A Generic Personal Assistant Agent Model for Support in Demanding Tasks

Tibor Bosse[1], Rob Duell[2], Mark Hoogendoorn[1], Michel Klein[1], Rianne van Lambalgen[1], Andy van der Mee[2], Rogier Oorburg[2], Alexei Sharpanskykh[1], Jan Treur[1], and Michael de Vos[2]

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

Abstract. Human task performance may vary depending on the characteristics of the human, the task and the environment over time. To ensure high effectiveness and efficiency of the execution of tasks, automated personal assistance may be provided to task performers. A personal assistant agent may constantly monitor the human's state and task execution, analyse the state of the human and task, and intervene when a problem is detected. This paper proposes a generic design for a Personal Assistant agent model which can be deployed in a variety of domains. Application of the Personal Assistant model is illustrated by a case study from the naval domain.

1 Introduction

Human task performance can degrade over time when demanding tasks are being performed. Such degradation can for instance be caused by available resources being exceeded [1]. Furthermore, the effectiveness and efficiency of the task execution are often dependent on the capabilities, experience, and condition of the actor performing the task. Different actors may require different degrees of assistance and various resources for the task execution. High effectiveness and efficiency levels are of particular importance for critical tasks. Furthermore, as a longer term aim, the human should remain healthy during the processes of task execution. To overcome the limitations of human cognition (e.g. in attention span, working memory and problem solving), the term *augmented cognition* (AugCog) has been proposed, which can be defined as a research field that aims at supporting humans by development of computational systems that 'extend' their cognition [2].

As examples of AugCog, intelligent personal assistants exist that support humans during the execution of tasks (see e.g. [3], [4]). 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. [5]). The considered aspect of human behaviour and of the execution of tasks is unique. 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 Personal Assistant agent model. The Personal Assistant can use specific dynamical models to monitor and analyse the current processes of the human. Specific sensors measure the human's psychophysiological state (e.g., heart rate) and the state of the environment (e.g., noise) to detect a possible problem and to test hypotheses. If needed, intervention actions are selected for the specific state, domain and task.

The paper is organized as follows. The generic model for a Personal Assistant agent which performs monitoring and guidance is described in Section 2. A scenario realised in a prototype implementation is described in Section 3. The multi-agent context for the Personal Assistant agent is described in Section 4. Finally, Section 5 concludes the paper.

2 The Generic Personal Assistant Agent Model

The *personal assistant agent* (PA) supports a human during the execution of a task. A personal assistant's main function is *monitoring and guidance* of the human to whom it is related. Personal assistants also interact with the physical world by performing observations (e.g., of the human's actions and their effects).The agent model for PA was 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. The component *Own Process Control* initiates and coordinates the internal agent processes. In the component *Agent Specific Task*, domain-specific tasks were modelled, in particular monitoring and guidance. At the highest abstraction level the component consists of 5 subcomponents: *Coordination*, *Monitoring*, *Analysis*, *Plan Determination*, and *Plan Execution Preparation.*

2.1 Coordination

The initial inputs for the process are the goals provided from PA's *Own Process Control* component, which are refined within the *Coordination* component into more specific criteria that should hold for the human's functioning (e.g., 80% of certain objects on a radar screen should be identified within 30 seconds). 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 human, the criteria for particular task execution states may be generated from task-related goals. More specifically, 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 human's well-being (see Table 1). Goals of two types are distinguished:

