

An Ambient Agent System Assisting Humans in Complex Tasks by Analysis of a Human's State and Performance*

Tibor Bosse¹, Fiemke Both¹, Rob Duell², Mark Hoogendoorn¹, Michel Klein¹,
Rianne van Lambalgen¹, Andy van der Mee², Rogier Oorburg²,
Alexei Sharpanskykh¹, Jan Treur¹, and Michael de Vos²

¹Vrije Universiteit Amsterdam, Department of Artificial Intelligence
De Boelelaan 1081, 1081 HV Amsterdam, The Netherlands
{tbosse, fboth, mhooogen, mcaklein, rm.van.lambalgen, sharp, treur}@few.vu.nl
²Force Vision Lab, Barbara Strozziilaan 362a, 1083 HN Amsterdam, The Netherlands
{rob, andy, rogier, michael}@forcevision.nl

Abstract. Human task performance varies depending on the task, environment, and states of the human over time. To ensure high effectiveness and efficiency in the execution of complex tasks, adaptive automated assistance of the human may be required. In this paper, a generic design for a multi-agent system architecture is presented and a personal assistant agent is presented that makes use of the proposed architecture. The agent constantly monitors the task execution and well-being of the human via non-intrusive sensors, and intervenes when a problem is detected. A human is given a complex task, while the future performance is predicted using observations and a dynamical model for the human's work pressure and exhaustion. If the predicted exhaustion becomes too high, the ambient agent can assist the human in a number of ways. Experiments show that the support system increases performance with around 13%, and that it enhances the feeling of control of the situation.

Keywords. Ambient Intelligence, Personal Assistant Agent, Human Functional State

1. Introduction

Human performance can degrade over time when demanding tasks are being performed [1]. However, high effectiveness and efficiency levels are of particular importance for critical tasks. In such cases, automated assistance to support humans in task execution is required. A challenging example of a critical task in the military domain is naval combat management. Combat management tasks are complex, time-constrained, take place in a dynamic environment, and require skill and expertise. The combination of these characteristics can easily lead to demanding situations in which the human operator is likely to endure mental and physical aggravation, resulting in degradation of task performance. In general, effectiveness and efficiency of task execution depend on the capabilities, experience, and state of the person performing the task. Different persons as well as one person at different time points may require different degrees and types of assistance. To achieve this, an intelligent personal assistant is needed that monitors the performance of the user, takes his or her personal characteristics into account and analyses the person's state at any given point in time.

* Parts of this paper are based on work presented at the 22nd International Conference on Industrial, Engineering & Other Applications of Applied Intelligent Systems (IEA/AIE 2009), the Fourth International Conference on Augmented Cognition and 13th International Conference on Human-Computer Interaction (HCI'09), and the Second IEEE International Conference on Intelligent Human Computer Interaction (IHCI'10).

A variety of intelligent personal assistants have been proposed to support humans during the execution of tasks (see e.g. [2], [3]). Such personal assistants usually include models that represent the state of the human and his or her tasks at particular time points, which can be utilized to determine when intervention is needed. An example of such a model addresses the cognitive load of the human (see e.g. [4]). Models differ in the considered aspects of human behaviour and of the execution of tasks. Thus, depending on the model type, a specific personal assistant may react only to particular events and states of the environment, in which a human performs his or her tasks. Hence, to provide a more proper form of assistance the availability of a larger set of models from which an appropriate instance can be chosen depending on circumstances may be essential. This would also address better the variation in measurable states of the world and the human, the dynamics of a human's states over time and the intervention possibilities which all may vary per domain. The existing models proposed for personal assistants focus on a certain domain and hence are not generic.

This paper presents a generic design for a multi-agent system architecture including personal assistant agents. The personal assistant used in this paper allows for self-configuration by loading domain specific models and thus alters its own functionality. These domain specific models also address the dynamics of states over time. The personal assistant agent can use these models to monitor and analyse the current state of the human. When a problem is detected, possible hypotheses are tested through specific sensors which measure the human's psychophysiological state (e.g. heart rate) and the state of the environment (e.g. noise). The best intervention method is selected (if needed) in the specific domain and task. In addition, an evaluation of the system by a user study is reported. The presented ambient agent system functions as a personal assistant for a human that is given a complex task in the context of a (simplified) simulation-based training environment related to the military domain. Given predictions on the person's state based on observations and a model for work pressure described in [5], the person can be supported (i.e. by reallocating part of the task).

The remainder of the paper is organized as follows. The multi-agent system architecture is described in Section 2. Model maintenance and state maintenance agents are discussed in Section 3. Further, the functions of a personal assistant agent are considered: self-maintenance in Section 4 and monitoring and guidance in Section 5. A prototype implementation is described in Section 6. Section 7 outlines the user study that is done for evaluation of the system, while in section 8 results of the evaluation are analysed. The paper concludes with a discussion.

2. The Agent-Based System Architecture

The overall process first has been modelled at the conceptual level. The developed conceptual component-based architecture presented in Figure 1 comprises a number of essential components:

- *Process*: ensures request/provision of data from/to different components (on behalf of other components). On request of the Reflection component, it selects/activates/executes an analysis method(s).
- *Reflection*: exercises control/monitoring over the functioning of the whole system. In particular, by performing meta-analysis/meta-reasoning using some data (e.g., human, task and system characteristics, inputs from the environment) this component selects/activates an analysis method(s) at some time point(s).
- *Library of specifications*: contains specifications of analysis methods, workflow, cognitive and dialogue models. Meta-information about the components of the library includes relations between requirements for components and descriptions of components.

- *Storage of execution information*: this is used for storage and retrieval of information about the human, the world, the execution of workflows, dialogues and systems. This component contains also meta-information on the required components from the library of specifications.

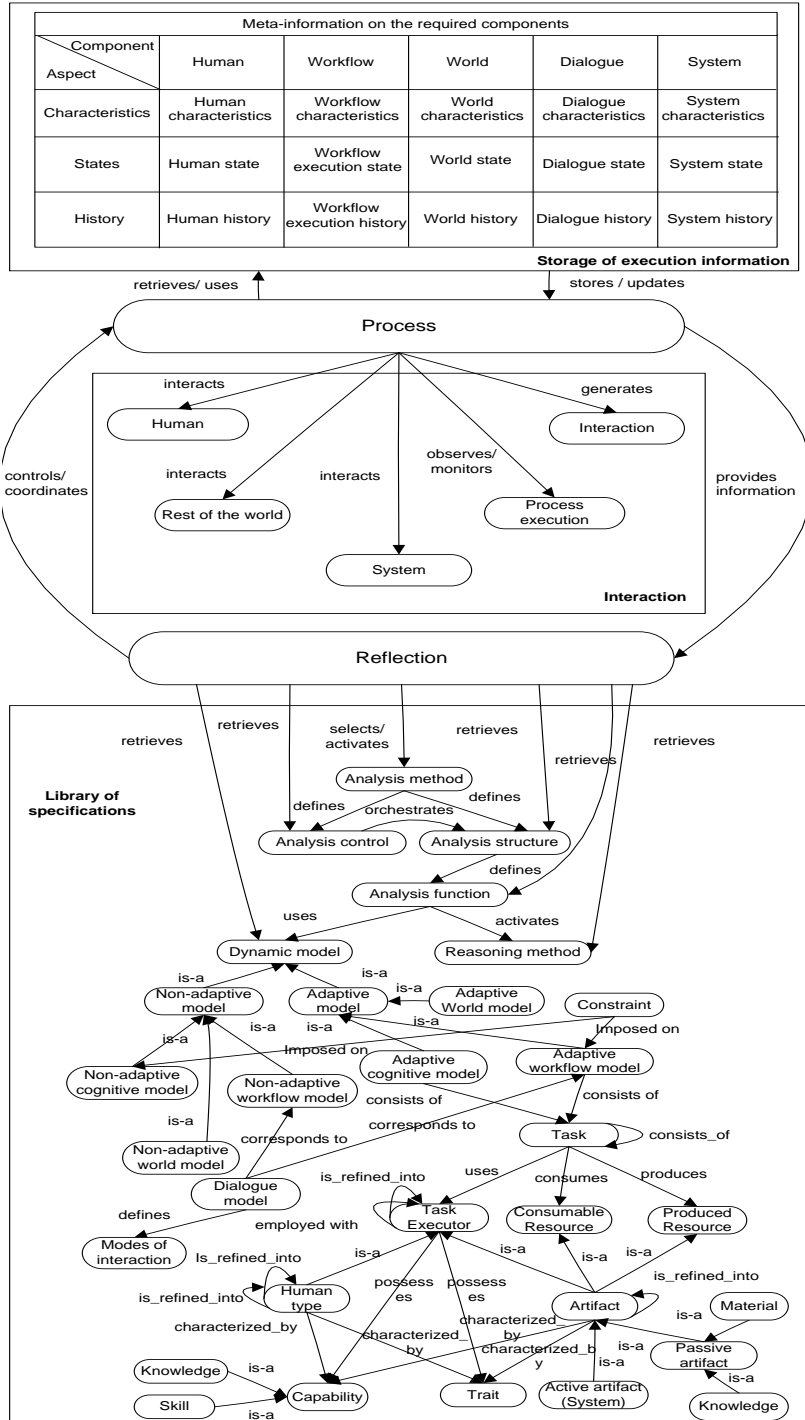


Figure 1. Overall conceptual component-based architecture

Given the essential components that have been identified above, the system has been modelled by a multi-agent system architecture consisting of the following types of agents:

