Towards Cooperative Cognitive Models in Multi-Agent Systems

Jan Charles Lenk*, Rainer Droste†, Cilli Sobiech‡, Andreas Lüdtke§ and Axel Hahn¶
*‡§*Transport Research Division*
OFFIS Institute for Information Technology
Escherweg 2, 26121 Oldenburg, Germany
Email: {lenk, cilli.sobiech, andreas.luedtke}@offis.de
†¶*Department of Computing Science*
University of Oldenburg
26111 Oldenburg, Germany
Email: {rainer.droste, axel.hahn}@uni-oldenburg.de

***Abstract*—Cognitive models capture the behavior of human agents and allow the assessment of risks by prediction of human performance in simulated hazardous environments. In this paper, a combined approach of bottom-up cognitive modeling and top-down operation process modeling is proposed to facilitate the psychologically plausible modeling of complex operations involving multiple human agents. Hereby, we aim at using process and cognitive models productively in the planning and a priori simulation of concrete scenarios to improve safety in offshore operations. We demonstrate our modeling approach with an exemplary multi agent scenario from the maritime domain and discuss its possible application in an operation planning tool aimed at domain experts.**

***Keywords*-Cognitive Modeling; Multi-Agent System; Simulation of human behavior**

I. Introduction

Cognitive modeling aims at creating models of cognitive processes of individual human agents. A common approach is to define a cognitive model as a set of production rules, which implement human behavioral procedures, enabling it to react on changes and manipulate states in its environment. Among the prime benefits of cognitive modeling are executable models which capture the behavior of a human agent interacting with a simulation environment. For instance, cognitive models hereby allow risk assessment by prediction of human performance in concrete simulated hazardous situations.

Many real-life situations demand complex interaction between multiple human actors. Accordingly, also the simulation of such situations requires an integration of multiple simulated human agents and their cognitive processes. Whereas the interaction between agents can be represented as Multi-Agent Systems (MASs), cognitive modeling is so far limited to MAS scenarios from domains where interaction is carried out in a highly compulsory manner, e.g., aeronautics, where aircraft crews have to follow precise procedures prescribed by international aviation laws. This is not the case in other domains such as the offshore wind industry, where personnel also follow procedures, but cooperate in a less formalized way and rely on generic guidelines. In this paper, we combine bottom-up cognitive modeling with top-down operation process modeling. Dynamic properties of MAS scenarios may be described conveniently with operation process models. While these can specify activities and inter-agent communication for a successful cooperative solution of the scenario, they do not state how the individual human agent solves its tasks and thus cannot be employed within simulations in the same ways as executable cognitive models.

Using operation process models for the specification of inter-agent cooperation and Hierarchical Task Analysis for normative individual behavior, we introduce a modeling paradigm as a combination of high-level modeling languages for cooperative cognitive models embedded in multi-agent scenarios. The process information is derived from expert knowledge, while human behavior is extracted from training manuals and experiments. The exemplary scenario from the maritime domain in this paper demonstrates the combination of the two different modeling languages and their future implementation in a human factors tool for domain experts.

We shall first provide an introduction to cognitive architectures in Section II, then to our scenario and modeling languages in Section III, and finally, present the mappings between these languages and exemplary conforming cognitive models in Section IV.

II. Related Work

A cognitive architecture is itself a model of the human mind, as well as its sensors and actuators. It provides a framework which imposes constraints on the modeler to prevent unrealistic models of human cognitive processes [1]. The most prominent of the cognitive architectures is ACT-R (Adaptive Control of Thought-Rational) [2], yet other more specialized architectures are built upon the same principles, for instance CASCaS (Cognitive Architecture for Safety Critical Task Simulation), which has been applied for pilot and driver modeling [3]. Even if they differ in their purpose and some other aspects, the symbolic elements

 ISBN: 978-1-61208-218-9

of a production system are common to both ACT-R and CASCaS.

Task execution is guided by *goals* and *subgoals* eventually. *Declarative knowledge* is encoded in the form of *memory variables* or *chunks*, whereas *procedural knowledge* is encoded in the form of *production rules*. These consist of a *conditional part* or *left-hand side* (LHS), which includes conditions on goals and other variables and triggers the execution or *firing* of a rule. The *action part* or *right-hand side* (RHS) of a rule is executed and may add new goals to the agenda or manipulate external states.

The environment is represented by *environment variables*, which provide an abstract view on world entities, e.g., concrete human machine interfaces. Thus, the cognitive model is able to perceive and manipulate external world states via the environment variables. The cognitive model is employed within simulations to predict human performance, such as response times and error rates, or cognitive workload and bottlenecks for specific tasks. It also may be used to evaluate interface design with a model-based approach [4].

