Proceedings of the 2009 IEEE International Conference on Systems, Man, and Cybernetics
San Antonio, TX, USA - October 2009

Simulation of Learning and Planning by a Novel Architecture for Cognitive Technical Systems

Dennis Gamrad and Dirk Söffker
Chair of Dynamics and Control
University of Duisburg-Essen
Duisburg, Germany
dennis.gamrad@uni-due.de, soeffker@uni-due.de

***Abstract*—A novel architecture for Cognitive Technical Systems with a homogeneous knowledge representation for all cognitive functions is presented. The approach is methodically based on Situation-Operator-Modeling and implemented with high-level Petri Nets. It is not restricted to certain application fields and can be combined with other AI methods. By the combination of several instances of the architecture, the complexity of the represented interaction can be reduced by a problem-oriented encapsulation of different action spaces. The contribution is focused on learning of interaction in and with the outside world and its inner structure, which is illustrated by the control of an arcade game.**

***Index Terms*—Learning systems, Mental models, Intelligent agent interaction, System modeling, Planning**

I. Introduction

Autonomous systems like mobile robots are able to localize and plan their actions by themselves (c.f. [1]). However, if the environment is dynamic or (partially) unknown, humans can perform better than technical systems due to their learning abilities. Hence, the field of Cognitive Technical Systems is focused on the investigation of cognitive-inspired control. For this, a representational level for knowledge is the key feature to describe the interaction in and with the outside world as well as its inner structure.

In this contribution, the concept and implementation of a novel architecture for Cognitive Technical Systems is presented. The representation of knowledge as well as the whole information processing is based on the Situation-Operator-Modeling (SOM) approach (cf. [2], [3]). Therefore, Reference nets (a special high-level Petri Net formalism) and a related software tool (Renew) are applied (cf. [4]). Such as SOM itself, the architecture is a meta model which can be combined with other AI methods. The main difference to other approaches (cf. [5]–[7]) is the SOM-based representation of knowledge with a homogeneous format for planning (action model) and perception (perception model). Both models can be edited by the designer or learned automatically from interaction. Furthermore, several instances of the proposed architecture can be connected to reduce the complexity of the represented interaction.

II. Modeling and simulation of cognition

According to Strube [8], cognitive systems are characterized by the ability to represent outside relations internally, which is the basis for cognitive functions and procedures like learning, anticipation etc. This allows flexible behaviors and yields in the consequence to what is usually called intelligence. A suitable structure (or format) for a representational level is Situation-Operator-Modeling which allows to structure and to reduce the complexity of interaction with the outside world [3].

Within the SOM approach the processes within the real world are considered as sequences of scenes and actions, which are modeled as situations (time-fixed description of the considered system or problem) and operators (changes within the considered system) respectively. A situation consists of characteristics and relations. In technical systems, the characteristics can be physical values measured by the sensors. The relations represent the inner structure of the situation by linking the characteristics to each other through arbitrary functions. The operators have the same quality as the relations of the situation. An operator transfers a situation to another. Depending on its functionality, the characteristics, the relations or both can be changed. The condition whether an operator can be applied is described by the operator's assumptions. An operator on a higher hierarchical level can be build by the combination of several operators termed as meta operator. In [9], the SOM approach has already been implemented to a mobile robot as a basis for its knowledge representation and information processing.

Due to structural similarities, SOM-based models can be represented by high-level Petri Nets (HPNs), which may be modeled graphically, simulated, and analyzed by established software tools. The SOM situation can be represented by a place with one or several tokens and the SOM operator can be related to the transition with guard functions and current bindings. In this contribution, the software Renew (Reference net workshop) for Reference nets, a special HPN formalism, is applied. As a special feature, Renew allows Java objects and nets as tokens. However, it does not provide automated state space generation or own analysis functions, like other common HPN tools [10]. Hence, these functionalities have to be realized by the model itself.

