In Proceedings of ICCM - 2007- Eighth International Conference on Cognitive Modeling. 199 - 204. Oxford, UK: Taylor & Francis/Psychology Press.

From a Formal Cognitive Task Model to an Implemented ACT-R Model

Fiemke Both[1] (fboth@few.vu.nl)
Annerieke Heuvelink[1,2] (heuvel@few.vu.nl)
[1]Vrije Universiteit Amsterdam, Department of Artificial Intelligence,
De Boelelaan 1081a, 1081 HV Amsterdam, the Netherlands
[2]TNO Defence, Security and Safety, Department of Training and Instruction
Kampweg 5, 3769 DE, Soesterberg, the Netherlands

Abstract

In order to generate behavior that can be validated, a cognitive task model needs to be implemented in software. This paper first introduces a cognitive task model based on a BDI framework and then focuses on the translation of that model into the ACT-R theory and software. Problems encountered during this translation process are described and further implications are discussed. It is found that the model's control structure matches well with ACT-R, but that the translation of its reasoning rules is complex. ACT-R's theory of cognition does not match with properties of our task model on three issues: 1) the number of memory items that can be stored in working memory, 2) the way memory items are retrieved from long-term memory, 3) the ability to execute complex computations.

Introduction

The development and implementation of cognitive task models in order to create agents that can replace humans for performing certain tasks in certain environments is a promising research activity (Gluck & Pew, 2005). Two important activities in the development of a cognitive model are the modeling of relevant task knowledge and the modeling of a cognitive valid theory of task execution. These modeling activities typically yield a formal design on paper of the cognitive task model. However, for studying the model's behavior and, to use the model as a replacement for a human in a certain environment it needs to be implemented in software.

The work presented in this paper is part of a greater research project that has as its goal the development of a cognitive agent that can perform the task of compiling a tactical picture on board of a ship in a training simulation. The current paper describes and reflects on the translation of the cognitive task model into the cognitive architecture ACT-R (Anderson et al, 1998). It will not go into details of the task, or into the validity of the behavior that the resulting cognitive agent shows.

First, we will briefly introduce the developed cognitive task model with its main properties concerning task knowledge and task control. Next, we discuss the translation of this model into ACT-R. To test the resulting cognitive agent we coupled it to an external simulation environment. This coupling and the general task performance of the ACT-R model are discussed. Furthermore, we reflect upon problems encountered during the translation as well as their implications. Finally, we lay down further research plans.

Research Domain

The task we modeled is the Tactical Picture Compilation Task (TPCT) within the domain of naval warfare. This task revolves around the classification and identification of entities in the surroundings. The warfare officer responsible for the TPCT monitors the radar screen for radar contacts and reasons with the available information in order to determine the type and intent of the contacts on the screen. The cognitive model of the TPCT is based on a Belief, Desire and Intention (BDI) framework (Georgeff and Lansky 1987). This choice facilitates the translation of domain knowledge into the model since domain experts tend to talk about their knowledge in terms similar to beliefs, desires and intentions. The domain knowledge needed for performing the TPCT has been elicited from naval experts by van Dam and Arciszewski (2002).

The goal of our research is the development of cognitive agents that can be used for training purposes. We therefore want to model cognitive behavior, which can vary in level of rationality. To make this possible, we developed a specific belief framework (Heuvelink, 2006). Three arguments are added to beliefs: a timestamp, the source of the information and a certainty level. Usually in BDI models, beliefs are thrown away as soon as a new belief is created that causes an inconsistency. However, because we want to reason over time, every belief is labeled with the time it is created and is never thrown away. The source and certainty labels make it possible to model an agent that can reason about information from multiple sources and with uncertainty, and that might do this in a biased way. A belief according to this new framework consists of a predicate P with an attribute A and value V, a timestamp T, source S and certainty C. An example belief is shown below.