A. High-Level Languages for Cognitive Modeling

For more complex tasks, cognitive models may easily contain several hundred rules. Maintenance and extension of a complex model is a tedious process. The need for high-level languages and tooling for cognitive modeling is evident, especially if task modeling is carried out by practitioners either in cooperation with an experienced modeler or alone. Several criteria for such a modeling language have been formulated [5]. Examples of high-level languages include the textual Herbal [6] and HTAmap [7], as well as a graphical language and prototypical editor [8].

III. Methods

In this section, we shall describe the scenario and our modeling languages.

A. Experimental Scenario

As domain for the scenario in this paper serves the offshore construction domain. A 940-ton quadpod has to be moved from the construction vessel to its future location with a heavy lift crane as foundation for a wind turbine. Apart from the banksmen who perform the actual rigging of the load, two further agents also participate in the scenario: the crane operator and the lift supervisor.

A simulation (Fig. 1) of the scenario in a quasi-experimental setting with experienced participants took place in the heavy lift simulator at the MariKom (Maritimes Kompetenzzentrum Wesermarsch) at Elsfleth, Germany. Processes and procedures were derived from the observations made during the simulation.

During the simulation, operator and supervisor communicated over the wireless. First, the crane operator had to exert a gentle pull of just about over 800 tons on the load as instructed by the lift supervisor. The supervisor stopped the lifting to check the rigging of the load, and gave order to slowly lift the load until the target height was reached. The crane operator had to follow the supervisor's instructions, as well as to monitor and report the weight of the load as tons on the hook, which were displayed in a mimic diagram in the crane's cabin. This critical information was hidden from the supervisor and had to be communicated by the operator.

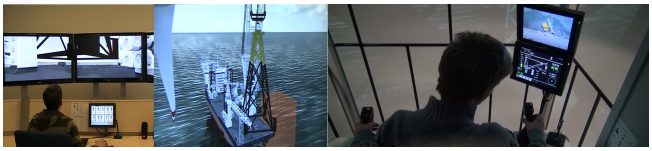

Figure 1. Heavy Lifting simulator scenario. From left to right: Lift Supervisor, Crane and Load, Crane Operator

B. Top-Down Operation Process Modeling

In the business community, suitable concepts and tools for business process specification provide an easily accessible way to identify, systematically describe, and measure processes. For the MAS process model, we have derived the main concepts from the Business Process Modeling Notation (BPMN) language [9]. Thus, we were able to describe and structure the respective processes in order to map different participants representing different interacting agents and their tasks. Fig. 2 shows the cooperative activities of the process step *Load Lifting* involving two **Participants** represented by two **Swimlanes**. The process step starts and ends with **Events**; **Sequence Flows** facilitate a scheduling of the different **Activities** performed by the Participants. Different types of Activities can be further distinguished: Send and Receive Tasks for e.g., reporting weight indicate communication and interaction between different Participants. **Message Flows** describe origin and destination of messages, e.g., reports or commands, in order to synchronize the **Activities** within the process. Resources (e.g., load) are associated to an **Activity** as **Data Objects**.

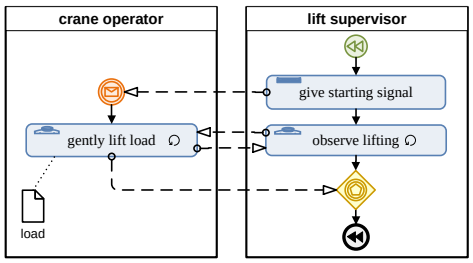

Figure 2. Cooperative activities of the operation process step *Load Lifting*

C. Bottom-Up Procedure Modeling

The Procedure Editor (PED) enables rapid prototyping of cognitive models with a high-level modeling language.

 ISBN: 978-1-61208-218-9

Based on Hierarchical Task Analysis, behavioral models for normative procedures may be defined using a simple graphical notation. Several procedures may be composed to an integrated cognitive model of a human agent.

PED is intended to be a tool to the experienced modeler, as well as to the domain expert with little experience in cognitive modeling. The graphical task definition language provides a high-level abstract view from the concrete procedural and declarative structures needed otherwise for a particular production system. Models may be simulated directly in PED , which shortens the evaluation cycle for the inexperienced modeler. Furthermore, models are validated constantly during the editing process to ensure formal constraints. PED has been used already to model tasks from the automotive and aeronautic domains. The PED language is a subset of CASCaS, although procedures created with PED could also be translated into models for other cognitive architectures. The main elements of the PED language are:

- **Goals**: The control structure of a model consists of goal hierarchy. Goals are executed by being instantiated and expanded onto the goal agenda, a stack-like structure. The last goal on the goal agenda becomes the active goal. Usually, a procedure has an entry goal which starts procedure execution.
- **Rules**: Each regular rule is fired if its top goal is active goal on the goal agenda and if the conditions in its LHS evaluate to true. The conditions are made on **Memory Variables**, which often serve as internal representation of **Environment Variables**. Thus, memory retrievals are necessary to check the condition on the rule.
  - **Percept rules** are also triggered by their top goal, but only if an environment variable is not encoded in a memory variable needed for one of the regular rules below the same top goal.
  - **Reactive rules** can fire at any time if their conditions match, thus they have no top goal. Rather, they listen to changes in the environment variables which are referenced in their LHS.
  - **LHS elements** consist of **Memory Read** items, to retrieve memory variables, and **Conditions** on the variables.
  - **RHS elements** are actions executed when the rule itself is executed. These are **Memory Store**, **Motor**, and **Voice** actions. The first assigns new values to memory variables, the latter two enable direct manipulation of environment variables.

IV. IMPLEMENTATION

The results of our combined top-down and bottom-up modeling approach are the mappings between both languages and their application in the cognitive models for the heavy lift scenario from Section III-A.

A. Mappings between Process and Procedure Languages

Elements from the process language (Section III-B) are mapped onto elements and structure patterns of the PED procedure language (Section III-C). Thus, if a process is specified first, procedures may be validated against the interface defined by the process, i.e., whether control and message flow, event handling, and activity allocation is implemented in the low-level procedures. The mappings are not yet complete and shall be subject to further modifications and extensions.

- **Activity $\longmapsto$ Procedure**
  Procedures are allocated to Agent Activities according to Swimlane
- **Signal $\longmapsto$ Environment Variable**
  The variable shall represent a concrete property of the system, e.g., an indicator light.
- **Error $\longmapsto$ Environment Variable**
  The variable shall represent an abstract faulty state of the environment.
- **Message $\longmapsto$ Environment Variable**
  The variable shall represent a communication channel, e.g., a wireless channel.
- **Intermediate Catch Event preceding Activity $\longmapsto$ Reactive Rule in Procedure**
  The Reactive Rule is the top most element of the Procedure. Intermediate Catch Events may be used with Messages, Signals, and Errors.
- **Sequence Flow between Activities $\longmapsto$ Top goal present in second Procedure**
  Upon termination of the first, the top goal of the second Procedure is put on the goal agenda.
- **Sequence Flow to Gateway $\longmapsto$ Top goal in succeeding Procedures**
  Control Flow to succeeding Procedures is constrained by conditions.
- **Incoming Message Flow in Activity $\longmapsto$ Listening Rule Sequence inside Procedure**
  The procedure has to listen for a message on an environment variable representing the communication channel.
- **Outgoing Message Flow from Activity $\longmapsto$ Motor or Voice action in procedure**
  There has to be a corresponding action in the procedure.

B. Example Bottom-Up Procedure Models

From the top-down process in Fig. 2, two procedures were selected and modeled bottom-up in PED, applying the mappings in Section IV-A and according to observations made during the simulation: *Gently Lift Load*, executed by the crane operator, and *Observe Lifting* (Fig. 3) for the lift supervisor. The process itself states that the two agents exchange messages and information, but not how and when. This is specified in the structure of the procedures. *Gently Lift Load* is initiated by a message event. For PED elements, this maps to a reactive rule which listens on the wireless channel for the starting command. In the *Observe Lifting* procedure, the supervisor listens in the first loop structure to the weight on the hook in tons as announced by the crane operator and issues a halt command when the load is barely lifted.

If the rigging holds firmly, the supervisor commands the operator to continue lifting until the desired altitude is reached. Here, both procedures terminate for this part of the scenario. Both procedures are executable and may be validated against the process, but also within the simulation when connected directly to the heavy lift simulator.