III. Conceptual fundamentals

The cognitive functions planning, perception, and learning as well as the interaction with the environment and the suitable representation of knowledge are strongly connected

978-1-4244-2794-9/09/$25.00 ©2009 IEEE

to each other [2]. In the following, a cognitive architecture is described, which provides a unique and homogeneous SOM-based representational level for all cognitive functions. Furthermore, the whole information processing is based on the same methodical background. Additionally, the architecture allows the integration of other AI methods for planning, learning, etc. As a special feature, several instances of the architecture can be combined.

A. Structure of the cognitive architecture

The proposed cognitive architecture (cf. Fig. 1) comes also with the known three levels for skill-based, rule-based, and knowledge-based input/output behavior [11]. The connections to the sensors and actuators of the technical system to be controlled (ego system) are realized by sensing and execution modules. The input from the sensing module (represented as situation) is further interpreted by a perception module applying relations stored in a perception model. The actions of the ego systems as well as the dynamic of the outside world (other agents etc.) are represented in an action model in detail. The action model is used as a basis for the calculation of a mental action space which can also be extended from interaction directly. The combination of action model, mental action space, and perception model builds the mental model of the architecture. The planning module gets a goal situation and the current interpreted situation as inputs and uses the mental action space to generate a plan containing a sequence of operators from the current situation to the desired situation. The execution of plans is organized by the execution module. During the execution of a plan or during random interaction as well, it is checked whether the mental model corresponds to the reality. If differences between the mental model and the reality are detected, the mental model can be modified accordingly by two modules for the learning of operators and the learning of the situation. Furthermore the learning modules are used to interpret the previously learned knowledge for generalization and complexity reduction.

B. Combination of several architectures

Often, the human interaction is divided into several action spaces differing with respect to the characteristics which have to be considered and the actions which can be performed. Furthermore, the human planning is a complex process leading to a hierarchical plan. The different steps of the plan are not detailed from the very beginning. They are detailed and modified dynamically if the human tries to reach the next sub goal.

For the representation of the described human planning and interaction behavior several instances of the proposed cognitive architecture can be combined in parallel or hierarchical. The term instance denotes a derivation of a general (template) architecture (according to the relation of object and class in computer science). Hence, the complexity of the real world can be reduced by a problem-oriented encapsulation of several (independent) action spaces. Each single instance has separate action and perception models.

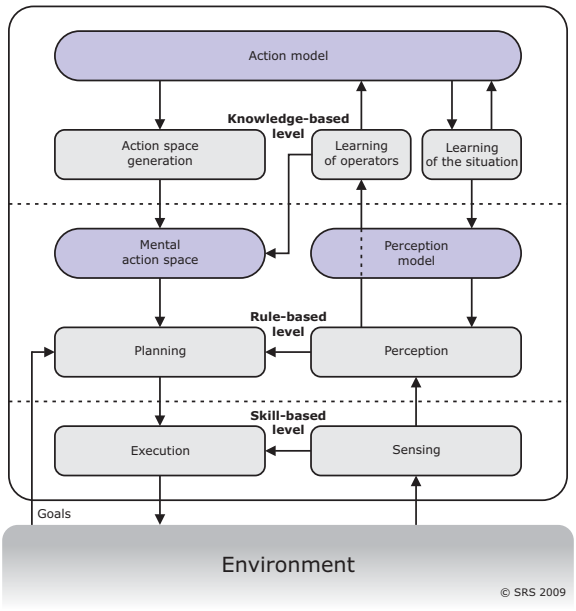

Fig. 1. Structure of the cognitive architecture

IV. Implementation

This section describes the interaction modeling, knowledge representation, planning, execution, sensing, perception, and learning of the proposed architecture in detail. As described in Section II, the SOM approach allows the modeling of interaction between humans or intelligent systems and their environment and due to its similarities to high-level Petri Nets it can be implemented with related tools. For the implementation of the learning and planning functions, the action and perception models have to allow modifications (not restricted to simple data types) and analysis (e.g., based on a state space). Although Renew does not provide state space generation or other analysis functions, it is chosen still because of its open structure and flexible programming alternatives.

