A GENERIC COGNITIVE SYSTEM ARCHITECTURE APPLIED TO THE UAV FLIGHT GUIDANCE DOMAIN

Stefan Brüggenwirth, Ruben Strenzke, Alexander Matzner and Axel Schulte
Institute of Flight Systems, Bundeswehr University, Munich, Germany

Keywords: Cognitive automation, Hybrid agent architecture, UAV flight guidance, Graph matching, Soar.

Abstract: We present an overview of our cognitive system architecture (COSA) with applications in the multi-UAV flight guidance and mission management areas. Our work is based on a modified version of the Rasmussen scheme, which is an established model for human behaviour in ergonomics and cognitive systems. We belief that modeling in close analogy with categories of human behavior simplifies human-machine interaction as well as the knowledge engineering process. Our hybrid agent architecture is comprised of a low-level, reactive layer with prestored procedures and a goal-oriented, deliberative layer that enables inference and dynamic planning. The first, fully functional implementation of our architecture used production rules and the Soar interpreter, enhanced with syntax extensions such as type-safety and class-inheritance specific to our modeling approach. We then developed a specialized inference algorithm based on graph matching, which natively supports these extensions and resulted in performance improvements over the original Rete algorithm of Soar. A major weakness of our current implementation still lies in its static planning functionality which is realized by a means-ends plan library. We discuss a concept that interleaves the planning process with knowledge about anticipated action outcomes, followed by an interpretation of projected future world states with respect to current goals. We illustrate this principle with a multi-UAV scenario.

1 COGNITIVE AUTOMATION IN A MULTI-UAV SCENARIO

In the following, a scenario is considered in which multiple simulated UAVs (Uninhabited Aerial Vehicles) have to accomplish a time critical and resource bounded military mission in a dynamic and continuous environment. The mission comprises flight planning, communication activities and payload operation tasks. We have implemented and evaluated a configuration in which a human team member steers one of the aircraft and controls an uncrewed team on a task-based level. The possible actions of the aircraft include selecting a flight plan, loitering, performing evasive maneuvers, payload management and sending messages to other participants like the request to fulfill a task. Experiments (Schulte et al., 2008) showed that an assistant system is required to aid the human operator of such a multi-UAV system to avoid human factors related problems (Rauschert et al., 2008).

In a different experimental setup a manned-unmanned teaming (MUM-T) mission was flown with human operators located aboard a simulated airborne helicopter. The results showed that it generates severe problems to guide the UAVs on a state-of-the-art waypoint-based level, rather than on an abstract, task-based level like in the system mentioned before (Uhrmann et al., 2009). Human decision makers and operators must trust in such a system of high autonomy, a fact that leads to the necessity that the system is able to inform about its states and future behaviors (Freed et al., 2005). For the intelligent agent to understand and correctly interpret the task commands and for the human operator to understand the agent's feedback, we base our software development approach on *cognitive automation*, which is described further in (Onken and Schulte, 2009) and the following sections.

Designing technical systems which are ergonomic and understandable for the human operator is a challenging task. A natural approach to designing such human-machine systems is to share work with the operator on a communication and cooperation level that is similar to that between two human operators. This is especially useful, if the technical system has the purpose of supporting the human operator as in the assistant systems mentioned in the previous section.

Brüggenwirth S., Strenzke R., Alexander A. and Schulte A. (2010).
A GENERIC COGNITIVE SYSTEM ARCHITECTURE APPLIED TO THE UAV FLIGHT GUIDANCE DOMAIN.
In *Proceedings of the 2nd International Conference on Agents and Artificial Intelligence - Agents*, pages 292-298
DOI: 10.5220/0002718202920298

As explained in (Schulte et al., 2008), these systems must support the human operator in maintaining or even improving the operator's situation awareness (SA). A well accepted theory of situation of SA in the human factors community stems from Endsley (Endsley and Garland, 2000).

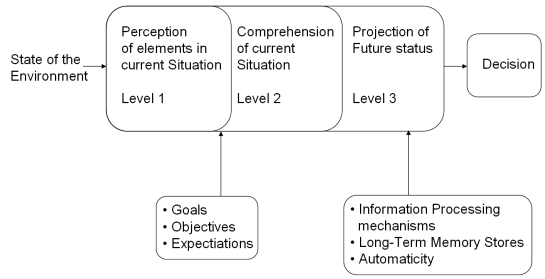

Figure 1: Situation awareness according to Endsley (Endsley and Garland, 2000).