- *Self-maintaining personal assistant agent* (SMPA): supports a human operator during the execution of a task;
- *Model maintenance agent* (MMA): contains a library of models used for the configuration of SMPA's.
- *State maintenance agent* (SMA): maintains characteristics, states and histories of other agents, of the world and of the executions of tasks.
- *Mental operator agent* (MOA): represents the mental part of the human operator.
- *Task execution support agent* (TESA): used by the human operator as an (active) tool during the execution of a task.

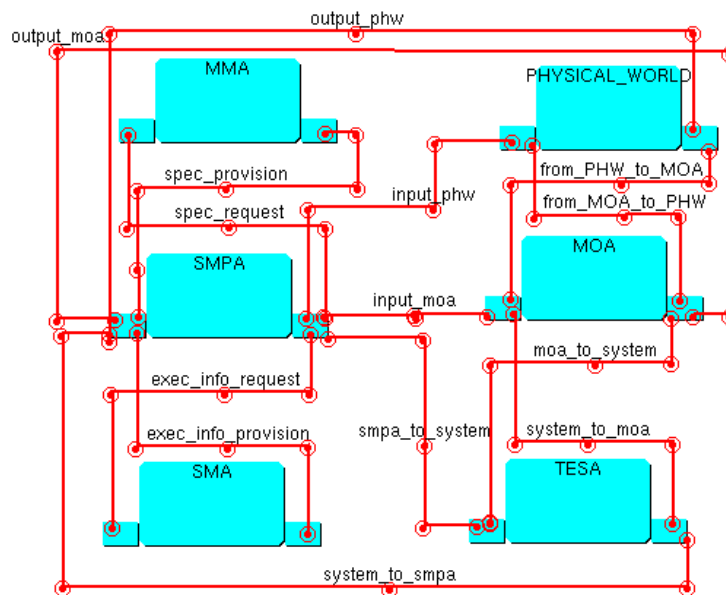


Figure 2. An overview of the multi-agent system architecture

These agents are depicted graphically in Figure 2. The agents are represented as squares, where a small box on the left of the square represents the input of the agent, and the small square on the left the output. Another component of the multi-agent system architecture is the physical world that comprises all material (or physical) objects including the body of the human operator. The personal assistants have two modes of functioning:

- *the self-maintenance mode*, in which they are able to reason about the model specifications required to perform their tasks and to achieve their goals, to request these models and to load them (altering their functionality);
- *the monitoring and guidance mode*, in which they perform monitoring and guidance of the human to whom they are related.

In the self-maintenance mode communication takes place between personal assistant agents and model maintenance agents. In the monitoring and guidance mode personal assistant agents communicate with state maintenance agents, the mental operator agent, and task execution support agents. Furthermore, they interact with the physical world by performing observations

(e.g., of the operator's state). The mental part of a human operator represented by a mental operator agent is connected to the operator's physical body, which can act in the physical world. To specify interaction between agents, and between agents and the physical world, the interaction ontology described in Table 1 is used.

Models for the different agents were designed based on the component-based Generic Agent Model (GAM) presented in [6]. Within the Generic Agent Model the component *World Interaction Management* takes care of interaction with the world, the component *Agent Interaction Management* takes care of communication with other agents. Moreover, the component *Maintenance of World Information* maintains information about the world, and the component *Maintenance of Agent Information* maintains information about other agents. In the component *Agent Specific Task*, domain-specific tasks can be modelled. The component *Own Process Control* initiates and coordinates the internal agent processes. Adopting GAM, the agent models have been obtained as a refinement, by incorporating components relating to the tasks in this application in the following manner.

In a self-maintaining personal assistant agent all components are used. In the component *Maintenance of Agent Information* the dynamic models of human functioning are maintained, as well as the states of the human, and the history thereof. Similarly, the component *Maintenance of World Information* stores a dynamic world model, a world state model, and a world history model, respectively. The component *Own Process Control* takes care of the agent's self-maintaining functionality; this is addressed in Section 4. The component *Agent Specific Task* has a number of subcomponents, devoted to the agent's monitoring and guidance task; these are addressed in Section 5. Finally, as in the Generic Agent Model, the components *World Interaction Management* and *Agent Interaction Management* prepare (based on internally generated information) and receive (and internally forward) interaction with the world and other agents. The other agents within the system are specified in a similar fashion; see the complete specification in [7] for more details.

3. Maintenance Agents

Two types of maintenance agents are included in the multi-agent system architecture: model maintenance agents (MMA) and state maintenance agents (SMA).

The model maintenance agent contains a library of models that can be used by self-maintaining personal assistant agents to perform their tasks. Models of four types are maintained in the library: monitoring and guidance task models, cognitive models, workflow models, and dialogue models. Models are provided by the model maintenance agent to self-maintaining personal assistant agents upon request. To facilitate the model acquisition process, each maintained model is annotated by particular parameters. The ontology used for the annotation is assumed to be known to the agent-requester. In the general case, such an ontology may be also provided by the model maintenance agent to an self-maintaining personal assistant agent upon request. In Table 2 and 3 some of the parameters and their possible values used to annotate the different types of models are listed. The models maintained in model maintenance agents may be specified using different knowledge representation languages. However, it is important to ensure that a model provided to a self-maintaining personal assistant agent can also be interpreted by this agent. The state maintenance agent maintains information about the characteristics, states and histories of the agent types *mental operator agent* and *task execution support agent*, of the physical world, of the workflows and of dialogues related to them. Information about states and histories (i.e., sequences of states) is stored in a time-indexed format using the predicate $at(prop, time)$, where a state property is specified by the first argument and the time point at which this property holds is specified by the second argument.

Such information is gathered and provided to the state maintenance agent by self-maintaining personal assistant agents, which may also use this information in their analysis. Information for which a self-maintaining personal assistant agent has no immediate need after being stored in the state maintenance agent can be removed from the assistant agent's Maintenance of Agent Information and Maintenance of World Information components. When stored information is required by a self-maintaining personal assistant agent, it can be requested from the state maintenance agent. An information request includes the identification of the element (i.e., mental operator agent, task execution support agent, physical world, a workflow, a dialogue), the aspect (i.e., characteristic, state, history) and the time interval for which information should be provided.

4. Self-Maintenance by SMPA

For each human operator that needs to be supported during the task execution a self-maintaining personal assistant agent is created. Initially, the personal assistant agent contains generic components only. The configuration of the personal assistant agent is performed based on an organisational role that needs to be supported by the agent, on the characteristics of a human operator who is assigned to this role, and on the goals defined for the personal assistant agent.

The human operator is assigned a role of being responsible for a package of tasks, which is provided to the personal assistant agent. For the whole task package, as well as for each task separately a set of goals and norms related to the execution of the task(s) may be defined. To determine the characteristics of the operator responsible for the execution of these tasks, the personal assistant agent sends a request to the state maintenance agent. If the operator is known to the state maintenance agent, his/her known professional, cognitive, psychological and physical characteristics are provided to the personal assistant agent. Otherwise, the state maintenance agent returns to the personal assistant agent the default profile (i.e., a standard set of characteristics).

