotor tasks of humans than with modeling cognitive processes. The main reason why it is possible for the model developers to abstract from internal cognitive mechanisms is that the tasks for which they model operators are commonly well-defined, procedural tasks.

Nowadays, this field starts to realize the potential of using cognitive models to evaluate systems for a wider range of tasks than currently looked at, i.e., ill-defined, complex tasks instead of well-defined, procedural tasks. For a paper promoting the synergy of research into cognitive modeling and human-machine interaction, especially concerning the evaluation of user interfaces, see Ritter et al. (2001). Two examples of such research efforts are described by Amant et al. (2007) and Ryder et al. (1998). Amant et al. describe a cognitive model developed in ACT-R for the evaluation of cell phone menu interfaces, while Ryder et al. describe a cognitive model developed in COGNET for the evaluation of an integrated telephone services workstation interface.

Relevancy Human-like operators developed for system analysis could in principle be used to play the operator role in a training simulation. Unfortunately, these models are often task specific and connected to a simulation of the domain they are developed for. This makes it hard to gather which aspects are relevant and reusable for our domain.

In addition, the developers of operator models are in general more interested in a model that is usable and approximately correct, than in modeling the detailed internal mechanisms giving rise to behavior (Ritter et al., 2003). A case in point is that specific operator models, and engineering-based architectures in general, do not model any type of errors.

Moreover, current operator models are often developed for tasks that are not at the level of complexity sought after. The knowledge developed in this field concerning the modeling of expert behavior for well-defined, procedural tasks is likely to be reusable for training of such tasks. But for modeling operators in complex, open environments that perform tactical tasks and make realistic mistakes, the field of systems analysis does not offer a solution.

2.4.2 Feedback Generation for Simulated Environments

Previously, we pointed out that an important factor in task training is the generation of feedback on the student's task performance (Bosch and Riemersma, 2004). Training opportunities would increase significantly when this feedback could be generated by a system instead of a human. Automatic generation of feedback is one of the focuses of the research field of intelligent tutoring systems (ITSs), see e.g. Polson and Richardson

(1988). Other focuses are the automatic construction of an individual-tailored curriculum, and the ability to answer questions about the exercises or domain in general. For a clear overview of all the capabilities an ITS can possess, see VanLehn (2006).

A system capable of generating feedback should foremost be able to diagnose the task performance of a student. This entails assessing the actions of the students, and deciding whether or not they are appropriate for the current task. Moreover, the system should be able to provide the student with a motivation for its diagnosis.

In this section we aim to shed light on the current state-of-the-art concerning feedback-generating systems, and on the relevancy of that work for generating feedback on open complex tasks in military simulations. After distinguishing various types of feedback, we discuss which feedback type is required, or suitable for, which task type. Subsequently, we elaborate on the models and techniques that are currently used by ITSs to diagnose a student's task performance and to generate appropriate feedback.

Feedback and Task Types

Three types of feedback can be distinguished that are relevant in the context of simulation-based training systems (Mioch et al., 2007). They differ in the types of information that they take into account, and in the level of sophistication of the feedback they generate.

- **Result-based feedback**: This type of feedback is based solely on the result of the task behavior of the student. Feedback is generated by comparing this result with the correct result, which is often hard-coded and formulated beforehand by domain experts. The feedback states only whether the student has completed the task successfully, and if not, which result would have been correct.

- **Model-based feedback**: This type of feedback is not only based on the result of the student's behavior, but also on contextual knowledge of the simulation environment and explicit task knowledge. Feedback is generated by reasoning about the result of the student and why it was good or false, for which it uses an expert model and the task circumstances.

- **Cognition-based feedback**: This type of feedback is also based on the student's result, the context of the task, and explicit task knowledge, but additionally takes a student model into account. This student model tracks behavior of the student over time and makes it possible to infer the cognitive strategies of the student. Using this extra knowledge, feedback can be generated not only on the final result of the student's behavior, e.g., the selected action, but also on the process, e.g., how he or she selected this action.