As shown in figure 1, she distinguishes three levels of SA with increasing complexity. While level 1 SA simply represents the elementary building-blocks of a perceived situation, level 2 SA strives to unify these elements in order to create a most comprehensible situational understanding. Interestingly, level 3 SA includes both, expected outcomes of own actions and the anticipated evolution of environment: Knowledge, which will also play a key role in the planning process. We argue that this human-oriented definition of situation awareness should be closely linked to what is commonly known as the belief state of an agent. Accordingly, an agent that builds up a human-like high-level SA, e.g. inspired by Endsley's categories, has advantages when when interacting with a human operator. Likewise, information that is used internally to derive this high-level abstraction should not be exposed to the operator.

Furthermore the artificial agents must be able to pursue goals which are either matching directly to the cognitive goals of the human operator or are subgoals that the operator has assigned to them. We summarize these concepts as *cognitive automation* (Onken and Schulte, 2009). Cognitive automation enables technical systems to have a human-oriented situation representation, situation awareness and an explicit representation of goals. We belief that, although in some cases computationally less efficient, cognitive automation simplifies human-machine interaction as well as the knowledge engineering and system development processes.

Although our first, Soar-based (Laird et al., 1987) implementation of the multi-UAV system has a relatively simple means-ends plan-library that is statically defined at development time, the knowledge based approach already shows emergent behavior not predefined by the developer. This behavior is not generated by directed reasoning or global optimization, instead it can be seen as a compromise of multiple actions related to multiple goals that arise sequentially.

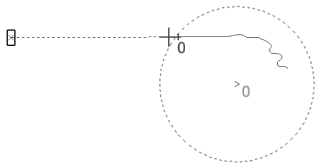

Figure 2: Emergent behavior due the explicit goals: avoid threat and return to base (Meitinger and Schulte, 2009).

An example is the generation of a flight trajectory shown in figure 2. In the depicted situation, two goals are relevant - one is the shortest possible flight path to the home base and the other is not to come too close to any SAM (Surface to Air Missile) site. The latter goal is (reactively) triggered as soon as a minimal distance is under-run and then replaces the first goal as the actively pursued one (Meitinger and Schulte, 2009). Even though the resulting flight path resembles the optimal curved trajectory, the emergent behavior described above is the result of a greedy algorithm, which follows a locally optimal heuristic (alternating between the shortest flight path and SAM site avoidance goal). Currently, no projection of future environment states is made and therefore loops in the behavior of the agent may occur.

2 THE COGNITIVE PROCESS

According to the principle of cognitive automation described in section 1, we have derived an information processing scheme, which we call the *cognitive process*. It is structured in close analogy to Rasmussen's model of human information processing (Rasmussen, 1983), often cited among cognitive psychologists and within the human factors engineering community.

In figure 3, we present an interpreted version of the traditional Rasmussen model, tailored for application in intelligent agent environments. Our model is comprised of a low-level, subsymbolic feedback control layer (*skill based*), a symbolic, reactive layer with prestored procedures (*procedure-based*) and a deliberative, goal-oriented layer (*concept-based*) comparable to a BDI-agent design. From an agent-theoretical point of view, the approach resembles a hybrid agent architecture with horizontal layering, comparable to Fergusons *TouringMachines* (Ferguson, 1992). Per-

ceived data (*feature formation*, often called the *see*-function) is available to the procedure based and concept based layer, but the control policy is currently straightforward: An exact match to a reactive rule always takes priority over the deliberative layer. We currently do not partition our model in hierarchical layers, e.g. competency-based or multi-agent-wise. Communication mechanisms between agents have to be explicitly encoded.

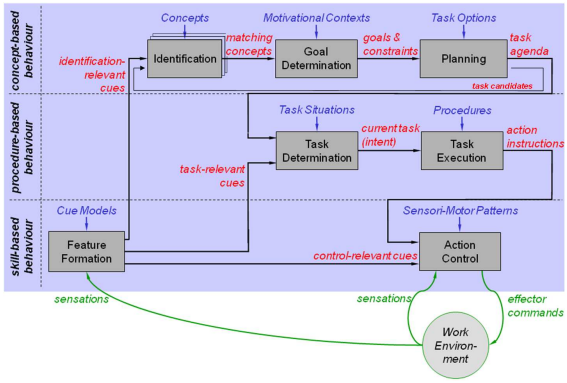

Figure 3: Interpretation of Rasmussen's model of human performance incorporating an information technology approach (Onken and Schulte, 2009).