A. Modeling of interaction

The action and perception models of the proposed architecture are represented by SOM-based Reference nets implemented with the high-level Petri net simulator Renew. The model can be extended during simulation by synchronous channels, allowing the communication between nets, which are hierarchized by a net-in-net formalism [12]. New nets consisting of places, transitions etc. can be loaded as tokens in super nets. Hence, the models representing the interaction and structure of the real world can be modified and restructured in a flexible manner. In the following, the representation of situations, characteristics, relations, and operators are described in detail.

A situation is represented by the class `Situation` contained in the Petri-Net-model as token. The characteristics of the situation are defined by a list attribute containing a list of a

class `Characteristic` (cf. [9]) consisting primarily of the attributes `Name` and `Value`. In contrast to [9], the relations are not attributes of `Situation`, but they are linked indirectly through a list of characteristics (assumptions of the relation). The assumptions of a relation define whether a relation belongs to a situation or not (different to previous implementations of SOM).

The relations and operators are represented by special nets (operator nets). An operator net (cf. Fig. 2) gets an initial situation as input and creates a final situation depending on its function and assumptions. It can be connected to additional operator nets in a sequence (different to the term meta operator) to calculate the effects to the initial situation successively. The function and assumptions can be set during modeling as a basic mental model or during runtime to learn the interaction with the environment autonomously.

The assumptions of operator nets can be represented positively or negatively by lists of the class `AssumptionList`. A positive assumption is a condition which has to be fulfilled and a negative assumption is a condition which has to be not fulfilled. Sometimes a single situation has to be set as an assumption and sometimes it has to be set as an exception or restriction. If the assumptions are not fulfilled, the operator net does not change a situation (initial situation is equal to final situation). The class `AssumptionList` contains a list of the class `Assumption` with the same quality as the class `Characteristic` and the integer value `SampleSize` as attributes. The attribute `SampleSize` stores how often a certain positive or negative assumption is added from the learning module. Thereby, the dynamic of the environment or the incompleteness of the measured scene is taken into account.

The function of an operator net is implemented by the combination of several 'function nets' changing the parameters of the characteristics. Hence, the whole function of an operator or the whole structure of a situation respectively is the combination of the functions of those operator nets with fulfilled assumptions. A function net contains an arbitrary function which outputs a list of the class `Characteristic` (characteristic list) and gets a list of parameters and two characteristic lists as input. One of the two characteristic lists is also used to parameterize the function and the other sets the characteristics for the output.

Also the interaction with dynamical environments can be represented with the proposed approach. For this, the whole observable dynamics of the environment is defined as several 'dynamic elements' which are independent from each other and the ego system. In the real world, a dynamic element could be everything which acts independently from the ego system. The behavior of each dynamic element is represented by an operator net describing a waiting action of the ego system (waiting operator) and characteristics which are mainly influenced by the related dynamic element.

B. Knowledge representation

The representation of knowledge is one of the key features for cognitive systems. The facts and connections of the outside world are represented internally with a suitable structure in a mental model. Here, the Situation-Operator-Modeling approach is used as common format for the representation of knowledge, interaction, and information processing. This leads to a clear structure of the architecture and simplifies the understanding as well as the further development. As described in Section IV-A, the knowledge of the cognitive architecture is stored in the mental model consisting of action model, mental action space, and perception model.

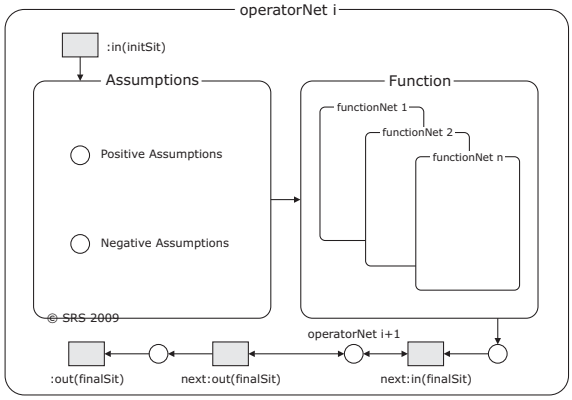

Fig. 2. Structure of an operator net

The actions of the ego system and the dynamic of its environment are represented in the action model by operator nets as well as in the mental action space by so called experiences [9]. Although both formats are different, they have the same methodical background and can be transferred into each other. The action model represents in detail how, in which way, and under which conditions an operator changes the characteristics of a situation. The representation as experience is a tuple $(s_{\text{initial}}, n_{\text{operator}}, s_{\text{final}})$ consisting of an initial situation s_{initial}, the name of the operator n_{operator}, and a final situation s_{final}. In contrast to the action model, the operator is not described in detail. Here, experiences represent the effect of an action to one special scene, rather than the general effects related to different scenes.