For the personal assistant agent a set of prioritized general goals is defined, which it strives to achieve. Some of these goals are related to the quality of the task execution, others concern the operator's well-being (see Table 4). Goals of two types are distinguished: (1) achievement goals (e.g., goals 1-3 in Table 4) that express that some state is required to be achieved at (or until) some time point, specified by `has_goal(agent, achieve(state, time))`; (2) maintenance goals (e.g., goals 4-7 in Table 4) that express that some state is required to be maintained during a time interval specified by `has_goal(agent, maintain(state, begin_time, end_time))`. Often the state over which a goal is expressed is defined as an expression over some performance indicator(s) (e.g., number of products, execution time, level of attention). A role description may contain role-specific goals that are added to general goals.

Although refinement may be defined for some general goals of the personal assistant agent, most of them remain rather abstract. Using the information about the human operator and the assigned tasks, some goals of the personal assistant agent may be refined and instantiated into more specific, operational goals. This is done by the Own Process Control component of the personal assistant agent. For example, one of the subgoals of goal 7 expresses '*It is required to maintain the operator's heart rate within the acceptable range*'. Based on the available information about the physical characteristics of the operator (e.g., the acceptable heart rate range is 80-100 beats per minute), this goal may be instantiated as '*It is required to maintain the operator's heart rate 80-100 beats per minute*'. Also the task-related generic goals can be refined into more specific goals related to the particular tasks from the provided package (e.g.,

'It is required to achieve the timely execution of the task repair sensor TX324'). New goals resulting from refinement and instantiation are provided by the Own Process Control component to the Agent Specific Task component of the personal assistant agent, which is responsible for checking if the generated goals are satisfied.

The configuration of the self-maintaining personal assistant agent begins with the identification of the suitable monitoring and guidance task model(s) that need(s) to be requested from the model maintenance agent. To this end, the model parameters are identified by the Own Process Control component based on the goals of the personal assistant agent. For example, to establish if the operator complies with a workflow model, diagnosis of the operator's state may need to be performed. Thus, the parameter type of analysis with value diagnosis is included in the query to the model maintenance agent. A query is specified using the function `model_query(query_id, param, list_of_values)`, where the first argument indicates a query identifier, the second argument indicates a parameter and the third argument indicates a list of parameter values. The elements of a list are specified using the function `is_in_list(element, list)`.

The choice of cognitive models is guided by the goals that concern internal states of the operator. From the goals in Table 4 and their refinements and instantiations, a number of internal states can be identified, among which experienced pressure and heart rate. To extract the suitable cognitive model, the query to the model maintenance agent should be defined as:

```
model_query("q1", states, l1) & is_in_list("experienced pressure", l1) & is_in_list("heart rate", l1) &
model_query("q1", type, l2) & is_in_list("cognitive", l2)
```

For each task from the task package the appropriate workflow and dialogue models are extracted from the model maintenance agent. In general, more than one workflow and dialogue model may be maintained in the model maintenance agent for a task. For example, different models of the task execution may exist for operators with different capabilities and traits (e.g., novices and experts). In this case the query to the model maintenance agent should be based both on the task descriptions, as well as on the characteristics of the operator, e.g.:

```
model_query("q2", task, l1) & is_in_list("maintenance of a sensor", l1) & model_query("q2", type, l2) &
is_in_list("workflow", l2) & model_query("q2", task executor capabilities, l3) &
is_in_list("little experience with the task", l3)
```

By matching the queries received from the personal assistant agent with the annotations of the maintained models, the model maintenance agent identifies the most suitable model(s), which is (are) sent to the requestor. The provided models are stored in the Maintenance of Agent Information component of the personal assistant agent.

5. Monitoring and Guidance by SMPA

Within the *Agent-Specific Task* of SMPA, the process of monitoring and guidance takes place. At the highest abstraction level the component consists of 5 subcomponents: *Coordination*, *Monitoring*, *Analysis*, *Plan Determination*, and *Plan Execution Preparation*. The interaction between these components is depicted in Figure 3. In the following sections, first a general explanation is given for each component followed by a more specific explanation directed at the user study which will be presented in Section 7.

The initial inputs for the process are the goals provided from SMPA's *Own Process Control* component, which are refined within the *Coordination* component into more specific criteria that should hold for the operator's functioning (e.g., 80% of certain objects on a radar screen should be identified correctly). Note that goal refinement may also occur after the initialization phase based on the results of particular observations. For example, based on the acceptance observation of a task by the operator, the criteria for particular task execution states may be

generated from task-related goals. The criteria are fed to the *Monitoring* component, which is discussed in more detail below.

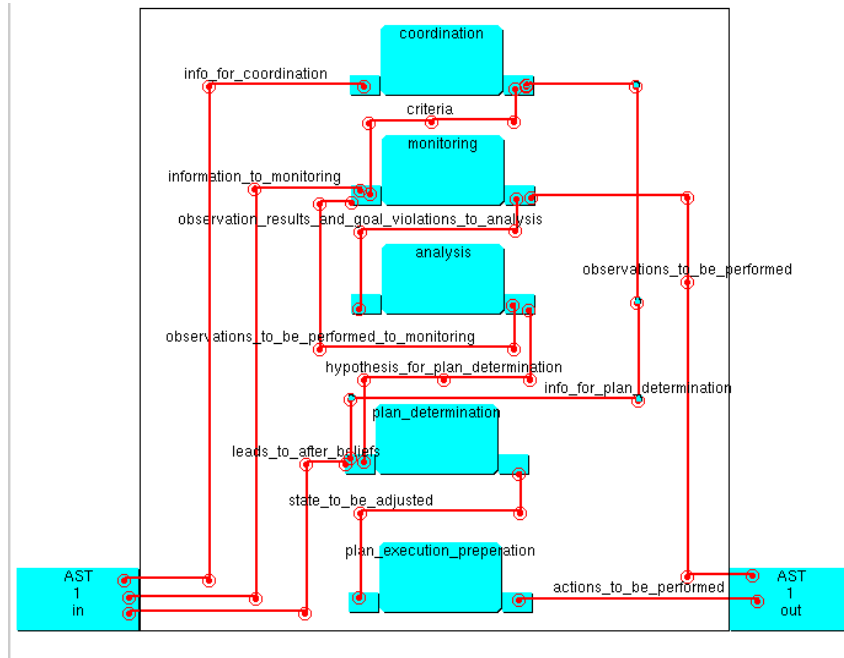


Figure 3. Monitoring and guidance model

5.1 Monitoring

Within the *Monitoring* component, it is determined what kinds of observation foci are needed to be able to verify whether the criteria hold. In the object identification example, this could be “identification event” (i.e. the event that the operator identified an object), and “correctness of identification”. The input and output ontology that is used within the Monitoring component is presented in Table 5.

The identified observation foci are translated into a number of concrete sensors being activated. As a form of refinement it is determined how specific information of a desired type can be obtained. For this a hierarchy of information types and types of sensors is used, as is information about the availability of sensors. For example, if the observation focus “identification event” is established, the monitoring component could refine this into two more specific observation foci “start identification” and “stop identification”. For the first observation an eye tracker could be turned on, while the second could be observed by looking at the events generated by a specific software component. Finally, *Monitoring* combines the detailed observations and reports the higher-level observation to *Analysis*.

5.2 Analysis

The *Analysis* component is responsible for detecting problems in the desired functioning of the human. If the *Analysis* component infers (based on a conflict between the criteria and the observations) that there is a problem, it aims to find cause of the problem. The input and output ontology is shown in Table 6.