Belief (Identification (contact1, friendly), 12, determine-identification, 0.7)

Cognitive Task Model

A cognitive task model typically consists of declarative knowledge, denoting facts, as well as procedural knowledge, denoting reasoning rules. Besides modeling how to reason, it is also necessary to model when to reason about what. In this paper, we mainly address the modeling of the declarative and procedural knowledge necessary for performing the TPCT. The modeling of cognitive control is not our current focus, and therefore we limit the complexity

of our control structure. First, we describe the format of the declarative and procedural knowledge embedded in the cognitive model. Then we elaborate on the control structure of the model and at the end, we present the conceptual design of our agent.

Reasoning over Beliefs

The goal of the TPCT is to correctly classify and identify all contacts in the surroundings. To draw these kind of conclusions about contacts, the agent needs knowledge about their behavior. There are two ways to gather such information. The first is from the external world, e.g., the agent can watch the screen that displays information from sensors such as the radar system. Additionally, the agent can decide to perform actions that lead to more knowledge about the situation, such as activating radar or sending a helicopter to investigate a contact. The second method to gain information is through the internal process of reasoning about beliefs to deduce new beliefs. In the reasoning process, often multiple beliefs form the evidence for the formation of a new belief. Any uncertainty in the source beliefs will be transferred to the new belief.

The following rule is an example of how a new belief is derived using other beliefs and domain knowledge. The position of the contact that the agent is currently reasoning about is compared to the position of every other contact that is detected by the radar system. A new belief is created for every pair that indicates how certain the agent is that they are within formation distance. Names of functions are depicted bold and names of parameters are italic.

Determine-Within-Formation-Distance-Contact(X)
For all Y
  If (
    belief(Position-Contact(X, P1),T1, S, C1)
    belief(Position-Contact (Y, P2), R1, S, C2)
    Position-Difference(P1, P2, D)
    Certainty-Handling- Difference-Between-Positions(C1, C2, D, C3)
    Possible-Distances(D,C3, [R])
    M = *maximum-distance-relevant-for-formation*
    C4 = (number-of-[R <= M]) / (number-of-[R]))
  Then (
    Reason-Belief-Parameter(Within-Formation-Distance- Contact (X, Y), determine-within-formation-distance-contact, C4))

The function Position-Difference calculates the distance between two positions, Certainty-Handling-Difference-Between-Positions calculates the certainty of the distance given the certainties of the positions, Possible-Distances returns all possible distances given the calculated distance and certainty, and the rule Reason-Belief-Parameter adds the timestamp and stores the belief in long-term memory. In this example, the latest beliefs about the positions are used. In other rules, beliefs are used that ever had a specific value, or those beliefs the agent is most certain about.

Control of Reasoning

Control is an important aspect of a cognitive agent; it determines when the agent does what. The TPCT has one main goal, which is considered a desire in the BDI model: to identify all contacts correctly. The three subtasks that the agent can perform in order to fulfill this desire are 1) processing information about contacts from the screen, 2) changing the activity of the radar system, and 3) sending the helicopter on observation missions to gain more information about a specific contact. These subtasks are the intentions of the BDI model that the agent can commit to.

A cognitive valid manner to determine when which intention becomes a commitment is to have events in the world trigger an intention. For example, when a contact suddenly changes its behavior, the attention of the agent should be drawn to this contact, regardless of the current intention. However, this type of control requires a parallel processing of all events in the world and a parallel checking of relevancy for all subtasks, which is very difficult to implement. That is why currently, we chose to implement a simpler, linear control system. To simulate parallel processing we let the agent alternately commit to one of the three intentions. Within the subtasks, the control is also kept simple, e.g., in the first subtask all contacts on the screen that are stored in a random list are monitored consecutively.

The following reasoning rule is an example part of the simple control structure. It determines when the agent starts committing to a new intention.

