Flexible Design and Implementation of Cognitive Models for Predicting Pilot Errors in Cockpit Design

Jurriaan van Diggelen, Joris Janssen, Tina Mioch, Mark Neerincx
TNO Human Factors,
Kampweg 5,
3769 DE Soesterberg,
The Netherlands
{jurriaan.vandiggelen, joris.janssen, tina.mioch, mark.neerincx}@tno.nl

Abstract

This paper describes an integrated design and implementation framework for cognitive models in complex task environments. We propose a task- and human-centered development methodology for deriving the cognitive models, and present a goal-based framework for implementing them. We illustrate our approach by modelling cognitive lockup as an error producing mechanism for pilots, and present the outcomes of the implemented cognitive models that resulted from applying our methods and tools.

Introduction

The HUMAN project seeks to use a cognitive architecture for simulating and predicting pilot errors in the aviation domain [3]. An ambitious endeavour such as this one poses many challenges for the project team, such as eliciting domain information, developing plausible psychological models of motor, sensing and thought processes, developing realistic scenarios, and implementing the cognitive model using state-of-the-art Artificial Intelligence (AI) techniques. Each research activity must be performed in close collaboration with the others, such that opportunities and constraints are properly observed and propagated throughout the project.

This principle also applies for the AI implementation of the cognitive architecture. We cannot expect to implement the architecture in a one-shot fashion, but must be prepared for continuous adjustments of the implementation due to changing requirements, functionality, and scope. In other words, the implementation must be flexible. The purpose of this paper is to show a flexible method for designing and implementing the cognitive layer using the HUMAN architecture and to report on our experiences modelling cognition using this methodology and architecture in the aviation domain.

The HUMAN project has adopted a multi-layered architecture where each layer processes information on a different level of abstraction, and functions relatively independent of the other layers. This has a number of benefits. Firstly, it

allows for a much more flexible implementation (by different parties) than a monolithic architecture would. By distinguishing different relatively independent components, developers can focus on more simple parts of the model, which makes it easier to comprehend and adjust.

Secondly, the different layers correspond well with cognitive engineering theory as proposed by Rasmussen [6], which makes implementation of psychological models more straightforward.

However, modelling mental processes in such a layered architecture also poses a number of challenges for design, implementation and evaluation. A design challenge is to make sure that domain and human factors knowledge are properly identified and used throughout the development process. For this purpose, we have applied the Situated Cognitive Engineering methodology [5].

Implementation challenges are to ensure interoperability between the layers (i.e. making sure that the output of one layer is properly understood by the other layer) and to decide which cognitive processes should be modelled in which layers, by which AI techniques. Also, we would like to separate domain-specific knowledge from general reasoning mechanisms, which allows the framework to be reused for different domains and easily altered when implementation requirements change. For the cognitive layer, we solved these implementation issues by coupling multiple AI technologies such as Protégé-ontologies, CLIPS expert systems, and goal hierarchies.

The evaluation challenge consists of validating the cognitive layer by comparing event traces from the computational model, with real data gathered from experiments with human pilots. We discuss how we could use these outcomes for refining the cognitive layer.

The paper is organized as follows. The next section describes the Situated Cognitive Engineering methodology as a way to iteratively design, implement and evaluate cognitive models. The generic software architecture is described in Section 3. In the fourth section, we describe how we have applied the architecture to implement the cognitive layer using a specific case. Section 5 provides a conclusion and future work.

Situated Cognitive Engineering

The cognitive models are developed using the situated Cognitive Engineering (sCE) methodology [5], as depicted in Figure 1.

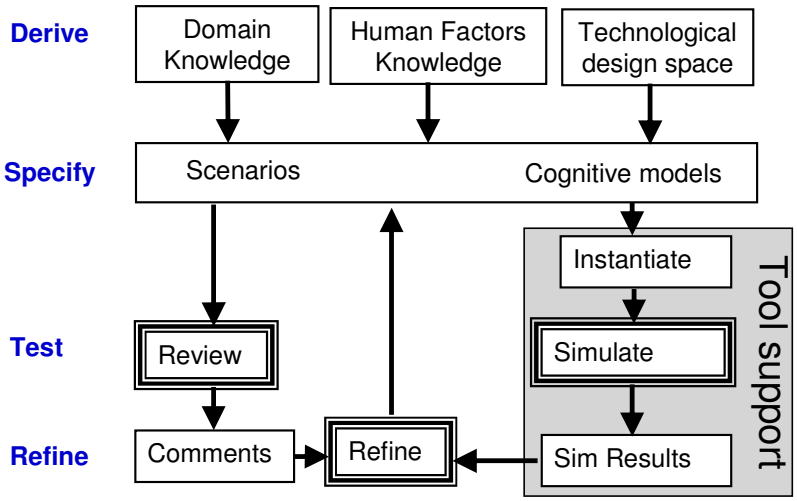

Figure 1: Situated Cognitive Engineering (sCE) for Cognitive Modelling

The methodology is characterized by the following properties:

- Cognitive models are developed incrementally, in an *iterative* process of specification, refinement and testing.
- Human factors, domain aspects, and technological issues are studied early in the engineering process, and used throughout the entire development process, leading to a *situated* cognitive engineering approach.
- The approach offers tool-support for implementing the cognitive models, running simulations, and obtaining results.