These types of feedback can be further illustrated by an example from a traditional domain of feedback-offering training systems: mathematics. In mathematics, tasks have to be trained for which the interpretation of the result is deterministic: a result is either correct or false. An example of such a task is addition. The correct answer for adding 9 to 32 would be 41. In case a student arrives at the result of 31, the *result-based feedback* would entail something like: 'No, your answer is wrong, the correct answer is 41.' The *model-based feedback* would for example be: 'No, your answer is wrong. The correct answer is 41. You calculate this by first adding 2 to 9 which gives you 11, after which you note down 1 and subsequently add the left over 1 to 3, which gives you 4.' On the other hand the *cognition-based feedback* might entail: 'No, your answer is wrong. The correct answer is 41. After adding the 2 to the 9 and noting down 1, you probably forgot to carry over the remaining 1 and add it to the 3.'

Another division of feedback types is given by VanLehn (2006), who distinguished two types: *minimal* and *error-specific* feedback. Minimal feedback is another word for result-based feedback, while error-specific feedback refers to both model-based and cognition-based feedback.

Most ITSs have been developed for the training of tasks in well-structured and small domains, like the addition example above. Such domains have little indeterminacy and are relatively closed. Therefore, they can be represented by a small number of rules. Moreover, the tasks that are trained are often individual, procedural and static. For these types of tasks and domains it is relatively easy to deduce all three types of feedback, although in general the feedback is limited to result- or model-based feedback. Feedback can be based on expert knowledge in the form of rules or constraints that can unambiguously, due to the task's nature, determine the correctness of certain actions in certain states.

More challenging is the generation of feedback on student behavior in simulated environments. The types of tasks that are commonly trained in simulation-based training systems share few of the properties mentioned above. In general, these tasks are dynamic, open, and complex, since they take place in the (simulated) world which shares these properties. This entails that the task domain cannot be represented by a small number of rules. A task is called *dynamic* when it takes place in an environment that changes constantly and independent of the task performer. It is called *open* when multiple correct options exist to reach its goal. It is called *complex* when in every situation multiple actions are possible, that lead to new situations in which again several actions are possible. The dynamic nature of a task makes that for evaluating a student's performance not only its actions, but also the timing of its actions needs to be taken into account. The open and complex nature of a task makes that no single correct behavior exists at any moment, which makes performance diagnosis hard. Moreover, the correctness of a certain action often depends on the correctness of the actions that follow it. In other words, on the strategy that is followed by a student.

ITSs that have been developed for training dynamic, open, and complex tasks generally limit themselves to providing model-based feedback (Stottler and Vinkavich, 2000). However, it would be beneficial if feedback is provided at the level of cognition, e.g., on the appropriateness of the followed strategy. For this, cognitive models can be used.

Ryder et al. (2000) compare several uses of cognitive models in ITSs, and focus on expert performance models and instructional agents in particular. The models they review are all based on the cognitive modeling framework COGNET and provide feedback at various levels. For example, the described Advanced Embedded Training System (AETS) (Zachary et al., 1998) provides model-based feedback, while the feedback provided by the instructional agent embedded in their Electronically Assisted Ground Based Learning Environment (EAGLE) is sometimes cognition-based.

Other well-known research on cognitive models for training cognitive tasks is performed by Anderson and colleagues (Anderson et al., 1990, 1995). They successfully implemented several *cognitive tutors* that were developed around task-specific cognitive models in the form of production system models in ACT. The cognitive task models and thus tutors were developed for were well-defined tasks, namely LISP programming, high-school geometry, algebraic manipulation, and word problems.

Required Models

The generation of feedback to a student on his or her task performance has much in common with the generation of support to a user on the basis of his or her performance. For this, it is first of all important to incorporate an accurate and complete task knowledge model, i.e., a so-called expert model. Moreover, it is important to tailor the feedback to the user, for which a student model is required. Last, an instructional, also called pedagogical, model should be incorporated to determine the form of the feedback (Polson and Richardson, 1988).