(1) achievement goals (e.g., goals 1-3 in Table 3) 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 3) 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))
```

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 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 (*'It is required to maintain a satisfactory health condition'*) expresses '*It is required to maintain the human's heart rate within the acceptable range'*. Based on the available information about the physical characteristics of the human (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 human'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 criteria are fed to the *Monitoring* component, which is discussed below.

Table 1. General goals defined for the 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 human's health condition appropriate for the task execution
7	It is required to maintain a satisfactory health condition of the human

2.2 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" (i.e. the event that the human identified an object).

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” 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*.

2.3 Analysis

If the *Analysis* component infers (based on a conflict between the criteria and the observations) that there is a problem, it aims to find a cause of the problem. Based on an appropriate dynamic model, hypotheses about the causes are generated using forward and backward reasoning methods (cf. [7]). First, temporal backward reasoning rules are used to derive a possible hypothesis 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 with a specific model (represented via the leads_to_after predicate), and the state S2 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. [7]) 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 a request from the *Monitoring* component to perform these observations. 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*.

2.4 Plan Determination

Within *Plan Determination,* based on the identified causes of undesired functioning, plans are determined to remedy these causes. 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 [7] 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 through the same process.

2.5 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.

3 An Example Scenario

A prototype of the system has been implemented in the modelling and prototyping environment for the component-based agent design method DESIRE [8]. This prototype has been used to evaluate the model for a specific scenario as specified by domain experts of the Royal Netherlands 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 immediately identify the problem. 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 to 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 mechanic's task package have been specified. For the sake of brevity, these models are not shown, but specified in [9]. Furthermore, a cognitive model concerning the experienced pressure is specified, which is shown in Figure 1. Hereby, the nodes indicate states and the arrows represent causal relationships between these states.

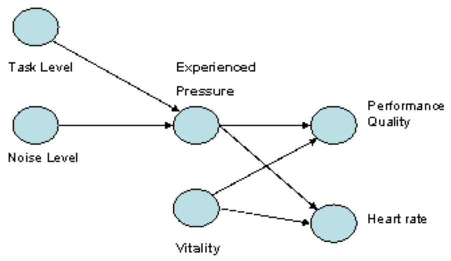

Figure 1: Simplified cognitive model for experienced task pressure

In the agent model, relations between the states have been represented 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 mechanic's task package that comprises two task types maintain_engine and solve_ciws_problem is provided to *Own Process Control* of *PA*. The mechanic 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 mechanic's health is further refined stating that the experienced pressure and the vitality should remain normal:

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

Here, '-1' indicates infinite time. Based on the goals related to the mechanic'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 *PA*:

maintenance of agent information (PA)

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 mechanic:

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, for example:

```
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 mechanic 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 - PA)

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 mechanic 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 - PA)
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 *PA* agent. *Agent Specific Task* derives that the task level should be adjusted:

```
plan determination (AST - PA)
input:   selected_hypothesis(at(and(abnormal_exp_pressure, normal_vitality), 65)
output:  to_be_adjusted(abnormal_task_level)
```

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

4 The Multi-Agent Context for the Personal Assistant Agent

The Personal Assistant agent PA functions within the context of a multi-agent system consisting of different types of agents. In addition to the Personal Assistant itself the following agents are involved; models for all of them were designed based on the component-based Generic Agent Model (GAM) presented in [6]. The *Model Maintenance Agent* (MMA) contains a library of four types of models: monitoring and guidance models, cognitive models, workflow models and dialogue models. Models can be provided to PA upon request; to facilitate this process, each model is annotated with specific parameters. The *State Maintenance Agent* (SMA) maintains characteristics, states and histories of other agents, of the physical world and of the workflows. Information can be requested by the PA's, using a specific element (i.e. agent, physical world, a workflow), an aspect (i.e. state, history) and a time interval for which information should be provided. In addition, the *Mental Operations Agent* (MOA) represents the mental part of the human. MOA is connected to the human's physical body, which can act in the physical worlds. The *Task Execution Support Agent* (TESA) is used by the human as an (active) tool during the execution of a task.

For each human that needs to be supported during the task execution a Personal Assistant agent is created. Initially, the Personal Assistant agent contains generic components only. The configuration of it is performed based on the role that needs to be supported by the agent, on the characteristics of a human who is assigned to this role, and on the goals defined for the Personal Assistant agent.