As can be seen in the figure, the development of situated cognitive models occurs in four stages: Derive, Specify, Test, and Refine. Each of these phases is further explained below.

In the *derive* phase, domain knowledge and human factors knowledge is collected using several techniques, such as field observations, critical incidents analyses and interviews with domain experts. For example, in the HUMAN project we have investigated the aviation domain, by literature reviews and performing interviews with pilots. Additionally, we collect human factors knowledge which is relevant for this domain. For example, we have identified *cognitive lockup* as a potential serious error causing mechanism for airplane pilots.

The next phase is the *specify* phase, where the knowledge obtained in the previous phase is made more concrete in scenarios and cognitive models. We can distinguish between two types of scenarios: scenarios which illustrate normative behavior, and extreme scenarios which illustrate potential errors occurring under certain conditions. The first type of scenarios results directly from the domain study. The second type of scenarios results from domain knowledge which is combined with human factors knowledge of error causing mechanisms. For exam-

ple, we could develop operationally relevant scenarios in which the pilot is faced with a combination of context factors from which we know (from a human factors perspective) that cognitive lockup is likely to occur. Simultaneously with the scenarios, we develop conceptual cognitive models which can be used to simulate the pilot's behavior which is described in these scenarios.

In the *test* phase, the cognitive models are evaluated. This can be done by obtaining feedback from colleagues, e.g. in scientific conferences or workshops. A more objective way of testing is to implement the cognitive models, and instantiate them with appropriate data, and obtain simulation results.

In the *refinement* phase, the simulation results can be compared with the actual data, which leads to a further refinement of the model.

Software Architecture

The tool-support we have developed for simulating cognitive models, is included in the general HUMAN architecture. The HUMAN architecture is based on Rasmussen's three behavior levels in which cognitive processing takes place: skill-based, rule-based and knowledge-based behavior [6]. The levels of processing differ with regard to their demands on attention control dependent on prior experience:

- autonomous layer: this layer models reflexive behavior.
- associative layer: this layer models procedural behavior in terms of signs.
- cognitive layer: this layer models deliberative behavior in terms of symbols.

In addition to the three levels, Rasmussen also assigns a type of information to each level. Information is categorized into signals, signs and symbols. At the skill-based level, signals represent the information as it has been perceived, e.g. altitude is 200 feet. Signals can then be enriched with further contextual information, e.g. altitude < 1000 feet, and transformed into signs, to be used at the associative layer. These signs can then be associated to semantic information and general knowledge and transformed into symbols, to be used at the cognitive level. For more details on the general architecture see [3].

Because the mental processes which are of interest to this paper are high-level processes, they are modeled at the cognitive layer. Most cognitive agent reasoning processes can roughly be divided in three phases: a *sense* phase, a *reason* phase and an *act* phase [7]. We can apply the same distinction for our cognitive simulation tool.

In the sense phase, the right knowledge is gathered which serves as a basis to make appropriate decisions. In the cognitive layer, new knowledge can be created in two ways. Firstly, new knowledge can arise from perceptions in the environment. In our framework, this knowledge enters the cognitive layer via the associative layer. For this purpose, a translation is needed from knowledge represented in the form of signs, to knowledge represented in the form of symbols. Secondly, new knowledge can be a result of reasoning with existing knowledge. This is performed by a knowledge reasoning component. We refer to both of these functionalities as *knowledge management*.

In the reason phase, the agent uses its knowledge to decide which action to perform next. Following the intelligent agent paradigm, we use a *goal hierarchy* to describe which actions must be executed, given the agent's *goals* and *beliefs*. Unlike many other approaches for goal-based agent deliberation, we do not only strive for efficiency and effectiveness, but also for realism (i.e. analogue to human deliberation). In this way, we can use the framework for modeling human errors as well. We refer to these functionalities as *decision making.*

In the act phase, the agent performs the action, or task. In the context of this paper, tasks are restricted to *mental tasks*. This means that a reasoning step is performed, resulting in some piece of new knowledge. We refer to this functionality as t*ask execution.*

For each of the three functionalities described above, we have developed a separate module as depicted in Figure 2.