The cognitive process separates static, a-priori knowledge in the form of production rules (i.e. the design-time knowledge: *concepts*, *motivational contexts* or *procedures*) from instantiated and parameterized situational knowledge (i.e. the run-time data: *matching concepts*, *goals & constraints* or *action instructions*). In practice, the cognitive subfunctions (e.g. *identification*, *planning*, etc.) may use all the knowledge that is available to the system. The conceptional processing of information, however, is sequential, that is, the output of one subfunction serves as the input to the subsequent one.

2.1 Identification

The first processing step for concept-based behavior matches identification relevant cues with known concepts. This inference step is done using Soar style production system syntax with extensions for an object-oriented representation of the scene, such as class encapsulation, inheritance and type-safety as shown in figure 5. After the forward chaining process has come to quiescence, all identification relevant cues have been translated into matching high-level concepts.

2.2 Goal Determination

Matching concepts represent the high-level abstraction that is meant to resemble human-like situation awareness. It also serves as input data to the goal determination module and may hence trigger the activation of certain goals or a subgoal hierarchy as shown in figure 4. A goal is met if the associated instance of a matched concept satisfies certain constraints, e.g. usually an environment attribute lies within a certain numerical interval. The agent typically strives to fulfill all currently active goals - In case it detects a failed goal, it will activate its planning module to establish the unmet conditions of the goal.

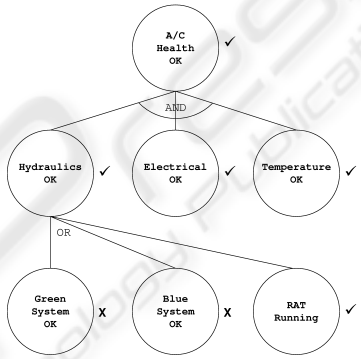

Figure 4: And-Or tree representing a goal-hierarchy example used for aircraft health monitoring

2.3 Planning

The planning module has the purpose of generating a sequence of tasks to bring the system from the current situation into a desired state - in our case to fulfill all its currently active goals. Similar to Strips (Fikes and Nilsson, 1971) planners we associate preconditions and expected effects (postconditions) with each possible action. To enable backward chaining, we also require that every production involved in the planning process is function free. We furthermore adhere to a *framing axiom*, that is, facts which are not explicitly mentioned as postconditions remain unchanged. Unlike a classical planner however, we do not add or remove literals which are checked for a goal state directly, but project the expected *identification relevant cue* of every feasible *task candidate* (as shown in figure 3). This introduces additional inference steps to identify matching concepts before the update and testing of the goal-hierarchy is performed, but has the advantage of reusing the knowledge for situation comprehension (concepts) and motivational contexts. The planning process announces failure, if it does not encounter a state that satisfies all active goals within a

```
class <belief> danger // class 'danger' is added to namespace 'belief'
{
  attributes:
   link     aircraft; // link to an object of class 'aircraft'
   string   threat;
  behavior:
   sp{create // Soar-like production
      (instance[belief::aircraft::*] <own> ^ownAC true) // IF WM contains an 'aircraft' object that has attribute 'ownAC' set to 'true'
      (instance[belief::aircraft::*] <other> ^otherAC true) // and an aircraft object that has attribute 'otherAC' with value 'true'
      (instance[belief::distance::*] <dist> ^between <ownAC> // and an object of class 'distance' the two aircraft as 'between' parameters
                                            ^between <otherAC> ^distance small) // and its 'distance' attribute with value 'small'
      -->
      (elaborate <i> ^aircraft <otherAC> ^threat high) // THEN create object of this class ('danger') ref. to other A/C, set 'threat':='high'
     }
}
```

Figure 5: Code-Example used in the cognitive process.

certain timespan. Currently, we use a PDDL 2.2 planner to encode inferred facts as actions and derived predicates. The re-implemented system may offer the option to perform planning by direct manipulation of the working memory graph

2.4 Monitoring

The outcome of a successful planning process is a *task agenda*. The items on the agenda are tuples, composed of tasks and associated *task-relevant cues* which trigger the execution of the task. A task-relevant cue might trivially be a simple time tag or any other indicator which initiates the task, depending on the ontology. For execution monitoring purposes, the agenda also contains the anticipated world-state generated from the pre- and postconditions of a task during the planning phase. This usually happens whenever the next task is toggled or at intermediate discretization points defined at design time. Should the system detect a deviation from the task agenda, it triggers a re-planning processes from the current world state. Alternatively it may ask the operator for instructions to alter the system knowledge, e.g. if a hardware failure renders a task option infeasible.