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 human complies with a workflow model, diagnosis of the human's state may need to be performed. Thus, a query to the model maintenance agent is given which includes the parameter type of analysis with value diagnosis. When a query is specified, the function model_query(query_id, param, list_of_values) is used, 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 choice of cognitive models is guided by the goals that concern internal states of the human. From the goals in Table 1 and their refinements and instantiations, a number of internal states can be identified, among which experienced pressure and heart rate. For such states and for each task the appropriate cognitive, workflow and dialogue models are extracted from the model maintenance agent. By matching 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) communicated to the requestor. The provided models are stored in the Maintenance of Agent Information component of the personal assistant.

More details about the multi-agent context of the personal assistant agent can be found in [10].

5 Conclusions

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 generic agent model for personal support during task execution has been proposed. This agent model allows the use of dynamical 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 [5; 6]. Besides being generic, the proposed 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. To provide the required functionality for personal assistant agents, the multi-agent context in which it functions includes model maintenance and state maintenance agents.

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 [11]. Situation awareness refers to the picture that people have of the environment (e.g., [12]). In case of low situation awareness this picture is wrong, which will often lead to wrong decision making (e.g., [13]). In the literature, it is known that automated systems can also impose a negative effect on cognitive workload or situation awareness [14]. Therefore, systems have been designed that are adaptive, e.g. in only providing aiding when it is necessary [5]. 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. [15], [5]). The personal assistant model 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 human. This allows for an optimal support of the human.

References

1. Posner, M. I., and Boies, S. J. Components of attention. *Psychological Bulletin* 78, 1971, 391-408.
2. Schmorrow, D.D., & Reeves, L.M. *21st century human-system computing: augmented cognition for improved human performance.* Aviat Space Environ Med 2007, 78(5, Suppl.), B7-11.
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. 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.
5. Wilson, G.F., & Russell, C.A. Performance enhancement in an uninhabited air vehicle task using psychophysiologically determined adaptive aiding. *Human Factors, 49(6)*, 2007, 1005-1018.
6. Brazier, F.M.T., Jonker, C.M., and Treur, J. Compositional Design and Reuse of a Generic Agent Model. *Applied AI Journal*, vol. 14, 2000, pp. 491-538.
7. 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.
8. Brazier, F.M.T., Jonker, C.M., and Treur, J., Principles of Component-Based Design of Intelligent Agents. *Data and Knowledge Engineering*, vol. 41, 2002, pp. 1-28.
9. http://www.cs.vu.nl/~wai/PersonalAssistant/Models.pdf
10. Bosse, T., Duell, R., Hoogendoorn, M., Klein, M.C.A., Lambalgen, R. van, Mee, A. van der, Oorburg, R., Sharpanskykh, A., Treur, J., and Vos, M. de. *A Multi-Agent System Architecture for Personal Support During Demanding Tasks*. In: Proc. of the 22nd Int. Conf. on Industrial, Engineering & Other Applications of Applied Intelligent Systems, IEA/AIE'09. Studies in Computational Intelligence, Springer Verlag, 2009, to appear.
11. Wickens, C.D. Situation awareness and workload in aviation. *Current Directions in Psych. Science, 11*, 2002, 128-133.
12. Endsley, M.R. Theoretical underpinnings of situation awareness. In M.R. Endsley & D.J. Garland (Eds.) *Situation awareness analysis and measurement* (pp. 1-21). Mahwah, NJ:Erlbaum, 2000.
13. Endsley, M.R. The role of situation awareness in naturalistic decision making. In C. Zsambok & G. Klein (Eds.), *Naturalistic decision making*: 269-284. Mahwah, NJ: Erlbaum, 1997.
14. Parasuraman, R., & Riley, V. Humans and automation: use, misuse, disuse, abuse. *Human Factors, 39(2)*, 1997, 230-253.
15. Prinzel, L.J., Freeman, F.G., Scerbo, M.W., Mikulka, P.J., Pope, A.T. A closed-loop system for examining psychophysiological measures for adaptive task allocation. *Int. Journal of Aviation Psychology, 10(4)*, 2000, 393-410.