Based on an appropriate dynamic model, hypotheses about the causes are generated using forward and backward reasoning methods (cf. [8]). First, temporal backward reasoning rules are used to derive possible hypotheses regarding the cause of the problem:

```

if    problem(at(S:STATE, I1:integers), pos)
then  derivable_backward_state(at(S:STATE, I1:integers));

if    leads_to_after(M:MODEL, S1:STATE, S2:STATE, I2:integers, pos)
and   derivable_backward_state(at(S2:STATE, I1:integers))
and   I3:integers = I1:integers - I2:integers
then  derivable_backward_state(at(S1:STATE, I3:integers));

if    intermediate_state(S:STATE) and derivable_backward_state(at(S:STATE, I:integers))
then  possible_hypothesis(at(S:STATE, I:integers))

```

Hereby, the first rule indicates that in case a problem is detected (a state *S* holding at a particular time point *I1*), then this is a derivable backward state. The second rule states that if a causal rule specifies that from state *S1* state *S2* can be derived after duration *I2* (represented via the *leads_to_after* predicate), and the state *S2* is has been marked as a derivable backward state (at *I1*), then *S1* is also a derivable backward state, which holds at *I1 - I2*. Finally, if something is a derivable backward state, and it is an internal state (which are the ones used as causes of problems), then this state is a possible hypothesis. Using such abductive reasoning of course does not guarantee that such hypotheses are correct (e.g. it might also be possible to derive *J* from another state). Therefore, the analysis component assumes one hypothesis (based upon certain heuristic knowledge, see e.g. [8]) and starts to reason forwards to derive the consequences of the hypothesis (i.e. the expected observations):

```

if    possible_hypothesis(at(S:STATE, I:integers))
then  derivable_forward_state_from(at(S:STATE, I:integers), at(S:STATE, I:integers));

if    leads_to_after(M:MODEL, S1:STATE, S2:STATE, I1:integers, pos)
and   derivable_forward_state_from(at(S1:STATE, I2:integers), at(S3:STATE, I3:integers))
and   I4:integers = I2:integers + I1:integers
then  derivable_forward_state_from(at(S2:STATE, I4:integers), at(S3:STATE, I3:integers));

if    observable_state(S1:STATE)
and   derivable_forward_state_from(at(S1:STATE, I1:integers), at(S2:STATE, I2:integers))
then  predicted_for(at(S1:STATE, I1:integers), at(S2:STATE, I2:integers));

```

The predictions are verified by doing requesting these observations from the *Monitoring* component. For example, if a hypothesis based on a cognitive model is that the undesired function is caused by an experienced pressure that is too high, then the observation focus will be set on the heart rate. The monitoring component selects the sensors to measure this. After these observation results come in, the selected hypothesis can be rejected in case the observations do not match the predicted observations. An example rule thereof is specified below:

```

if    observation_result(at(S1:STATE, I1:integers), neg)
and   selected_hypothesis(at(S2:STATE, I2:integers))
and   predicted_for(at(S1:STATE, I1:integers), at(S2:STATE, I2:integers))
then  to_be_rejected(S2:STATE);

```

Eventually, this leads to the identification of one or more specific causes of the problems, which are communicated to *Plan Determination*.

5.3 Plan Determination

Within *Plan Determination*, based on the identified causes of undesired functioning, plans are determined to remedy these. This makes use of causal relations between aspects in a dynamic model that can be affected and the (internal) states identified as causes of the undesired functioning. Hereby, backward reasoning methods (as explained for the *Analysis* component) are used. These use the specific cause of the problem as input, and derive what actions would remedy this cause. To decide which actions are best, the *Plan Determination* component also uses knowledge about the compatibility of solutions, their effectiveness and their side effects. See [8] for more a detailed overview of possible selection strategies. In the example, this component could conclude that the “noise level” should be reduced to lower the experienced pressure. The analysis component monitors the effectiveness of this measure. If it does not solve the problem, or causes undesired side effects, this will be considered as a new problem, which will be handled via the same process.

Subsequently, this information will be forwarded to the *plan determination* component. The *plan determination* component derives actions to be performed such that the work pressure experienced by the human does not exceed the maximal experienced pressure which has been set. An example of achieving this is by reducing the task level with a given percentage. If the percentage is calculated, two possible approaches can be followed, namely *task based reallocation*, and *task support*. In the former case, the reduction in task load is accomplished by reallocation of tasks. The tasks can be allocated to other humans, but the deadline of tasks could be altered as well (e.g. postponing tasks). The second option, *giving task support* reduces the task level by giving the human support, e.g. giving a calculator when having to perform calculations. Hereby, each support action reduces a certain amount of task level and appropriate support can be given.

The input and output ontology of the component is shown in Table 7. In the remainder of this section, two methods are presented to determine how much the task level should be reduced.

In the first method called what-if simulation, *plan determination* uses the input from *analysis* concerning the predicted trend of the experienced pressure. Suppose the prediction is that the experienced pressure will become too high, namely x_1 in y time units (which is above the boundary b that has been set). The component performs a what-if simulation using the work pressure model whereby the task level has been reduced by $z\%$ to investigate how effective such a change is. This results in a prediction that the experienced pressure will become x_2 after y time units. Now, the *plan determination* component can calculate how much the task level should be reduced in order to stay precisely within the boundaries that have been set:

$$\text{task_level_reduction_percentage} = (x_1 - b) / ((x_1 - x_2) / z)$$

Here, $x_1 - b$ is assumed to be sufficiently small, so that a linear approximation of the experienced pressure function can be used.

Figure 4 shows an example of a prediction. The current time point is 30, and the boundary for maximum experienced pressure (b) is set to 0.186. Given the current trend in the task level, the experienced pressure is prospected to surpass the maximum level in 3 time units, whereby the predicted experienced pressure is $x_1 = 0.1863$ to be precise. The *plan determination* component performs another what-if simulation whereby the task level is reduced with $z = 10\%$ in the future. This prediction is shown in Figure 4 as well. In this prediction, the experienced pressure with a task level reduction of 10% will be $x_2 = 0.183$ after 3 time units.

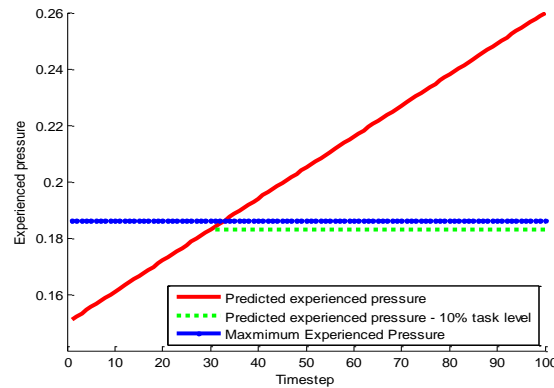


Figure 4. What-if simulation for plan determination

Plan determination has enough information to calculate how much the task level should be reduced at time point t , given this prediction:

$$(0.1863 - 0.186) / ((0.1863 - 0.183) / 10) = 0.91\%.$$

Another approach is to use a *fixed strategy* to reduce the task level. Hereby, a strategy is utilized which increases the reduction in task level with each consecutive time point at which an experienced pressure exceeding the maximum experienced pressure boundary b is predicted. The fixed strategy proposed in this paper is to decrease the task level with a percentage equal to the number of time units at which the prediction is that the boundary b will be crossed. As a result, an increasing decrease in task level will be performed until the prediction no longer forecasts exceeding b . So, at time point 1 the task pressure will be decreased with 1%, at time point 2 with 2% (in addition to the previous 1%), etcetera until the prediction is satisfactory.

5.4 Plan Execution Preparation

Finally, within *Plan Execution Preparation* the plan is refined by relating it more specifically to certain actions that have to be executed at certain time points. For example, reducing the noise level could be achieved by reducing the power of an engine, or closing a door.

6. A Prototype Implementation

A prototype of the system has been implemented in DESTOOL (the prototyping environment for the DESIRE modelling framework [9]). This prototype has been used to evaluate the model for a specific scenario as specified by domain experts of the Royal Dutch Navy. The scenario concerns the mechanic Dave, who works on a ship of the Navy:

Dave just started his shift when he got an alarm that he had to do a regular check in the machine room; he accepted the alarm and walked towards the room. There he heard a strange sound and went to sit down to find the solution. However, he could not identify the problem and a solution immediately. At the same time, Dave received a critical alarm on his PDA: the close-in weapon system (CIWS) of the ship was broken. He immediately accepted the alarm, however continued to work on the engine problem, resulting in the more critical task to fix the close-in weapon system not being performed according the schedule.

To apply the approach presented in this paper for this scenario, a number of models have been specified. First of all, the workflow models for the two tasks from the operator's task package have been specified. For the sake of brevity, these models are not shown, but specified

in [10]. Furthermore, a cognitive model concerning the experienced pressure is specified, which is shown in Figure 5.