Expert Model Anderson (1988) identifies three approaches to encoding expert knowledge. The first is independent of the way it is present in human intelligence. For example, in some domains it is possible to build a mathematical model whose subsymbolic equations deliver the same results as reached by humans after symbolic reasoning. The second is by building a standard (rule-based) expert system. The knowledge is extracted from a human expert, but the representation and the way it is applied does not necessarily have to correspond to the way it is represented and applied by humans. The third possibility is to build a cognitive model that simulates the way humans use and apply knowledge. On the whole, it is important that the feedback generating system incorporates a knowledge representation that facilitates the abstraction from low-level knowledge to high-level knowledge, since the latter is often required for feedback (Gomboc et al., 2005).

Which specific approach to encoding expert knowledge should be followed depends on the type of feedback that has to be based on it, and the method that is used to generate this feedback. For generating result-based feedback the ‘final’ observable result of a student’s task behavior has to be compared with that of the expert. How this expert result is determined is unimportant, so the first method to encode expert knowledge suffices. For generating model-based feedback, task knowledge and contextual circumstances have to be taken into account in addition to the final result. This can be done when expert knowledge is encoded following the second approach. The rules of the expert model embed the task knowledge, and give insight in the relationship between the current context and the correctness of the result of the student’s behavior. For generating cognition-based feedback, the cognitive strategies followed by the student have to be addressed as well. In order to do so, expert knowledge should be encoded as a cognitive model, because such a model provides the required insights in the cognitive reasoning processes for generating this type of feedback.

Student Model Feedback should not be based solely on the expert model, but should be adapted to the individual knowledge state of the student. Therefore, the feedback generating system needs to keep a model of the student’s knowledge, ideas, and beliefs. This model should be dynamic since these aspects change constantly during task execution.

VanLehn (1988) introduces two types of student models. The first are student models that can only represent which task conceptions the student is missing. Such models are also called overlay models, since they represent the overlay of the student’s knowledge with the domain knowledge, i.e., it models the student’s knowledge as a subset of the domain knowledge. A system taking this approach requires a knowledge base that contains the domain knowledge, i.e., the facts, rules, and competences that the student has to learn, and the ability to mark the content that the student has mastered. The second type of student models are those that can represent besides missing conceptions also the student’s misconceptions. The way in which these can be diagnosed is the topic of the next section, which describes several diagnostic techniques.

An important factor for diagnosing a student’s current knowledge-state is the level of input that is available to the diagnostic module, also called the *bandwidth* of the information (VanLehn, 1988). Broadly stated, three categories of information bandwidth exist. The lowest category is **final states**. This category is applicable when the diagnostic module only has access to the final state of the task execution process, i.e., its result. The second category is **intermediate states**, which is applicable when besides the result of the task execution process also intermediate steps toward reaching this result are available. The third category is (approximate) **mental states**. For this category to apply, the diagnostic module should know about the mental states the student traverses while performing the task. These states are obviously not directly observable, so should be inferred. The derivation of a student's mental state from his or her behavior is called *cognitive diagnosis*.

Techniques for Diagnosing Student Performance

For a detailed overview of nine diagnostic techniques, each of them useful for a specific type of information bandwidth system, see VanLehn (1988). Here we elaborate on the two techniques that have surfaced in the last decades for generating cognition-based feedback, as this is the type of feedback that is required for open, complex military tasks. These two techniques are **model-tracing** and **constraint-based modeling**.

Model-Tracing Model-tracing is a technique that follows a *process-centric* approach. The diagnostic module tries to infer the (erroneous) process by which a student arrived at a solution, and bases its feedback on this. Famous examples of ITSs that took a model-tracing approach are the cognitive tutors developed at the Pittsburgh Advanced Cognitive Tutor Center at Carnegie Mellon University (see, e.g., Anderson et al., 1990, 1995). This group also developed Cognitive Tutor Authoring Tools (CTAT) that assist in the creation and delivery of ITSs based on model tracing (Aleven et al., 2005). Zachary et al. (1999) also followed this approach for their Advanced Embedded Training System (AETS).

An ITS that embeds the model-tracing approach incorporates an executable cognitive expert model that possesses correct declarative and procedural domain knowledge. Using this knowledge, the system can simulate adequate task behavior. In addition, the system incorporates false domain knowledge, which can simulate typical mistakes made by students. Moreover, the ITS has the capability to trace the student's behavior step-by-step, and to compare these steps with the correct and false steps that are dynamically generated by the expert rules and so-called buggy rules, respectively. Feedback on the student's process can be generated at every step, based on the rule with whose result the student's behavior matches.