Determine-New-Intention(I)
  If (
    I = Monitor-Contacts
    belief(Number-of-Contacts-Monitored(X), T1, S, C)
    X = *maximum-number-of-contacts-to-monitor)*
  Then (
    Start-New-Intention-Selection(I))
  Else if (
    I = Monitor-Contacts
    Number-of-Contacts(X)
    X < *maximum-number-of-contacts-to-monitor)*
  Then (
    Select-Next-Contact-To-Monitor()
    Reason-Belief-Parameter(Number-of-Contacts-Monitored(X+1), determine-new-intention, 1))

The input parameter *I* is the current intention, the rule Start-New-Intention-Selection determines which intention is selected next depending on beliefs about contacts, and the rule Select-Next-Contact-To-Monitor selects the next contact from the list.

Conceptual Agent Design

We have made a conceptual design of the agent capable of performing the TPCT using the DESIRE (DEsign and Specification of Interacting REasoning components, Brazier, Jonker and Treur, 2002) method. DESIRE's view of an agent is that of a composed structure consisting of interacting components. This conceptual agent model in DESIRE completes the formal model of the TPCT with the control of the different subtasks. DESIRE enables us to model the flow of information within the agent and between the agent and the external world. The agent components of the conceptual model are displayed in Figure 1, representing different kinds of tasks at different levels.

At the top level, there are two components: Warfare Officer and External World. This makes it possible to model communication between the agent and the external world. At the second level, three components are selected from the Generic Agent Model (GAM, Brazier, Jonker & Treur, 2000): (1) Own Process Control, (2) Maintenance of World Information and (3) World Interaction Management. The component Own Process Control, responsible for desire, intention and belief determination, is refined according to the addition to GAM for BDI models (Brazier et al, 2001). The component Maintenance of World Information stores the beliefs created by a subcomponent of Own Process Control. The last GAM component, World Interaction Management, manages communication with the external world via two subcomponents, Action and Observation Request Management. The fourth component at the second level, Agent Specific Task, performs tasks that require an internal decision to be made (Decision Control), and tasks that require reasoning about beliefs to derive new beliefs (Reasoner).

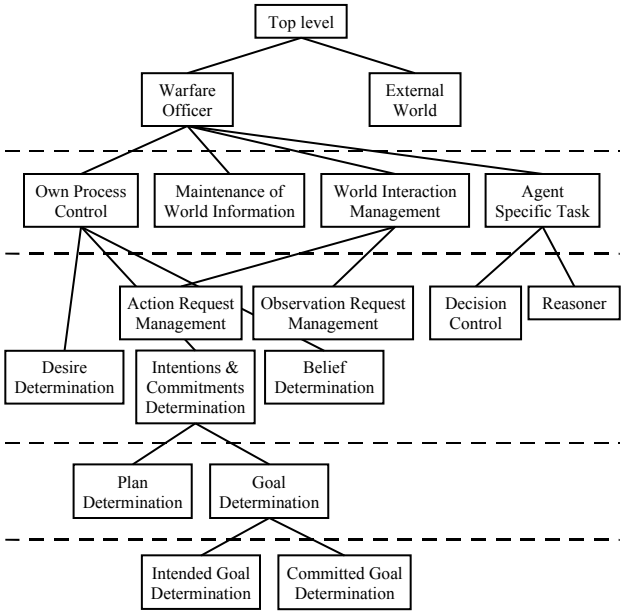

Figure 1 Process decomposition for the agent

Translation Process

This section first introduces ACT-R and than continues with a description of the translation process of the reasoning rules and the control of these rules into the ACT-R architecture. Finally, it introduces the environment with which the resulting agent interacts.

ACT-R

ACT-R incorporates two types of memory modules: declarative memory and procedural memory. Declarative memory is the part of human memory that can store items; procedural memory is the long-term memory of skills and procedures. ACT-R consists of a central processing system, where the *production rules* representing procedural memory are stored and executed. The central processing system can communicate with several modules through buffers. One of those modules is the *declarative memory module* where memory items are stored. These memory items, called *chunks*, are of a specific chunk-type, which can be defined by the modeler. In a chunk-type definition, the modeler defines a number of slots that chunks of this type can assign values to. Chunks from the declarative memory module can be placed in the *retrieval buffer* if they match a retrieval request made by a production rule. A retrieval request must contain the requested chunk-type, and may contain one or more slot-value pairs that the chunk must match. The matching chunk is then placed in the retrieval buffer, so it can be read by a production rule. All buffers in ACT-R, including the retrieval buffer, can only store one chunk at a time, even when more chunks match the conditions of the request.