2.5 Procedure-based Behavior

We follow the general convention that an exact match of a procedure-based behavior rule immediately fires its associated reactive task, while suspending concept-based behavior during execution. Once the procedure-based task has completed, the monitoring function will recognize a deviation from the agenda and initiate re-planning from the current world state.

Experiments and surveys among pilots and UAV-operator personnel showed that it is this procedural knowledge, extracted directly out of manuals, regulations, combat operation procedures or gained from experience, that frees pilots from spending cognitive resources on routine tasks and enables them to direct their attention on high-level mission goals. Procedure-based behavior therefore poses an important feature for knowledge extraction and engineering in the UAV guidance domain.

Exact matching of procedure-based rules however implies a closed and consistent world, and is hence prone to reproducing typical problems encountered in conventional automation, such as literalism or brittleness (Billings, 1997). We therefore envision as future work a concept-based, high-level supervision mechanism, which concurrently projects the expected outcome of the procedure-based behavior and may explicitly interrupt it, in case it interferes with certain high-level goals. Ideally, procedure-based behavior can increase performance, since only the effects of the associated task need to be projected against the goals, instead of going through the complete planning phase.

3 IMPLEMENTATIONS OF THE COGNITIVE PROCESS

Both our implementations of the cognitive process use a production system that holds in its working memory the current situation representation including the system's beliefs, desires and intents and contains a knowledge base consisting of two types of production rules that are applied to the working memory to infer new facts or make decisions based on existing facts. The two rule types are (i) *inference rules* that cause

a reversible modification which is retracted, once the rule's left hand side does not match anymore and (ii) *operator rules* that cause persistent modifications of the state. The application of the production rules is directed by a simple automaton as depicted in figure 6 that operates on the working memory.

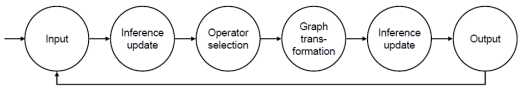

Figure 6: Decision cycle controlling the Cognitive Process.

After reading the input the automaton updates the inference rules by firing the newly applicable ones and retracting the invalidated ones. In this phase all firings occur in parallel until no inference rule is applicable. This state is referred to as *quiescence*. Inference rules can propose the creation operator nodes that enable graph transformation rules by generating operator proposal nodes in working memory. During *operator selection* the automaton selects one of the operator proposal nodes and creates an operator node from it. This operator node serves as a precondition for the operator rules that are then applied. In the next step the inference rules are again updated to incorporate the modifications of the operator rules and the operator node created during operator selection is deleted. Finally the automaton generates the output and starts over again.

This way the automaton provides the flow control required by the cognitive process that (i) separates the parallel execution of the inference rules from the sequential execution of the operator rules and (ii) allows priority based decisions on the order of the application sequence of the latter (Laird et al., 1987).

The described model of the cognitive process was initially implemented in our framework COSA. It used the well established production system Soar as knowledge processor but extends the functionality and the Soar syntax to enable the object oriented elements of our programming language. The framework has been extensively used in various applications in the field of multiple co-operating semi-autonomous aerial vehicles (Schulte et al., 2008; Uhrmann et al., 2009). Respective prototypes have been demonstrated in a collaborative simulation environment with great success, but with respect to forthcoming field experiments including the use on embedded platforms and increasing complexity of the encoded knowledge the observed run-time performance of the existing system is not sufficient.

Our experimental results showed that the pattern matching phase within the graph transformation is the bottleneck in the Soar-based implementation of the cognitive process – more than 90% of the run-time is spent in this phase (Matzner et al., 2008). Therefore an efficient graph pattern matching for the left hand side of the rule is the key to performance of such an implementation. To achieve this, we re-implemented the system by transforming the situation representation from untyped facts in the Soar-based implementation into directed, typed multigraphs. We also developed an incremental pattern matching algorithm that on the one hand incorporates techniques from the field of graph transformations to increase performance and on the other hand extends the expressiveness our modeling language by directly supporting inheritance. As the performance of a pattern matching algorithm very much depends on the quality of the search plan which again depends on the situation specific cardinality of objects in the host graph we further augmented our algorithm with an evolutionary search plan optimizer.