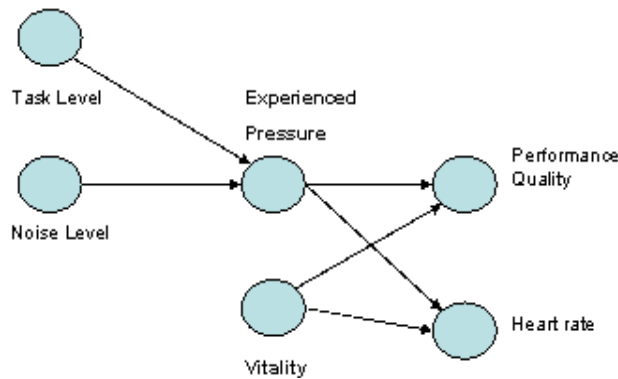


Figure 5. Cognitive model for operator

In the model, relations between the states have been identified using the `leads_to_after` predicate, specified by means of four parameters: the model name, a condition state, a consequence state, and a delay between the two. For instance, the relation

```
leads_to_after(cogn1, and(normal_exp_pressure, normal_vitality), high_perf_quality, 1)
```

indicates that a normal experienced pressure combined with normal vitality leads to a high performance quality of the task in one step.

The presented scenario has been simulated within the prototype of the proposed architecture. Below, a brief overview of the steps the system takes is presented. When the system is started, the operator's task package that comprises two task types `maintain_engine` and `solve_ciws_problem` is provided to *Own Process Control* of *SMPA*. The operator is characterized by the default profile with standard characteristics (e.g., the heart rate range is 60-100 beats per minute). Furthermore, a set of generic goals provided to *Own Process Control* is defined to achieve timely task execution for each task, and to maintain a good health for the human it supports. The goal related to the operator's health is further refined stating that the experienced pressure and the vitality should remain normal:

```
own_characteristic(has_goal(SMPA, achieve(ontime_task_execution, -1))
own_characteristic(has_goal(SMPA, maintain(good_health_condition, 0, -1)))
own_characteristic(has_goal(SMPA, maintain(normal_exp_pressure, 0, -1)))
own_characteristic(has_goal(SMPA, maintain(normal_vitality, 0, -1)))
```

Here, '-1' indicates infinite time. Based on the goals related to the operator's health condition, the query for a cognitive model with the value `normal_exp_pressure` of the parameter `states` is generated and communicated by *Own Process Control* to *MMA*. As a result of this query, the model annotated by the corresponding parameters is indeed retrieved from *MMA*, and stored within the component *MAI* within *SMPA*:

maintenance of agent information (SMPA)

input: belief(leads_to_after(cogn1, and(normal_exp_pressure, normal_vitality), high_perf_quality, 1), pos) etc.

output: see input

The workflow models for the assigned tasks are extracted from *MMA* in a similar manner.

Eventually, the models and the goals are also received by the *Coordination* component in *Agent Specific Task*. Based on this input *Coordination* generates specific criteria. In particular, based on the goals to maintain `normal_exp_pressure` and `normal_vitality`, the criteria to maintain the medium heart rate and the high performance quality are generated using the cognitive model. The generated criteria are provided to the *Monitoring* component, which sets the observation foci corresponding for these criteria.

After this has all been done, a new assignment of a task is received from the *World* component, namely that a task of type `maintain_engine` has been assigned to the operator:

physical world

input: -

output: `observation_result(at(assigned_task_at(maintain_engine, 3), 3), pos)`

Based on this information *Coordination* generates new criteria using the workflow model corresponding to the task. Most of these criteria establish the time points at which the execution states from the workflow should hold (e.g., `achieve(walk_to_engine, 4)`).

These criteria are again sent to the *Monitoring* component within *Agent Specific Task*. Therefore, the component sets the observation foci to the states within the workflow. If no goal violation is detected, no actions are undertaken by the agent. After a while however, a new task is assigned, namely the task to fix the close-in weapon system (of type `solve_ciws_problem`), which is outputted by the world:

`observation_result(at(assigned_task_at(solve_ciws_problem, 23), 23), pos)`

Again, the appropriate criteria are derived based on the corresponding workflow model. The *Monitoring* component continuously observes whether the criteria are being violated, and at time point 66 (when the operator should walk to the close-in weapon system) it observes that this is not the case. Therefore, a criterion violation is derived by the *Monitoring* component.

monitoring (AST - SMPA)

input: `observation_result(at(walk_to_ciws, 66), neg)` etc.

output: `criterion_violation(walk_to_ciws)` etc.

This criterion violation is received by the component *Analysis*, which is triggered to start analysing why the operator did not perform the task in a timely fashion. This analysis is performed using the cognitive model. The first hypothesis which is generated is that the cause is that the experienced pressure is normal, but the vitality abnormal. The *Analysis* component derives that a low heart rate must be observed to confirm this hypothesis (an observation that is not available yet):

analysis (AST - SMPA)

input: `observation_result(at(walk_to_ciws, 66), neg); criterion_violation(walk_to_ciws)`

output: `selected_hypothesis(at(and(normal_exp_pressure, abnormal_vitality), 65); to_be_observed(low_heart_rate))`

Since the heart rate is not observed to be low, but high, the *Analysis* component selects another hypothesis that is confirmed by the observation results that are now present (after the heart rate has been received). The resulting hypothesis is abnormal experienced pressure, and normal vitality. This hypothesis is passed on to the *Plan Determination* component within *Agent Specific Task* of the *SMPA* agent. *Agent Specific Task* derives that the task level should be adjusted:

plan determination (AST - SMPA)

input: `selected_hypothesis(at(and(abnormal_exp_pressure, normal_vitality), 65)`

output: `to_be_adjusted(abnormal_task_level)`

To achieve this adjustment, the operator is informed that the maintenance task is not so important, and that the operator should focus on the close-in weapon system task. This eventually results in a normal task level of the operator.

7. User Study

A user study was conducted to examine the effectiveness of the support given by the ambient agent. One experienced male participant took part, with the main task to perform tasks in a simulation-based training environment. The environment and the procedure of the study are described in Sections 7.1 and 7.2. Section 7.3 and 7.4 explain how the agent uses the user information for respectively monitoring and analysis of human performance.

7.1 Simulation-based training environment

In the study, the main task consisted of identifying incoming contacts and, based on the outcome of the identification, deciding to eliminate the contact (by shooting) or allowing it to land (by not shooting). The object at the bottom of the screen represents the participant's (stationary) weapon. In addition, contacts (allies and enemies in the shape of a dot each accompanied by a simple mathematical equation) will appear at a random location on the top of the screen and fall down to random location at the bottom of the screen. The rate at which the contacts appear can vary in demanding versus less demanding circumstances.

The identification of a contact is performed by checking the correctness of its equation, incorrect equations correspond to enemies and correct ones to allies. Points are gained by shooting down the enemies and by allowing the allies to land. The participant can shoot a missile by executing a mouse click at a specific location; the missile will then move from the weapon to that location and explode exactly at the location of the mouse click. When a contact is within a radius of 50 pixels of the exploding missile, it is destroyed. The number of points a participant receives for destroying an enemy is proportional to the proximity of the explosion. When a participant shoots an ally or when an enemy reaches the bottom of the screen 10000 points are lost. When an ally reaches the bottom of the screen the participant receives 1000 points.

7.2 Procedure

The user study consisted of two sessions; the first was used to measure the participant's profile and the second session was used for examining the effect of the support system using the model. For the first session, the participant started with the first part by filling out a personality questionnaire, which contained questions from the NEO-PI-R and the NEO-FFI [11]. Answers to these questions served as input for the participant's personality profile. After the questionnaire, the participant performed the three small tests in order to determine his basic cognitive abilities and expertise profile (as explained in Section 2.1).

The goal of the second session was to test the effectiveness of support using the personalized model. For this, the participant performed two conditions of the task explained in Section 3.1. In condition 1 support was given according to the personalized model. In condition 2 no support was given. In both conditions the situational demands were high as every 2.25 to 4.5 seconds one contact entered the screen and the complexity of the equations was relatively high (e.g. $271/17=23$). The participant started with condition 1 ("support") and after some rest he continued with condition 2 ("no support"). In both conditions the task duration was 10 minutes.

7.3 Agent-Based Reasoning

In this section, the specific choices that are made regarding the user study are explained. for each component within the personal agent.