Reasoning Rules

In the following paragraphs, the implementation of the reasoning rules of the formal task model in the declarative and procedural memory modules is described.

Declarative Memory The cognitive model of the TPCT requires the ability to retrieve beliefs with specific time and certainty constraints. The rule Determine-Within-Formation-Distance-Contact for example, requires the latest beliefs about the position of two contacts. Because beliefs are not thrown away, there may be many older beliefs about the positions of these contacts. It is therefore necessary to retrieve the latest belief of both contacts in order to determine the current distance between them. In ACT-R, creation and retrieval times of chunks are registered, but these properties are not available to the modeler. There is also no feature that enables the agent to retrieve a chunk with the highest value of a slot.

The method ACT-R provides for retrieving chunks from memory is an activation function for chunks. The activation of a chunk indicates how easy it should be to retrieve it. Every chunk that matches a retrieval request receives an activation score based on three components: a *base level activation,* a *context component* and a *noise value.* The base level activation uses the number of representations and the time since the representations. Representations are the initial entry in the declarative module and a retrieval in the declarative buffer. The context component is based on the activation of related chunks. The noise value adds a random factor to the activation value.

Our task model assumes that when the last belief is required for a reasoning process, that the last belief is always retrieved. In ACT-R, it is possible that an older chunk has been retrieved more often than the latest chunk, which results in a higher activation value. Although there are several parameters that can be set, the activation function cannot guarantee that the latest chunk is always the most active. To meet the requirements of the task model, it

is therefore necessary to implement the retrieval process of those beliefs using LISP, the language ACT-R is built in.

Procedural Memory There are several theories about working memory (WM), most of which agree on the idea that multiple memory items can be stored at the same time in WM (Miller, 1956; Baddeley & Hitch, 1974; Hulme et al, 1995; Cowan, 2005). The number of items that can be stored in WM ranges from two to seven in these theories. ACT-R's theory of WM is that there can only be one chunk of memory stored in WM at a time. This representation of WM is not consistent with the most commonly accepted cognitive theories of WM, as well as with the formal task model based on the developed belief framework.

In the formal task model, several beliefs need to be in WM at the same time to be able to reason. For instance, different positions over time are compared in order to determine a contact's heading and speed. To implement such a rule in the ACT-R model, multiple production rules are needed to retrieve all required beliefs. Each production rule would have to draw a sub conclusion about the retrieved information so far. This method is inefficient for this task and it is inconsistent with the reasoning methods described by the naval experts. Therefore, we have tried two different methods to solve this problem. The first solution is to merge all the beliefs that need to be compared into one chunk. This solution however, still requires many production rules to retrieve all the necessary beliefs and undermines the ACT-R theory of WM being able to store only one chunk at a time.

The second method, used in the final model, consists of two parts: using the goal buffer for temporary storage and using LISP functions to retrieve beliefs. The *goal buffer* can hold just one chunk, so the different memory items need to be stored in the slots of the goal chunk. Many ACT-R modelers use this solution to be able to compare information items (see e.g., the tutorials of ACT-R 6.0). For example, in our task model several beliefs about the location of a contact at different time points form the antecedent of the logical rule that calculates the belief about the speed of a contact. These different locations can be stored in the goal buffer until all relevant locations are retrieved and the speed can be calculated. The second part of our solution is that we used LISP functions instead of production rules to access chunks in the declarative module of ACT-R. LISP functions can be called within a production rule that is firing. Then, the LISP function can retrieve multiple beliefs at the same time, compare them, and return the result to the active production rule.