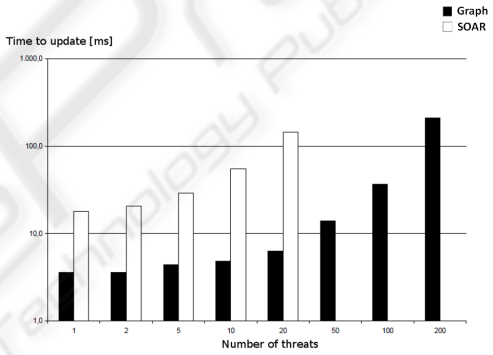

Figure 7: Re-implemented system shows significant increase in performance.

Examination of the re-implemented system in a benchmark multi-UAV application in which an increasing amount of rules were imposed on the system, showed a significant decrease of overall memory consumption which enabled the system to handle scenarios with over 200 instantiated SAM site (threat) objects, whereas the existing system reached the memory limits at 50 objects. Also, the time used to complete the pattern matching in each cycle was decreased by around an order of magnitude (see figure 7).

4 APPLICATION FOR MANNED-UNMANNED TEAMING

The manned-unmanned teaming scenario regards a manned transport helicopter with a crew consisting

of a pilot flying and a commander, who is responsible for the helicopter and may delegate tasks to a self-organizing pool of UAVs. Figure 8 shows the helicopter cockpit plus multi-UAV simulator in use. The transport helicopter has the mission goal to deliver troops to a specified object which is reachable via a predefined route. The way home is also predefined. Both routes include a corridor which may only be used during specific time intervals. For the protection of the transport helicopter there is an additional attack helicopter, that flies in front of the transport helicopter and may also leave the route in order to provide protection. This is necessary because there are previously unknown vehicles distributed in the mission area, which are either posing a threat or not. To enable the helicopter commanders to assess the situation, these vehicles have to be photographed by the UAVs early enough before any manned helicopter passes the corresponding area. So both helicopter crews are in need of the capabilities of the UAV pool. The intelligent agent software on-board the UAVs has to cope with payload management (i.e. set camera angles, take picture) and flight management (i.e. trajectory planning) at the same time to create the photos the commanders need in order to identify the vehicles. Furthermore, the tasks given to the UAV pool have to be sorted temporally and distributed among its participants, which leads to the necessity of a mission management capability. This capability also enables the UAVs to receive, distribute and execute tasks that originate from various sources in addition to the two manned helicopters mentioned above. From the view point of the human operators the UAVs represent capabilities (like reconnaissance) which is the reason why we speak of capability management instead of vehicle management or the like.

Figure 8: Helicopter plus multi-UAV simulator with the commander viewing UAV sensor data (Uhrmann et al., 2009).

4.1 Planning and Resource Problems

As our work is evolving towards a non-deterministic, real-world application (which will culminate in the deployment of our physical UAVs, see figure 9), we have to deal with much more complex, open environments. Another recent trend is the focus on the manned-unmanned teaming application, leading to a more seamless integration of manned aircraft and/or human operators into the missions. In military missions which include manned components, predefined time schedules have to be kept to reduce risk of life. This can only be ensured, if the artificial agents themselves have a concept of time and are able to generate plans that consider certain temporal constraints. In addition, the human operator in such a dangerous environment expects the artificial agents to act on a high optimization level.

Figure 9: Unmanned helicopter (left) / plane (right) as MUM-T testbed.

In the scenario described above, time is the most critical resource, but one may easily think of other bounded resources that could play a role if realism is raised, e.g. fuel or payload.

In the example presented in figure 10 two helicopters are on their way to the operation area. There are two UAVs to support them flying in front, tasked to reconnoiter the route, then move towards the operation area and eventually build a communications relay (necessary as soon as the transport helicopter has landed). Only one UAV is needed to work as relay, but the reconnaissance task can be covered by multiple UAVs to either achieve higher quality results or faster accomplishment. Therefore a suitable plan for the UAVs is to cooperate concerning the reconnaissance tasks until one UAV takes over the role as relay, while the other one is free to work on tasks that are out of the scope of this example.