7.3.1 Monitoring. For the present user study, the monitoring component makes two types of measurements in the external world, initial one-time measurements and measurements during the simulation-based training program. The one-time measurements determine the initial settings of the work pressure model for the expertise and personality profiles and basic cognitive abilities (further explained in Section 2.2). The first initial monitoring task is measuring four personality characteristics via questions from the NEO-PI-R and the NEO-FFI personality questionnaires [11]: extraversion, neuroticism, vulnerability and ambition. The second part of the initial measurements consists of three small tests. In the first test simple Reaction Time is measured (*choice*). In the second test (*calc*) humans have to solve mathematical equations. In the third test a human has to move the mouse to a target presented on the screen (*mouse*). In the second and the third test the human's speed and accuracy are measured. Data from all tests is transferred to the analysis component.

During the training program, the analysis component will request observations to be able to determine (using the work pressure model described in Section 2.2) whether the work pressure level of the human remains below the boundary. The observations required for this are the actions of the human in order to calculate performance quality and the situation on the screen in order to calculate situational task demand. These observations are requested by the analysis component continuously and will be sent back to analysis continuously.

7.3.2 Analysis. The model that the analysis component uses in the present user study is a dynamic model about work pressure to detect problems in human functioning (Figure 6, adapted from [5]). The key concept in this model is the experienced pressure. The analysis component maintains a maximal desired value for this concept, called the experienced pressure norm. Using the work pressure model, the estimated experienced pressure for a time point in the near future (set to 3 minutes in the experiment) is continuously calculated. This calculation requires input about the functioning of the human. The analysis component therefore continuously requests information from the monitoring agent about the task execution state, which is measured via the scores that are received in simulation-based training environment. If the predicted experienced pressure will be higher than the norm, the analysis component will conclude that the experienced pressure norm will be exceeded and the following assessment will be derived:

```
assessment(predicted_threshold_exceeded_by(exp_pressure, x))
```

Subsequently, this information will be forwarded to the plan determination component, which will plan the measures to be taken.

The work pressure model is based on two different theories: 1) the cognitive energetic framework [12], which states that effort regulation is based on human resources and determines human performance in dynamic conditions; 2) the idea, that when performing sports, a person's generated power can continue on a *critical power* level without becoming more exhausted [13]. According to the model, a person's experienced pressure is influenced by a combination of *exhaustion*, the amount of *generated effort*, but also the *critical point*. The critical point is the amount of effort someone can generate without becoming more exhausted. In addition, the pressure is determined by external factors (task demands and environment state) and personal factors (cognitive abilities and personality profile) taken from the literature ([14], [15]). More details of the work pressure model can be found in [5].

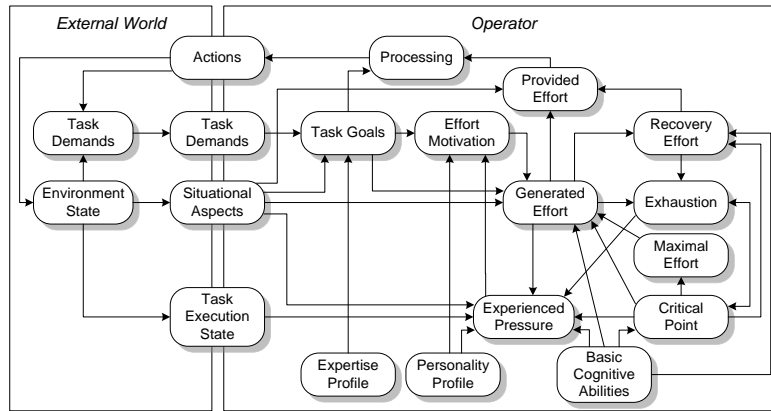


Figure 6. A graphical representation for the work pressure model [5]

7.3.3 Plan Determination. In the condition with support, the task-based reallocation was used (described in section 2.3). In this scenario, this indicates that when support is needed, a number of cases are reallocated (e.g., disappear from the screen). Support is given based on the (predicted) level of experienced pressure: if this exceeds 0.8 within 180 seconds the fixed-strategy (see Section 2.3) is used.

8. Analysis of Experimental Results

This section analyses the results obtained during the experiment. First, the key results are presented in a graphical manner. Thereafter a more formal verification of the experimental results is performed.

8.1 Comparison With and Without Support

In this subsection a thorough comparison is made between the two conditions of the experiment (i.e. with and without support). Obviously, one clear metric on the performance is the performance quality during the experiment. The results hereof are shown in Figure 7. On the x-axis of the figure time is shown in seconds, whereas the y-axis represents the performance quality (for the two dark lines, the scale is shown on the left side of the figure starting at 0.6), and the reallocation percentage (for the gray line, the scale is shown on the right side). Of course, the reallocation percentage belongs to the case with support, as no reallocation takes place in the other condition. It can be seen that the performance quality is generally higher for the case with support. Furthermore, as the performance quality decreases, the task allocation percentage increases, resulting in the performance quality going up again. Hence, the support is very effective. In addition the final score of the user in the support condition was much higher than in the no-support condition (600000 vs 50000 points).

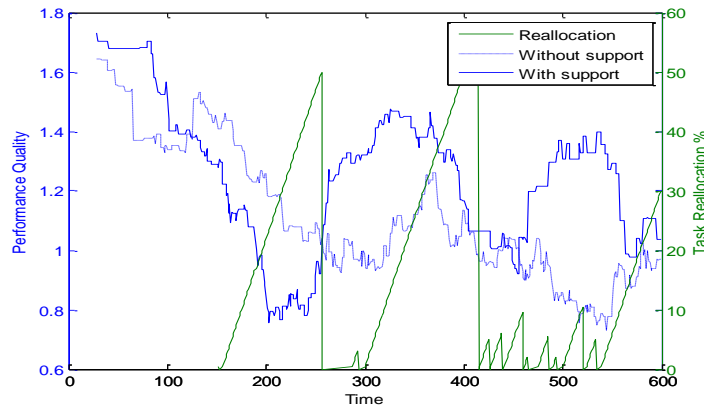


Figure 7. Performance quality with and without support

Besides the performance quality, the internal concepts in the model also give an indication how good the support functions. Figure 8 shows the internal state experienced pressure for the case with and without support. It can clearly be seen that the experienced pressure is a lot lower for the case with support. The task reallocation stops the increase of the experienced pressure (and sometimes even makes it decrease).

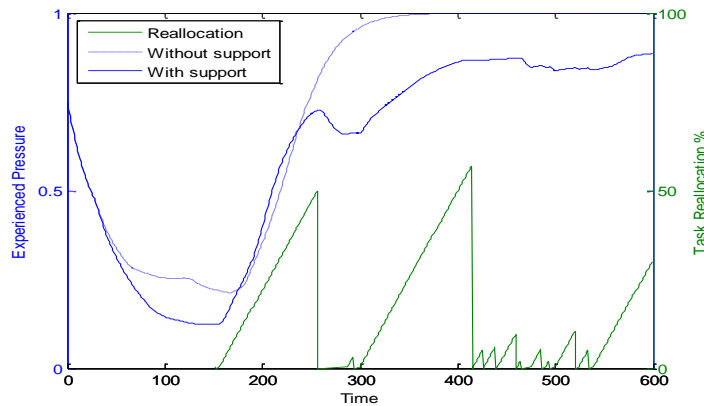


Figure 8. Experienced pressure with / without support

Finally, in Figure 9 the internal state exhaustion is shown. Hereby, the exhaustion for the condition without support is far higher than the condition with support. Only between time point 300 and 400 there is some exhaustion in the case with support. In the other condition, exhaustion builds up, and never disappears.

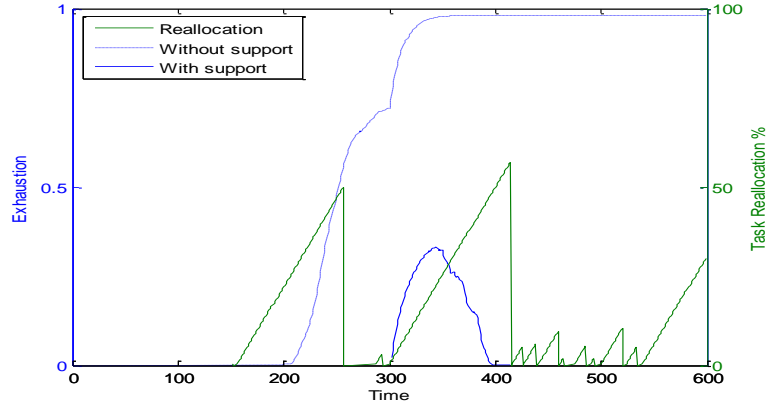


Figure 9. Exhaustion with and without support

8.2 Formal Analysis

Besides the analysis of the experimental results, the results have also been analyzed by verification of dynamic properties. Following [16], dynamic systems can be studied by specifying dynamic statements that are (or are not) expected to hold in terms of temporal logical expressions, and automatically verifying these statements against logs of the system. A typical example of a property that may be checked for the agent proposed in this paper is whether the performance of users is higher in the scenarios with support than in the scenarios without support. However, more functional properties of the ambient agent may be checked, such as ‘every time that the system assesses that there is a risk for a critical situation, it will provide some form of support’.