The many calculations the task model requires, e.g., for calculating the certainty of a new belief from the uncertainties of the beliefs it is derived from, is a second problem we encountered during translation. In most ACT-R models of low-level tasks, calculations are modeled in production rules. The addition of three and four for example, would take ten rules: one for the initialization, two for every addition step (increment by one, remember how much has been added) and one for the finalization. It would take many more production rules to calculate the speed of a contact from its positions over time. In addition, most of the calculations are not an exact representation of the cognitive processes of a warfare officer, but are a more abstract representation. It is therefore not necessary and not efficient to model all computations in production rules. We chose to use LISP functions for those calculations.

By executing the above processes in LISP functions instead of in production rules, the amount of production rules is reduced to half of the programming code of the agent. The other half consists of supporting LISP functions. The following ACT-R production rule is an example of how we implemented the task model rule Determine-Within-Formation-Distance-Contact and how we solved the problems described above.

```
(p determine-within-formation-dist-goal1-plan6
   =goal>
      ISA         commitment1
      goal        monitor-contacts
      state       determine-formation-distance
      contact     =contact
==>
   !bind! =within-dist (calc-within-formation-distance =contact)
   !bind! =next-step (if =within-dist 'determine-formation 'determine-
   classification')
   =goal>
      state       =next-step
      result      =within-dist )
```

The antecedent of the rule requires that the agent is currently committed to commitment1; processing information about contacts from the screen, and that the agent is going to determine whether the current contact is within formation distance of any other contact. In the consequent, the agent uses a LISP function to find all contacts within formation distance (function calc-within-formation-distance). In this LISP function, multiple beliefs with the latest timestamp are retrieved, and calculations are performed to determine the distance. If a contact exists that is close enough to the current contact, the agent starts with determining whether they actually are moving in formation. Otherwise, the next step is to determine the classification of the current contact.

Control of Reasoning Rules

As explained above we implemented a simple, linear control structure, in which the agent alternately commits to its various subtasks. We showed the reasoning rule Determine-New-Intention, which determines that after reasoning about a number of contacts one of the other subtasks becomes the agent's intention. We did not run into any problems when implementing these task model control rules in ACT-R. In ACT-R, the control structure is also linear: only one production rule can fire at a time. The antecedent determines which rule matches the current contents of the buffers. The current intention and current step in the plan can be stored in the goal buffer. The consequence of a fired rule can influence which production rule will fire next. In

our agent, this was generally done by changing the contents of the goal buffer. The following ACT-R production rule shows how we implemented the rule Determine-New-Intention.

```
(p select-next-contact-goal1
   =goal>
      ISA                 commitment1
      goal                monitor-contacts
      state               next-contact
      contact             =contact1
==>
   !bind! =contact2 (determine-next-contact)
   !bind! =new-intention (= (mod *contact-counter* *max-number-of-
   contacts*) 0)
    =goal>
      Plan                read-basal-info
      state               start-step
      contact             =contact2
      new-intention       =new-intention )
```

The antecedent of the rule requires that the agent is currently committed to monitoring contacts, and that the agent is going to select a new contact. In the consequent, the agent determines whether to select the next contact or to start a new intention. The slot new-intention of the goal chunk can be true or false, influencing the next production rule that can fire.

Coupling Environment

ACT-R has different modules for simulating visual and aural stimuli and vocal and manual actions. The visual module can be used to show letters and numbers in a window, for example to simulate the Stroop-task. However, the current task requires the visualization of a naval scenario and the features of the visual module are too limited for this. Therefore, a coupling has been made between ACT-R and an external simulation environment: Game Maker (see Game Maker). The simulation environment we developed in Game Maker consists of a screen that shows radar contacts. An additional screen displays detailed information about a contact when it is selected by a mouse click. This same window can be used to change the classification and identification value of the selected contact.