We now consider the unforeseen event that a vehicle is discovered along the route and the commander of the attack helicopter generates additional tasks for the UAV pool: identify the vehicle, then mark it (see figure 10). This leads to a re-planning process, that should answer the questions, which UAV can take over the new tasks and will there still be any UAV that

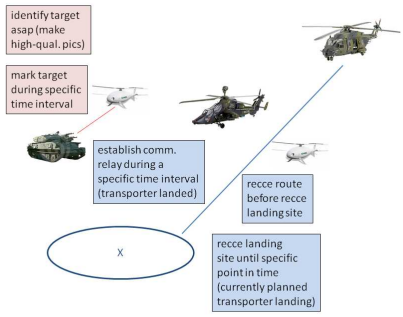

Figure 10: Manned-unmanned teaming use case.

is able to work as communications relay in the given time frame. In case the temporal constraints cannot be satisfied, it may be an option to re-plan the whole mission, e.g. slow down the manned helicopters so that the constraints become loosened.

4.2 Future Work

To solve such a problem without too much human interaction, the intelligent agent needs to predict the dynamic development of the situation. Furthermore, the agent needs to keep track of time constraints, consumable and renewable resources. As already mentioned, this will require more elaborate planning algorithms, potentially operating directly on the working memory graph. Interesting improvements could come from features of recent AI planners, e.g. strong and soft constraints and preferences, that were introduced with PDDL3 (Gerevini and Long, 2005).

As furthermore mentioned in section 2.5, another current research activity focuses on the control policy between reactive and deliberative layer. Finding this right balance of flexibility is a major question in the agent community - other hybrid agent architectures have already introduced sophisticated (but sometimes for our purposes too complex) concepts.

REFERENCES

Billings, C. E. (1997). *Aviation Automation - the Search for a Human-Centered Approach*. Lawrence Erlbaum Associates, Mahwah, NJ.

Endsley, M. R. and Garland, D. J. (2000). *Situation Awareness Analysis and Measurement*. Lawrence Erlbaum Associates, Mahwah, NJ.

Ferguson, I. A. (1992). *TouringMachines: An Architecture for Dynamic, Rational, Mobile Agents*. PhD thesis, University of Cambridge. Clare Hall.

Fikes, R. and Nilsson, N. (1971). Strips: A new approach to the application of theorem proving to problem solving. In *Artificial Intelligence*, volume 2, pages 198–208, Heidelberg, Germany. Springer.

Freed, M., Bonasso, P., Ingham, M., Kortenkamp, D., Pell, B., and Penix, J. (2005). Trusted autonomy for spaceflight systems. In *AIAA First Space Exploration Conference*, Orlando, FL.

Gerevini, A. and Long, D. (2005). Plan constraints and preferences in PDDL3. Technical report, Dipartimento di Elettronica per l'Automazione, Universita degli Studi di Brescia.

Laird, J. E., Newell, A., and Rosenbloom, P. S. (1987). SOAR: An architecture for general intelligence. In *Artificial Intelligence*, volume 33, pages 1–64, Heidelberg, Germany. Springer.

Matzner, A., Minas, M., and Schulte, A. (2008). Efficient graph matching with application to cognitive automation. In *Applications of Graph Transformations with Industrial Relevance*, pages 297–312, Berlin, Germany. Springer.

Meitinger, C. and Schulte, A. (2009). Human-uav cooperation based on artificial cognition. In *Engineering Psychology and Cognitive Ergonomics*, pages 91–100, Heidelberg, Germany. Springer.

Onken, R. and Schulte, A. (2009). *System-ergonomic Design of Cognitive Automation in Work Systems*. Springer, Heidelberg, Germany.

Rasmussen, J. (1983). Skills, rules and knowledge, signals, signs and symbols, and other distinctions in human performance models. In *IEEE Transactions on Systems, Man, and Cybernetics*, volume SMC-13, pages 257–266, Heidelberg, Germany. Springer.

Rauschert, A., Meitinger, C., and Schulte, A. (2008). Experimentally discovered operator assistance needs in the guidance of cognitive and cooperative uavs. In *Proceedings of HUMOUS conference*, Brest, France. Springer.

Schulte, A., Meitinger, C., and Onken, R. (2008). Human factors in the guidance of uninhabited vehicles: Oxymoron or tautology? the potential of cognitive and co-operative automation. In *International Journal on Cognition Technology & Work*, Heidelberg, Germany. Springer.

Uhrmann, J., Strenzke, R., Rauschert, A., C.Meitinger, and Schulte, A. (2009). Manned-unmanned-teaming: Artificial cognition applied to multiple uav guidance. In *NATO RTO SCI Symposium on Intelligent Uninhabited Vehicle Guidance Systems*, Neubiberg, Germany.