Due to the fact that the action model represents the effect of the operators in more detail, but experiences are easier to process, here the action model is transferred to a set of experiences. A space of all possible actions and situations, stored as experiences, is generated. Because a mental action space of complex systems can be extreme large or even infinity, partial mental action spaces can be generated based on the current situation [13]. Besides the generation of experiences from the action model, experiences can also be generated from interaction, to explore the effects of the ego system's actions in the environment or the dynamic of the environment itself (waiting operators).

The perception model contains operator nets from the same quality as the ones in the action model. These operator nets represent the relations and are linked to the situations by their assumptions. Due to the fact that the relations can not

be measured, new operator nets can only be added through the designer in advance (human interprets structure of the situation) or through learning from experiences (application of pattern recognition methods). The operator nets are used from the perception model to interpret situations.

C. Planning, execution, sensing, and perception

The cognitive function planning is the a priori mental simulation of following actions based on the mental model allowing the performance of goal-directed behavior. Here, the planning module uses the experiences stored in the mental action space to generate a plan from the current situation to a given goal situation. The plan as the output of the planning module is the input of the execution module performing the actions corresponding to the plan.

For planning different search or optimization algorithms can be implemented. They get the set of experiences and a goal represented by a situation as input. The output is a sorted sequence of experiences. The initial and final situations are integrated in the plan for a later comparison with the reality.

The sensing and execution modules are interfaces between the cognitive architecture and the real or simulated environment. In technical systems, it is strongly connected to the hardware of a considered system. The execution of actions can be based on inputs from the sensing module (reflexes), inputs from the planning module (plans), or simple behaviors, like random or rule-based exploration.

The perception module gets situations from the sensing module and interprets them based on the operator nets from the perception model. For this, the assumptions of the operator nets are checked whether a relation can be applied or not. To one situation several relations can be applied. Contradictions are excluded categorically by a consistency check in the learning module. The situations coming from the sensing module consist exclusively of characteristics measured by the ego system's sensors. Through the application of operator nets, further virtual characteristics can be added to a situation (cf. Fig. 3). Hence, it is possible to derive problem-relevant information with a higher degree of abstraction from the measured characteristics and reduce the complexity of the considered system my means of the interpretation of the situation's structure.

D. Learning

Due to the fact that the environment is usually not static and partially unknown, a cognitive system has to be able to learn from interaction with the environment. By the cognitive function learning, it is able to extend and refine its mental model to improve its performance successively with respect to a given task. In the proposed architecture, the operators as well as the structure of the situation can be learned. For the learning of operators two different approaches are implemented, the learning of experiences (effects of an operator to a special situation) from the interaction and the learning of the operator's function and assumptions (effects of an operator to situations in general).

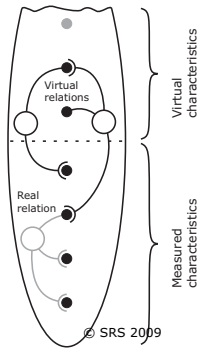

Fig. 3. Measured and virtual characteristics

If experiences are learned from interaction, the initial and final situations in combination with the performed operator are stored in the mental action space. This can be used for the exploration of unknown parts of the environment or for the observation of the dynamic of the environment (waiting/observation operator). The learned experiences can be combined with the experiences calculated from the action model to refine the knowledge in certain parts of the environment, which are not represented by the operator's function and assumptions in detail. Furthermore, removing of experiences from the action model provides a fast possibility to renew a plan. In this case, the difference between the action model and the reality is ignored, which can make sense if the differences are not large. If the differences are too large or if they should not be ignored due to other reasons, the action model has to be modified.