ACT-R and Game Maker communicate through two text files. In the first file, the ACT-R agent writes information and action requests, which Game Maker reads and performs. This can be, for instance, a request for detailed information, simulating a mouse click on a contact. In the other file, Game Maker writes the requested information and feedback about the performed action, which ACT-R reads and processes. The response of Game Maker to the mouse click would be to write the detailed information of the contact in the text file.

Results and Discussion

Now that we have implemented the TPCT model in ACT-R and coupled the resulting agent with a simulation environment, we can test the behavior of the agent. In this section, we will describe and discuss the results of the translation process and the coupling with Game Maker.

Before translating the task model into ACT-R, we developed a conceptual agent model in DESIRE that gave us a structured overview of the control and information flows within the task model. We find that the structure of this conceptual agent model matches well with ACT-R. Each of the components of the conceptual model (see Figure 1) can be identified in the ACT-R code. This made the conceptual model a useful structure to base the ACT-R model on.

In addition, the serial control was easily translated from the DESIRE model to ACT-R production rules. In ACT-R, one production rule can fire at a time. This principle is the same as we chose for our agent. However, if we had chosen a more complex control, we probably would have had more trouble translating the conceptual model to ACT-R. For example, a function that determines which contact on the screen deserves attention requires calculations. Since we have already shown that calculations are difficult to model in ACT-R, a more complex control system will be more difficult to implement.

During the implementation process, we encountered some problems when translating the reasoning rules of our model to ACT-R production rules. The ACT-R theory entails that one chunk can be stored in WM, that the chunk with the highest activation value is retrieved, and that it is difficult to combine low-level calculations and high-level reasoning rules in one model. Our solution has been to implement those processes in LISP functions. However, by using so much LISP code and by not using the declarative memory module properly, the structures built into ACT-R that constitutes its theory of cognition are denied. It is likely that both theories are incomplete. ACT-R should support the storage of multiple chunks in the retrieval buffer at the same time since most researchers agree that human WM can store multiple beliefs. We should reconsider the many calculations performed in the rules of the cognitive task model.

A more practical issue we encountered during implementation concerns the number of chunks in ACT-R's memory. Every couple of minutes the agent is active, its number of chunks doubles. As a result, ACT-R becomes very slow when the agent tries to retrieve a chunk from memory. We partly solved this by creating fewer chunks during reasoning. In the original task model, every reasoning step resulted in a new belief. We identified types of beliefs that were never retrieved by the agent, and stopped adding them to ACT-R's memory. For example, only the result of a calculation is remembered, instead of all steps leading to this result. Furthermore, we deleted irrelevant chunks from the agent's memory. For instance, if the speed of a contact has been constant for a long time, the first and last beliefs about that speed provide all the information ever needed.

The coupling of the ACT-R agent and the simulation environment Game Maker consists of communication through text files; one file in which ACT-R writes its requests and. one file in which Game Maker writes its

results. It is computationally impossible for both Game Maker and ACT-R to check the text file constantly for new information. The less often the text file is checked for new information, the slower the communication process becomes. However, if Game Maker and the agent would check it more often, the entire simulation would slow down.

We tried to find a balance by having Game Maker read the text file every second. The agent only reads the text file when it expects new information. Thus, when the agent communicates a request to Game Maker, it keeps reading the text file until the new information has arrived. In practice, this means that ACT-R reads the text file approximately every half second. In the future, we would prefer a different type of coupling that enables streaming of information, so the two parties do not have to actively check for new information.

Conclusion and Future Research

We implemented a cognitive task model in ACT-R and tested how it functioned. During implementation, we found that the DESIRE control model fits well with ACT-R, but that the model's reasoning rules that are based on the developed belief framework do not fit.