The main advantage of model-tracing is that following this technique, error-specific feedback can be generated. Another advantage is its modularity: buggy rules are easily added. However, relying on buggy rules to generate feedback also has several disadvantages. First, feedback can only be given when the student's behavior matches that of one of the modeled rules. Hence, it is important that all possible errors are represented by buggy rules, which can be a hard and time-consuming effort. In addition, it is possible that the student's behavior matches that of multiple rules. In that case, it is a complex task to ascertain what the student's knowledge state is.

Research, among others the projects mentioned above, has shown that the model-tracing methodology of following a student's behavior step-by-step, possibly generating feedback at every step, is applicable for well-structured, procedural tasks. However, whether this is also the case for open, complex tasks is yet unclear.

Constraint-Based Modeling Constraint-based modeling is a technique that follows a *result-centric* approach. The diagnostic module bases its feedback solely on the final-state the student arrived at, independent of the process that led him or her there. Nonetheless, this feedback is different from result-based feedback: the diagnosis does not only say whether this final-state is correct or what the final-state should have been, it also gives feedback on the type of error the student has made.

Constraint-based modeling is based on the idea that students learn from performance errors, and that those errors are the result of declarative knowledge that has not yet been internalized by the student. This approach assumes that correct solutions all satisfy the general principles of the domain, which can be described by *constraints*. When the student's final state violates one these constraints, this signals an error. The violated constraint gives direction to the incomplete or incorrect knowledge the student has, which can be used as basis for feedback.

Well-known examples of constraint-based tutors are those developed at the Intelligent Computer Tutoring Group at the University of Canterbury (see, e.g. Mitrovic et al., 2007). This group also developed a tool that assists with the creation and delivery of constraint-based tutoring systems, named ASPIRE (Mitrovic et al., 2006). The constraint-based modeling approach is also used by Ryder et al. (2000) to construct the instructor agent embedded in their Electronically Assisted Ground Based Learning Environment (EAGLE) for the training of UAV operators, see page 57.

Comparison of Techniques A significant advantage of constraint-based modeling over model-tracing is the smaller required development effort. There is no need to specify an executable expert model, nor a complete set of buggy rules. Moreover, the student is not constrained in the process leading to its results. As long as its final-state does not violate a constraint, the student is free to think of new solutions. However, the fact that a student's process is not constrained has raised discussion on the applicability of constraint-based modeling to the training of procedural tasks (Kodaganallur et al., 2005). Moreover, Kodaganallur et al. question its capability to generate error-specific feedback, in specific whether it is possible for all types of tasks to specify abstract constraints that must always hold for a solution. This last point is relevant to our research objective. For training open, complex tasks in which not the outcome of a decision is of main importance, but the reasoning-process leading to this decision, it might be difficult to specify such constraints. For extensive comparisons of the model-tracing and constraint-based modeling techniques and discussions on their suitability for various types of tasks and domains, see Mitrovic et al. (2003); Kodaganallur et al. (2005); Mitrovic and Ohlsson (2006); Kodaganallur et al. (2006).

2.5 Conclusion and Prospect

In this chapter we introduced and discussed research from various research fields. We introduced this wide variety of research topics because the research described in the rest of this dissertation is related to it all. We took the freedom given by the general topic of 'cognitive models for training simulations' to perform a PhD study spanning several research topics and research fields.

In the coming chapters we address five different research topics. Each chapter embeds papers that are all but one published, and starts with an introductory section. In these introductions we link the papers within the chapter to each other and place them within the general research framework as introduced in this chapter. When required, we also elaborate on more specific related work than described in the current chapter.

Although the topics addressed are quite diverse, the same research thread is prevalent in them all: using *natural science knowledge* concerning cognition and education and *design science knowledge* concerning artificial intelligence and software engineering to construct models that, when implemented in software, form agents that can display, or provide feedback to, human (biased) behavior.