The variable net structure of the action model enables the learning of the operator's function and assumptions. All elements of the action model including operator nets, function nets, and assumption lists can be created or modified. For this, the learning module gets experiences (new or in contrast to the already known experiences) from interaction as inputs and generates modifications in the action model as output. In the following example (cf. Fig. 4), the learning of operator's functions and assumptions are illustrated.

A situation s_i consists of the three characteristics c_1, c_2, and c_3. The assumed function of the operator o_1 in the action model adds 1 to c_1 and do not change c_2 and c_3. Then, during interaction in the real world o_1 adds 1 also to c_2 of the situation s_1. Now, the learning module detects the difference between the assumed and the real operator o_1. A new operator net can be created, which is linked after the previous operator net (first operator net). The first operator net gets the situation s_1 as negative assumption and the second operator net gets s_1 as positive assumption. Additionally, the second operator net gets a function net, which adds 1 to c_2. After the calculation of a new mental action space, the refined mental model can be used for planning.

Beside the learning of experiences and operators' functions and assumptions, the sample size of the assumptions and the effort of operators related to the needed time or energy

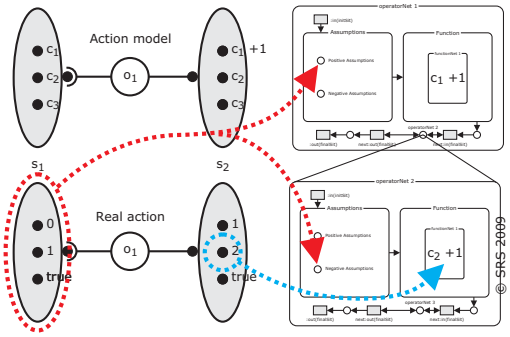

Fig. 4. Visualization of the learning example

can be learned, too. Furthermore, a sequence of operators can be combined to a meta operator. They can be used as a representation of successful plans or frequently performed action sequences.

To reduce the amount of stored positive and negative assumptions, different interpretation methods can be applied. Hence, the calculation time, to generate the mental action space can be also reduced. Furthermore, the knowledge about effects of operators to a few special situations can be generalized to enhance the knowledge about the environment without exploration.

The interpretation of one or several characteristics can be represented by operator nets. Before interpretation, the (partially) learned assumptions have to be filtered/reduced to those which are truly relevant. Then, the relevant characteristics can be analyzed in detail. For example, a threshold function can be implemented, which creates a boolean characteristic depending on the characteristic to be analyzed and a parameter. Finally, the new relation is stored in the perception model and the assumptions of the corresponding operator are replaced or extended by the new characteristic.

A possible SOM-based method to derive new knowledge from previous knowledge is presented in [14] proposing an approach which allows learning explicitly from the occurrence of errors. During the planning process hypotheses about the assumptions and functions of known operators are used to bridge the lack of knowledge. The hypotheses are inspired by human errors according to Dörner's [15] classification, which is related to the interaction within complex dynamical systems. Afterwards, in the execution phase the hypotheses are checked. The deviation between the supposed effect and the real effect is used to extend and to refine the mental model.

V. Experimental Example

As an example, an arcade game application [16] is chosen, where an agent interacts in a grid-based environment. It is clear to illustrate the used functions and can be extended arbitrary to get a higher grade of complexity. The environment consist of different kinds of fields and the agent can perform the actions 'up', 'down', 'left', and 'right'. In general, the task of a human operator consists of first picking up a certain number of 'emeralds' or 'diamonds', by entering related fields, and then finishing the level by leaving the scenario through an exit door (cf. Fig. 5).

Here, the human operator is replaced by the proposed cognitive architecture. The situation s_i, as the input of the architecture, consists of the characteristics 'x-position' (`integer`), 'y-position' (`integer`), 'type of the current field' (`string`), 'collected points' (`integer`), and 'environment' (list of the class `Field`). The output can be a plan containing a sequence of operators, a single planed, or random operator. An interface program transfers the operators to actions and creates the current situation from the scene or state of the arcade game respectively.