ACT-R is a cognitive architecture that consists of several components. Two of those components are used for the agent: the declarative memory module and the procedural memory system. The communication with the chunks in the declarative memory module is not done by the use of the retrieval buffer, but through LISP functions. LISP functions are used to solve three issues. First, the task model requires the comparison of more than one belief, and ACT-R can only hold one belief in the representation of WM. Second, the task model requires the latest or most certain belief to be retrieved, and ACT-R does not offer a function to request beliefs with these specifics. Third, the task model describes many calculations, which are difficult to implement using ACT-R's production rules. Because only part of the theory of ACT-R matches the formal cognitive task model, this cognitive architecture is not very well suited for this kind of task model and this forced us to implement a great part of the agent in LISP code.

Overall, the implemented agent is slow. ACT-R is a software package that uses about half of the computer's CPU power, and Game Maker uses the other half. In ACT-R the slow speed is mainly due to the extensive search for specific beliefs caused by the exponential grow of chunks. To improve the performance of the agent it is necessary to extend the task model with a cognitive model for the decay of beliefs, or with a system that throws away beliefs that are not relevant for the task anymore.

In the future, we want to translate the cognitive task model to SOAR to research how well it matches with that cognitive architecture. This activity will yield a SOAR agent whose performance we can compare with the current ACT-R agent. To increase the cognitive validity of the task model we also want to focus on the development of a more cognitive plausible control system.

Acknowledgments

The authors like to thank Tibor Bosse and Karel van der Bosch for their guidance during this research project. Furthermore we like to thank Jan Treur for commenting an earlier draft, Willem van Doesburg for developing an initial version of the training environment, and the naval instructors of the Opschool for providing domain knowledge and for taking part in the experiment.

References

Anderson, J. R., & Lebiere, C. (1998). The Atomic Components of Thought. Mahwah, NJ: Erlbaum.

Baddeley, A.D. & Hitch, G.J. (1974). Working Memory, In G.A. Bower (Ed.), *Recent advances in learning and motivation (Vol. 8, pp. 47-90)*, New York: Academic Press.

Brazier, F.M.T., Dunin-Keplicz, B., Treur, J. & Verbrugge, R. (2001). Modelling Internal Dynamic Behaviour of BDI Agents. In Gabbay, D., and Smets, Ph. (eds.), *Dynamics and Management of Reasoning Processes.* Series in Defeasible Reasoning and Uncertainty Management Systems, 6. Kluwer Academic Publishers, 2001, 339-361.

Brazier, F.M.T., Jonker, C.M. & Treur, J. (2000) Compositional Design and Reuse of a Generic Agent Model. *Applied Artificial Intelligence Journal, 14.* 491-538.

Brazier, F.M.T., Jonker, C.M., & Treur, J., (2002). Principles of Component-Based Design of Intelligent Agents. *Data and Knowledge Engineering, 41*, 1-28.

Cowan, N. (2005). *Working memory capacity*. New York, NY: Psychology Press.

Dam, B. J. van & Arciszewski, H. F. R. (2002). *Studie Commandovoering DO-2: Beeldvorming* (FEL Technical Report FEL-02-A242). The Hague, the Netherlands, TNO FEL.

Game Maker. http://www.gamemaker.nl/

Georgeff, M.P., & Lansky, A.L. (1987). Reactive Reasoning and Planning. *Proceedings of the Sixth National Conference on Artificial Intelligence*, 677 – 682. Menlo Park, Ca: AAAI Press.

Gluck, K.A., & Pew, R.W. (Eds.) (2005). *Modeling Human Behavior with Integrated Cognitive Architectures: Comparison, Evaluation, and Validation*. Mahwah, NJ: Lawrence Erlbaum Associates.

Heuvelink, A. (2006). Modeling Cognition as Querying a Database of Labeled Beliefs. In D. Fum, F. Del Missier, and A. Stocco (Eds.), *Proceedings of the 7th International Conference on Cognitive Modeling (ICCM 2006)*, 365-366. April 5-8, Trieste - Italy.

Hulme, C., Roodenrys, S., Brown, G., & Mercer, R. (1995). The role of long-term memory mechanisms in memory span. *British Journal of Psychology, 86*, 527-536.

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