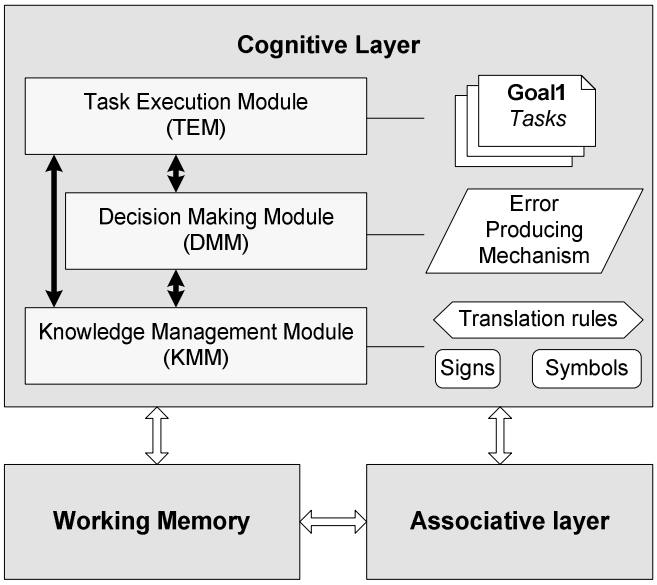

Figure 2: The components in the cognitive layer

DMM: The decision-making module is also called goal management, and determines which goal is executed. Each goal contains *preconditions* which specify when the goal is *active*. To check the truth value of a precondition, it consults the knowledge represented in the KMM. Which of the active goals will be selected to be executed is determined by the goal-prioritization mechanism. In the human factors analysis phase of the cognitive engineering method, we identified Cognitive Lockup (see [4]) as a relevant error producing mechanism (EPM). In the decision-making module, we have modeled this by introducing *task switch costs* (TSC) representing the difference that goal priorities must have before switching goals.

KMM: The Knowledge Management Module communicates with the Working Memory (WM) in both directions. Signs are sent from WM to the KMM to enable translation from signs to symbols. Symbols are sent from KMM to WM to store these newly derived symbols for future use by the cognitive layer. We apply two types of technology: ontologies and expert systems. Ontologies make syntactic and semantic assumptions of signs and symbols explicit to facilitate implementation and communication [2]. To convert signs in AL into symbols in CL, we have implemented a sign-symbol translator using the rule-based language CLIPS [1].

TEM: The Task Execution Module executes tasks that can lead to fulfillment of the goal which has been selected by the DMM. Tasks (or lower-level goals) that the AL can handle are passed to the AL. Tasks that involve sign-symbol translations or involve other kind of deductive reasoning are passed to the KMM.

Case

To demonstrate the functionality of the cognitive layer, we describe a case which shows the occurrence of cognitive lockup.

Scenario

The scenario development is the result of the domain analysis and the application of human factors knowledge (see Section 0). During the domain analysis, pilots were interrogated about possible tasks and events that match the human factors knowledge about cognitive lockup. For example, as we are modeling tasks on the cognitive layer, only tasks which the pilot executes consciously and non-routinely should be chosen in the scenario. In addition, pilots could provide an idea of importance and priority of different tasks.

This has resulted in a scenario where during the cruise phase, the pilot is flying towards his destination. At one point a thunderstorm appears on the weather radar, close to the destination airport. As it is not clear whether the thunderstorm affects the current trajectory and the pilot needs to redirect to the alternate airport, the pilot watches the storm closely to decide on its importance and development over time. This task can be seen as an engaging task, which demands attention of the pilot. During this monitoring task, the system indicates a malfunction with one of the aircraft engines. The pilot recognizes this event (at timestep 1), but does not immediately try to solve the issue. Instead the pilot continues the monitoring task of the thunderstorm (at timestep 2). After a certain time (at timestep 3), the urgency to handle the problem with the engines is realized by the pilot and the pilot starts solving the system malfunction.

Implementation

As described in Section 0, three modules need to be instantiated to implement a scenario. First, the decision-making module needs to be set to the cognitive-lockup bias. Second, the Knowledge Management Module and the Task Execution Module need to be implemented.

For the TEM, this means that the top-level goals and the low-level goals need to be defined. The top-level goals for this scenario have been identified by the domain experts to be the goals to monitor the thunderstorm (*WatchStorm*), and to handle the system malfunction (*HandleSystemMalfunction*).