A. Example scenario

In Fig. 5, an example scenario of an arcade game is shown. The agent has to pick up at least one emerald or diamond and leave the level by reaching the exit door in the lower right corner. All fields can be entered, besides the straight and crumbly walls.

The functions and assumptions of the operators are modeled roughly in the action model to enable goal directed behavior by the generation of plans. However, the crumbly wall and the diamond field are not modeled. They are represented in the situation by the characteristic 'environment' and it is assumed that they can be entered, but the functions of the operators related to these fields are not known in detail. Hence, if the agent enters a field with a diamond or crumbly wall only the changing of the characteristics 'x-position' and 'y-position' respectively are represented.

B. Simulation

The graphical representation of the calculated mental action space used in the simulation is shown in Fig. 6. According to the SOM symbolism, the situations are represented by the gray ellipses and the operators are represented by the arrows with dotted head. The names of the operators are arranged next to the arrows. If an operator comes several times in a sequence, this is abbreviated with `number*operator`.

At the beginning of the simulation the agent is in the lower left corner (s_1). The cognitive architecture gets the position of the exit door as goal situation (s_{57}). Due to the fact that

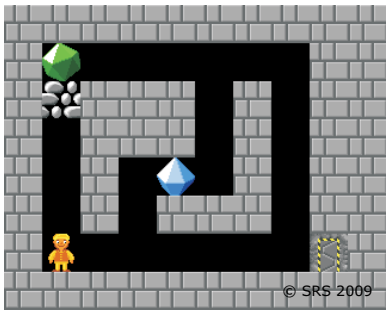

Fig. 5. Example scenario

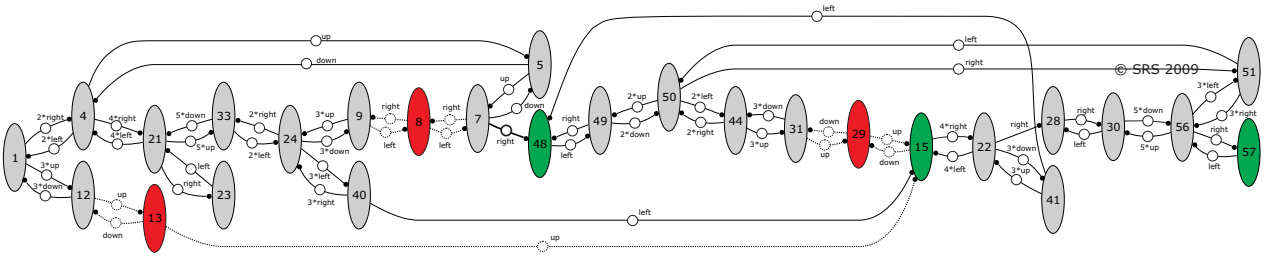

Fig. 6. Graphical representation of the calculated mental action space

the action model describes that the exit field can only be entered if the number of collected points is higher than zero, a plan is generated, which contains a path to the field with the emerald (s_{15}) and then to the field with the exit door (s_{57}). During the execution of the plan, it is detected that the field with the crumbly wall (s_{13}) can not be entered. Hence, the plan can not be executed further and a new one has to be generated. To avoid the same plan again, the wrong experience ($o_{up} : s_{12} \rightarrow s_{13}$) is removed from the mental action space. During the execution of the new plan (o_{down}, o_{down}, o_{down}, o_{right}, o_{right}, o_{up}, o_{up}, o_{right}, o_{right},...), the agent enters the field with the diamond. In this situation, the increasing of the characteristic 'collected points' was not contained in the mental action space. Hence, the action 'right' leads to the situation (s_{48}) instead of the previously calculated situation (s_8), which is not existing in the reality. Now, the agent does not have to enter the field with the emerald anymore and a new plan from the current field to exit door can be calculated. To improve the performance for a following simulation, also the last invalid experience ($o_{right} : s_7 \rightarrow s_8$) is removed from the mental action space. Furthermore, the new experience ($o_{right} : s_7 \rightarrow s_{48}$) is stored. If the simulation is started again, the agent plans directly to the field with the diamond and then to the field with the exit door.