 ISBN: 978-1-61208-218-9

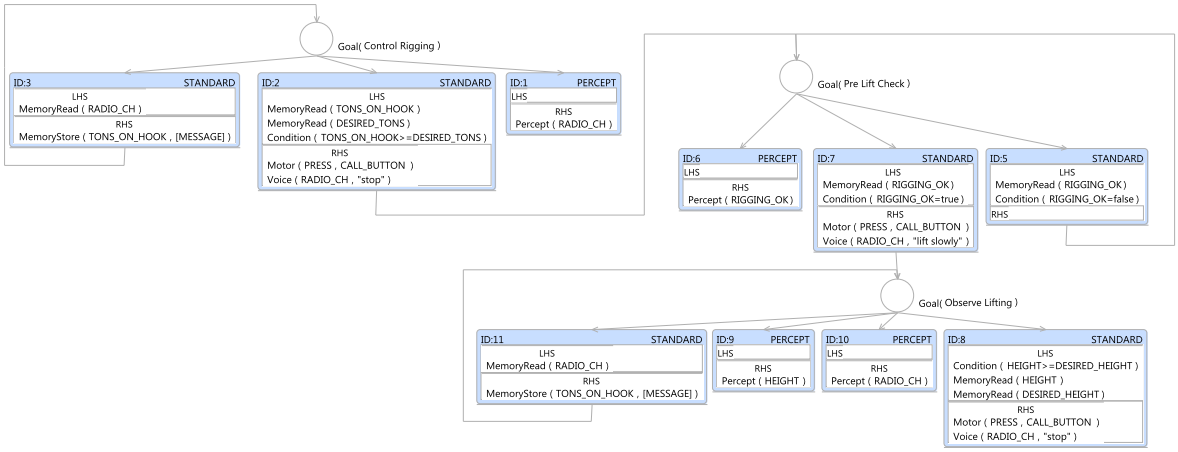

Figure 3. *Observe Lifting* procedure for the lift supervisor

V. OUTLOOK

Our combined top-down and bottom-up modeling approach enables the fast prototyping of executable cognitive models for complex scenarios involving multiple agents. These can be used to assess cognitive workload for specific, highly cooperative, tasks. Process and procedure models may be transformed into state automata as input for formal model checking with the goal of safety analysis.

We aim at using process and cognitive models productively in the planning and a priori simulation of concrete scenarios to improve safety in offshore operations. For this, we will enhance the Procedure Editor with a process-based MAS editing component. Future domain-specific adaptations of the process language shall also consider physical aspects, such as constraints on perceptual and manipulative capabilities due to agent location. We also consider automatic generation of procedures from processes for our tooling. The domain expert thus shall be enabled to prototype and eventually simulate complex operations in advance for increased efficiency. PED procedures are easily transformed into executable cognitive models for the CASCaS architecture. Thus, safety critical states in an operation may be identified, either by formal analysis, or by prior simulation of the process and its associated procedures within a simulation environment.

ACKNOWLEDGMENT

This modeling approach has been developed within the SOOP (Safe Offshore Operations) project funded by ERDF. The authors would like to thank Prof.-Ing. H. Korte, B. Zerhusen, J. Richter, and I. Ihmels for their assistance during the simulator experiments. We would like to thank MariKom, Elsfleth, Germany for project support.

REFERENCES

[1] W. D. Gray, *Integrated Models of Cognitive Systems*. Oxford: Oxford University Press, 2007.

[2] J. R. Anderson, D. Bothell, M. D. Byrne, S. Douglass, C. Lebiere, and Y. Qin, "An integrated theory of the mind," *Psychological Review*, vol. 111, pp. 1036–1060, 2004.

[3] A. Lüdtke, L. Weber, J.-P. Osterloh, and B. Wortelen, "Modeling pilot and driver behavior for human error simulation," in *Digital Human Modeling. Second International Conference, ICDHM 2009, Held as Part of HCI International 2009*, ser. Lecture Notes in Computer Science, V. G. Duffy, Ed., vol. 5620/2009. Springer, 10 2009, pp. 403–412.

[4] F. Ritter and R. Young, "Embodied models as simulated users: Introduction to this special issue on using cognitive models to improve interface design," *International Journal of Human-Computer Studies*, vol. 55, pp. 1–14, 2001.

[5] F. E. Ritter, S. R. Haynes, M. A. Cohen, A. Howes, B. John, B. Best, C. Lebiere, R. M. Jones, J. Crossman, R. L. Lewis, R. St. Amant, S. P. McBride, L. Urbas, S. Leuchter, and A. Vera, "High-level behavior representation languages revisited," in *Proceedings of ICCM - 2006- Seventh International Conference on Cognitive Modeling*, 2006.

[6] M. Cohen, F. E. Ritter, and S. Haynes, "Herbal: A high-level language and development environment for cognitive models in soar." in *Proceedings of the 14th Conference on Behavior Representation in Modeling and Simulation*, 2005.

[7] M. Heinath, "High-level ACT-R modeling based on SGT task models," in *Proceedings of the 9th International Conference of Cognitive Modeling*, 2009.

[8] J. C. Lenk and C. Möbus, "An MDA high-level language implementation for ACT-R," in *Proceedings of KogWis 2010: 10th Biannual Meeting of the German Society for Cognitive Science*, J. Haack, H. Wiese, A. Abraham, and C. Chiarcos, Eds. Universitätsverlag Potsdam, 2010, p. 137.

[9] "Business process model and notation (BPMN)," Object Management Group, 2011, [retrieved: 05, 2012]. [Online]. Available: http://www.omg.org/spec/BPMN/2.0/

 ISBN: 978-1-61208-218-9