For the ambient agent presented in this paper, a number of such dynamic properties have been formalized in the language TTL [16]. TTL is built on atoms referring to *states* of the world, *time points* and *traces*, i.e. trajectories of states over time. In addition, *dynamic properties* are temporal statements that can be formulated with respect to traces based on the state ontology *Ont* in the following manner. Given a trace γ over state ontology *Ont*, the state in γ at time point t is denoted by $\text{state}(\gamma, t)$. These states can be related to state properties via the formally defined satisfaction relation denoted by the infix predicate \models , comparable to the Holds-predicate in the Situation Calculus: $\text{state}(\gamma, t) \models p$ denotes that state property p holds in trace γ at time t . Based on these statements, dynamic properties can be formulated in a formal manner in a sorted first-order predicate logic, using quantifiers over time and traces and the usual first-order logical connectives such as \neg , \wedge , \vee , \Rightarrow , \forall , \exists . Below, a number of dynamic properties are introduced, both in semi-formal and in informal notation.

P1 - Support Correctness

For all time points t ,
if the system assesses that the experienced pressure norm will be exceeded
then within one time step the system will perform case reallocation.

$$\begin{aligned} P1(\gamma:\text{TRACE}) \equiv & \forall t:\text{TIME} \forall x:\text{REAL} \text{ state}(\gamma, t) \models \text{assessment}(\text{threshold_exceeded_by}(\text{exp_pressure}, x)) \\ & \Rightarrow \exists t2 [t \leq t2 \leq t+1 \ \& \ \text{state}(\gamma, t2) \models \text{to_be_performed}(\text{case_reallocation})] \end{aligned}$$

This property has been checked against two traces γ . The property was satisfied for trace1, a log of the experiment under the “support” condition, whereas it failed for trace2 (a log of the experiment under the “no support” condition). This proves that the support system indeed performed an action in each case that it deemed this necessary.

As a next step, the effect of the support actions on the user's performance quality, experienced pressure, and exhaustion was verified. To this end, the average values of these concepts were compared for the different traces. The average value of performance quality is specified in property P2:

P2 - Average Performance i

For trace γ , the average performance quality that a person has is i .

Note that the formal form of this property has been omitted for the sake of brevity. In addition to P2, variants of the property have been specified in order to calculate the average values of experienced pressure and of exhaustion. The results of checking these properties to trace1 and trace2 are shown in Table 8. Note that the averages are calculated over traces with a length of 10 minutes, where each second a new value was logged (600 measurements). As shown in Table 8, the agent's support increased the average performance quality with an increase of 13%. Moreover, the support decreased the (estimated) experienced pressure with 13.5% and decreased the (estimated) exhaustion almost completely (94.8%).

9. Discussion

In every organisation a set of critical tasks exists that greatly influence the satisfaction of important organisational goals. Thus, it is required to ensure effective and efficient execution of such tasks. To this end, automated personalized assistance for the task performers may be used. In this paper, a multi-agent system architecture for personal support during task execution has been proposed. This architecture includes self-maintaining personal assistant agents with a generic design. Such agents possess self-configuration abilities, which enable them to dynamically load domain-specific models, thereby specializing these agents for the execution of particular tasks in particular domains. Using these models and information about the assigned goals and tasks, the personal assistant agent performs monitoring and analysis of the behaviour of the supported human in his/her environment. In case a known problem is detected, the agent tries to identify and execute an appropriate repair action. The fact that the architecture is generic differentiates the approach from other personal assistants such as presented in [2]; [3]. Besides being generic, the proposed self-maintaining personal assistant agent has an advantage of being relatively lightweight, as it only maintains and processes those models that are actually needed for the performance of the tasks. It can therefore run upon for instance a PDA or cell phone.

When performing a task, especially in highly demanding circumstances, human performance can be degraded due to increased cognitive workload. A possible negative effect of high cognitive workload is that it leads to a reduction in attention and situation awareness [17]. Situation awareness refers to the picture that people have of the environment (e.g., [18]). In case of low situation awareness this picture is wrong, which will often lead to wrong decision making (e.g., [19]). In the literature, it is known that automated systems can also impose a negative effect on cognitive workload or situation awareness [20]. Therefore, systems have been designed that are adaptive, e.g. in only providing aiding when it is necessary [4]. For this, a human's cognitive state should be assessed online; since this is difficult, often adaptive systems like this are based on psychophysiological measurements, like brain activity and eye movements (e.g. [4]; [21]). The personal assistant described in this paper makes use of such measurements, but in addition uses models of cognitive states and dynamics, and the current workflow to be able to assess the online state of the operator. This allows for an optimal operator support.

In this paper the application of the multi-agent study in a user study shows the effectiveness of such highly personalized operator support during execution of a demanding task. The

assistance provided by a personal assistant agent in this system is sensitive not only to the task and environmental conditions at hand but also to personal aspects such as the characteristics and states of the human at the given point in time. It constantly monitors the task execution and well-being of the human via non-intrusive sensors, and intervenes when an unsatisfactory situation is expected. Analysis takes place on the basis of continuously made predictions using observations and a dynamical model for the human's work pressure and exhaustion. Experiments have shown that the human's performance increases up to around 13% in a scenario with support. In addition, the subject in the experiments reported that he had the feeling that he was able to handle the situation, in contrast to a scenario where no support was provided.

A number of approaches for prediction of human behavior in ambient intelligence environments have been developed. Most of these approaches perform time series analysis based on sensory data collected. In particular, data mining prediction techniques (e.g., case-based reasoning [22]), soft computing prediction techniques (e.g., fuzzy rule based learning [23]) and statistical modeling prediction techniques (e.g., based on Markov chains [24]) are used often. Usually cognitive models underlying human behavior are not considered in these approaches. However, combining information about observations of human behavior with dynamic specifications of internal processes as proposed in this paper provides a stronger basis for prediction. To ensure correct prediction and appropriate support, cognitive specifications should be precise and valid. The work pressure model used in this paper has a strong support from psychology [14]; [15].

Some characteristics and behavioral modes of the human may vary during the task execution. To ensure high reliability of predictions and adequacy of the support provided by the assistant, real-time fine-tuning of the parameters of the work pressure model can be performed based on the observed human behavior and characteristics. To this end, real-time parameter estimation techniques can be applied, based on the global probabilistic optimization, gradient-based algorithms or filters [25]. In the future a dedicated component will be elaborated and added to the proposed agent architecture, which realizes one of these techniques. In addition, in future research the proposed architecture will be applied for supporting different types of tasks performed by different types of humans. Furthermore, the performance parameters of the technical realization of the system will be evaluated. In particular, the frequency of interaction and the amount of information transmitted between the maintenance agents and the personal assistant agents will be evaluated.

10. References

- [1] Posner, M. I., and Boies, S. J. 1971. Components of attention. *Psychological Bull.* 78:391-408.
- [2] Modi, P.J., Veloso, M., Smith S.F., and Oh, J., CMRadar: A Personal Assistant Agent for Calendar Management. In: Bresciani, P. et al. (eds.), *AOIS II, LNCS 3508*, Springer, 2005, pp. 169-181.
- [3] Myers, K., Berry, P., Blythe, J., Conley, K., Gervasio, M., McGuinness, D.L., Morley, D., Pfeffer, A., Pollack, M., and Tambe, M., An Intelligent Personal Assistant for Task and Time Management. *AI Magazine Summer 2007*, pp. 47-61.
- [4] Wilson, G.F., & Russell, C.A. (2007). Performance enhancement in an uninhabited air vehicle task using psychophysiological determined adaptive aiding. *Human Factors*, 49(6), 1005-1018.
- [5] Bosse, T., Both, F., Lambalgen, R. van, and Treur, J. (2008), An Agent Model for a Human's Functional State and Performance. In: Jain, L. et al. (eds.), *Proceedings of IAT'08*. IEEE Computer Society Press, 2008, pp. 302-307.
- [6] Brazier, F.M.T., Jonker, C.M., and Treur, J. (2000). Compositional Design and Reuse of a Generic Agent Model. *Applied AI Journal*, vol. 14, 2000, pp. 491-538.