Event traces

During the execution of the case scenario, the model runs and produces traces to show its activities. Figure 3 shows the priority value of each of the goals over time. At timestep 0, the model is executing the *WatchStorm* goal. At timestep 1, the virtual pilot notices the system malfunction, so the goal *HandleSystemMalfunction* becomes active. The initial priority of the goal is higher than the current priority of *WatchStorm*. The model does not switch goals however, since the priority added with additional task switch costs is clearly higher than the priority of *HandleSystemMalfunction*. At this point in time, cognitive lockup occurs. At timestep 3, the total priority of *HandleSystemMalfunction* exceeds the total of *WatchStorm*. This is the case because additional priority is added if a goal is active for some time but not selected. The execution of *WatchStorm* is interrupted and *HandleSystemMalfunction* is started. Watching the storm is still relevant, so the goal stays active and can be executed further on in the scenario.

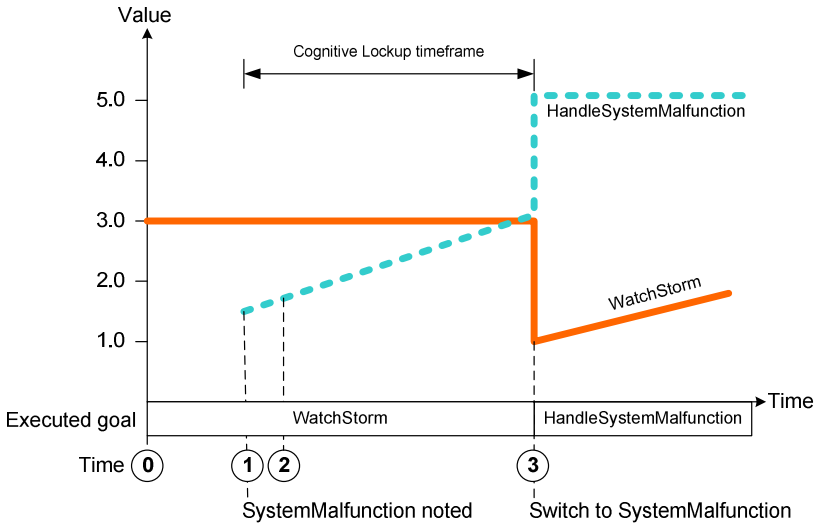

Figure 3: Goal priorities during storm-avoidance scenario.

The output of the model shows the occurrence of cognitive lockup. It prevents the model to switch goals immediately, but instead the model chooses to continue pursuing the current goal.

Conclusion

In this paper we have discussed methodological and developmental aspects of modelling cognition in complex task environments. In particular, we have argued for a flexible development approach, enabling iterative design of the cognitive model, tailored to realistic settings, and led by human factors knowledge.

For this purpose, we adapted the situated Cognitive Engineering development approach, and presented a modular goal-based support tool for implementing cognitive models. We believe that the combination of these two frameworks have been successful in deriving and modelling the aviation scenarios in which cognitive lockup was a source of human error.

In the future, we would like to perform more development iterations of the cognitive model. Also, we intend to perform more thorough testing of the cognitive model by comparing simulated behaviour traces with real pilot behaviour. This allows us to better incorporate the *lessons learned* from the previous iteration in the next version of the cognitive model.

Acknowledgment

The work described in this paper is funded by the European Commission in the 7th Framework Programme, Transportation under the number FP7 – 211988. This study is part of the research program "Autonomous Training" (V1023) under contract for the Netherlands Department of Defense.

References

[1] CLIPS: a tool for building expert systems. http://clipsrules.sourceforge.net/

[2] Diggelen J. van, Beun R.J., Dignum F., Eijk R.M. van, Meyer J.-J.Ch., *ANEMONE: An Effective Minimal Ontology Negotiation Environment*, Proceedings of the Fifth International Conference on Autonomous Agents and Multi-agent Systems (AAMAS06), ACM Press, pp. 899-906, 2006

[3] Lüdtke, A., Osterloh, J.-P., Mioch, T., Rister, F., Looije, R. (2010). Cognitive Modelling of Pilot Errors and Error Recovery in Flight Management Tasks. In *Human Error, Safety and Systems Development*, LNCS5962, Springer

[4] Mioch, T., Osterloh, J.-P., Javaux, D. (2010). Selecting Human Error Types for Cognitive Modelling and Simulation. Under review.

[5] Neerincx, M.A., Lindenberg, J., Smets, N.J.J.M., Bos, A., Breebaart, L., Grant, T., Olmedo-Soler, A., Brauer, U., & Wolff, M. (2008). The Mission Execution Crew Assistant: Improving Human-Machine Team Resilience for Long Duration Missions. Proceedings of the 59th International Astronautical Congress (IAC2008), Glasgow.

[6] Rasmussen, J. (1983). Skills, rules, knowledge: Signals, signs and symbols and other distinctions in human performance models. *IEEE Transactions: Systems, Man and Cybernetics,SMC-13(3)*, 257–266.

[7] Russel, S., Norvig, P. Artificial Intelligence, a modern approach (2003), Prentice Hall publishers.