C. Results

The simulation shows that the proposed cognitive architecture is able to refine the mental model in a suitable way to realize a goal-directed behavior. The mental model is used to enable planning and interaction with the environment, the differences between model and reality are detected, and learning is used to improve the performance.

VI. Summary and Outlook

The contribution describes the concept of a novel architecture for cognitive technical system, its implementation, and an example simulation. The architecture provides a homogeneous representation of knowledge for the cognitive functions planning, perception, and learning, which is based on Situation-Operator-Modeling and high-level Petri Nets. By the implemented learning modules and the combination of several instances of the architecture, the complexity of the represented interaction can be reduced.

The work in the future will be focused on the learning modules as well as on the implementation to real technical systems, like assistance systems or mobile robots.

References

[1] S. Thrun, W. Burgard, and D. Fox, *Probabilistic Robotics*. Cambridge: MIT Press, 2005.

[2] D. Söffker, "Systemtheoretic Modeling of the knowledge-guided Human-Machine-Interaction (In German)," Habilitation Thesis, U Wuppertal, 2001, published at Logos Wissenschaftsverlag, Berlin, 2003.

[3] ——, "Interaction of Intelligent and Autonomous Systems - Part I: Qualitative Structuring of Interactions," *MCMDS - Mathematical and Computer Modelling of Dynamical Systems*, vol. 14, no. 4, pp. 303–339, 2008.

[4] O. Kummer, F. Wienberg, and M. Duvigneau, *Renew - User Guide*, 2nd ed., University of Hamburg, Department for Informatics, Theoretical Foundations Group, Distributed Systems Group, Hamburg, July 2008.

[5] J. E. Laird and A. Newell, "SOAR: An Architecture for General Intelligence," *Artificial Intelligence*, vol. 33, pp. 1–64, 1987.

[6] D. Kieras and D. E. Meyer, "An overview of the EPIC architecture for cognition and performance with application to human-computer interaction," *Human-Computer Interaction*, vol. 12, pp. 391–438, 1997.

[7] J. R. Anderson and C. Lebiere, *The Atomic Components of Thought*. Lawrence Erlbaum Associates, 1998.

[8] G. Strube, C. Habel, L. Konieczny, and B. Hemforth, *Handbuch der Künstlichen Intelligenz*, 4th ed. Oldenbourg Wissenschaftsverlag, 2004, ch. 2 Kognition, pp. 19–72.

[9] E. Ahle, "Autonomous Systems: A Cognitive-Oriented Approach Applied to Mobile Robotics," Dr.-Ing. Thesis, University of Duisburg-Essen, 2007, published at Shaker Verlag, Aachen, 2007.

[10] K. Jensen, S. Christensen, and L. M. Kristensen, *CPN Tools State Space Manual*, University of Aarhus, Department of Computer Science, Aarhus, January 2006.

[11] J. Rasmussen, "Skills, rules, knowledge: Signals, signs, and symbols, and other distinctions in human performance models," *IEEE Trans. Syst., Man, Cybernet.*, vol. 13, pp. 257–267, 1983.

[12] R. Valk, "Petri nets as token objects: An introduction to elementary object nets," in *Application and Theory of Petri Nets*, ser. LNCS, J. Dessel and M. Silva, Eds., no. 1420, ICATPN '98. Lisbon: Springer, June 1998, pp. 1–25.

[13] D. Gamrad, H. Oberheid, and D. Söffker, "Automated Detection of Human Errors based on Multiple Partial State Spaces," in *MATHMOD 2009 - 6th Vienna International Conference on Mathematical Modelling*, 2009.

[14] D. Gamrad and D. Söffker, "Learning from Errors: A Bio-inspired Approach for Hypothesis-based Machine Learning," in *SICE International Conference on Instrumentation, Control and Information Technology*, Tokyo, Japan, August 20th - 22nd 2008, pp. 647–652.

[15] D. Dörner, *The Logic of Failure: Recognizing and Avoiding Error in Complex Situations*. Basic Books, 1997.

[16] (2004, 19th June) Artsoft Entertainment - Rocks'n'diamonds. [Online]. Available: "http://www.artsoft.org/rocksndiamonds/"