- [7] Appendix A: <http://www.double-blind.741.com/pa.html>
- [8] Duell, R., Hoogendoorn, M., Klein, M.C.A., and Treur, J., An Ambient Intelligent Agent Model using Controlled Model-Based Reasoning to Determine Causes and Remedies for Monitored Problems. In: Proceedings of the Second International Workshop on Human Aspects in Ambient Intelligence, HAI'08. IEEE Computer Society Press, 2008.
- [9] Brazier, F.M.T., Dunin Keplicz, B., Jennings, N., and Treur, J., DESIRE: Modelling Multi-Agent Systems in a Compositional Formal Framework. *Int. J. of Coop. Inf. Systems*, vol. 6, 1997, pp. 67-94.
- [10] Appendix B: <http://www.double-blind.741.com/Models.pdf>
- [11] Costa Jr, P.T. and McCrae, R.R., Revised NEO Personality Inventory (NEO-PI-R) and the NEO Five-Factor Inventory (NEO-FFI) professional manual, Psychological Assessment Resources, Odessa, FL (1992).
- [12] Hockey, G.R.J. (1997). Compensatory control in the regulation of human performance under stress and high workload: a cognitive-energetical framework. *Biol. Psychology* 45, 73-93.
- [13] Hill, D.W. (1993). The critical power concept. *Sports Medicine*, vol.16, pp. 237-254.
- [14] Rose, C.L., Murphy, L.B., Byard, L., & Nikzad, K. The role of the Big Five personality factors in vigilance performance and workload. *European Journal of Personality* 16: 185-200 (2002).
- [15] Plomin, R., and Spinath, F.M. Genetics and general cognitive ability. *Trends in Cognitive Science* 6(4), 369-176 (2002).
- [16] Bosse, T., Jonker, C.M., Meij, L. van der, Sharpanskykh, A., and Treur, J. (2009), Specification and Verification of Dynamics in Agent Models. *Int. J. of Coop. Information Systems*, vol. 18, 2009, pp. 167-193.
- [17] Wickens, C.D. (2002). Situation awareness and workload in aviation. *Current Directions in Psych. Science*, 11, 128-133.
- [18] Endsley, M.R. (2000). Theoretical underpinnings of situation awareness. In M.R. Endsley & D.J. Garland (Eds.) *Situation awareness analysis and measurement* (pp. 1-21). Mahwah, NJ: Erlbaum.
- [19] Endsley, M.R. (1997). The role of situation awareness in naturalistic decision making. In C. Zsombok G. Klein (Eds.), *Naturalistic decision making*: 269- 284. Mahwah, NJ: Erlbaum.
- [20] Parasuraman, R., & Riley, V. (1997). Humans and automation: use, misuse, disuse, abuse. *Human Factors*, 39(2), 230-253.
- [21] Prinzel, L.J., Freeman, F.G., Scerbo, M.W., Mikulka, P.J., Pope, A.T. (2000). A closed-loop system for examining psychophysiological measures for adaptive task allocation. *Int. Journal of Aviation Psychology*, 10(4), 393-410.
- [22] Ma, T., and Y. Kim, Context-Aware Implementation based on CBR for smart home, *IEEE Trans. Wireless And Mobile Computing, Networking And Comm.*, pp.112-115, 2005.
- [23] Rutishauser, U., J. Joller and R. Douglas, Control and learning of ambience by an intelligent building, *IEEE Transaction on Systems, Man, and Cybernetics-Part A: Systems and Humans*, vol.35, pp.121-132, 2005.
- [24] Kaushik, A.R., B.G. Celler and E. Ambikairajah, A methodology to monitor the changing trends in health status of an elderly person by developing a Markov model, *Proc. of the 27th Annual Conference on Engineering in Medicine and Biology*, pp.2171-2174, 2005.
- [25] Sorenson, H.W. *Parameter estimation: principles and problems*. Marcel Dekker, Inc., New York, 1980.

Table 1. Interaction ontology

Ontology element	Description
to_be_observed(I:INFO_ELEMENT)	I is to be observed in the world (active observation)
observation_result(I:INFO_ELEMENT, S: SIGN)	I with sign S is observed in the world
communicated_by(I:INFO_ELEMENT, S:SIGN, A:AGENT)	I with sign S is communicated by A
to_be_communicated_to(I:INFO_ELEMENT, S:SIGN, A:AGENT)	I with sign S is communicated to A
to_be_performed(A:ACTION)	Action A is to be performed

Table 2. Monitoring/guidance model parameters

Parameter	Some possible values
Name	string value
Type of analysis	diagnosis, configuration, classification, causal network analysis
Reasoning techniques	forward/backward chaining, tableau-like methods, constraint reasoning
Required time	integer value
Required memory	integer value
Processing power	integer value
Language type	causal logic-based relations
Characteristics	qualitative or quantitative or combination of both; stochastic; statistical; discrete or continuous

Table 3. Cognitive and workflow model parameters

Parameter	Some possible values	Parameter	Some possible values
Cognitive models		Workflow models	
Name	string value	Name	string value
Cognitive processes	reasoning, consciousness, perception	Task type	string value
States	stress, motivation level, fatigue level	Task executor capabilities	excellent analytic skills, quick typing, domain-related knowledge
Agent type	robot, human, animal	Task executor traits	openness, extraversion, neuroticism
Related physical parts	frontal lobe, parietal lobe, temporal lobe	Minimum/maximum duration	integer value / integer value
Characteristics	qualitative/quantitative; stochastic; statistical	Consumable resources	building materials

Table 4. General goals defined for the self-maintaining personal assistant agent

#	Goal
1	It is required to achieve the timely task execution
2	It is required to achieve a high degree of effectiveness and efficiency of the task execution
3	It is required to achieve a high degree of safety of the task execution
4	It is required to maintain the compliance to a workflow for an assigned task
5	It is required to maintain an acceptable level of experienced pressure during the task execution
6	It is required to maintain the operator's health condition appropriate for the task execution
7	It is required to maintain a satisfactory health condition of the operator

Table 5. Input/output ontology of Monitoring.

Predicate	Explanation
Input	
to_be_observed: INFO_ELEMENT	The component is requested by <i>analysis</i> to observe a certain information element.
observation_result: INFO_ELEMENT x SIGN	An observation result is received from the <i>external world</i> indicating the information element observed, and the value thereof (i.e. pos or neg)
Output:	
to_be_observed: INFO_ELEMENT	An active observation to be performed in the <i>external world</i> . Note that a translation occurs between the information requested by <i>analysis</i> , and the observations performed in the world.
observation_result: INFO_ELEMENT x SIGN	An observation result passed on to <i>analysis</i> . Again, a translation takes place in the monitoring component.

Table 6. Input/output ontology of Analysis

Predicate	Explanation
Input	
observation_result: INFO_ELEMENT x SIGN	An observation result is received from <i>monitoring</i> , indicating the information element observed, and the value thereof (i.e. pos or neg)
Output:	
to_be_observed: INFO_ELEMENT	<i>Analysis</i> outputs a request to observe a certain information element.
assessment: ASSESSMENT_VALUE	The component has assessed the current situation, and detected a problem. Therefore, the assessment is outputted, and passed on to <i>plan determination</i> .

Table 7. Input/output ontology of Plan Determination

Predicate	Explanation
Input	
assessment: ASSESSMENT_VALUE	The assessment of the situation comes in from <i>analysis</i> .
Output:	
to_be_performed: ACTION	The <i>plan determination</i> component has derived that a certain action should be performed, e.g. reallocation of a percentage of the tasks.

Table 8. Averages of performance quality, exhaustion, experienced pressure

	Trace 1 (support condition)	Trace 2 (no-support cond.)
Performance quality	1.188	1.051
Experienced pressure	0.556	0.643
Exhaustion	0.026	0.498