

An Ambient Agent Model for Monitoring and Analysing Dynamics of Complex Human Behaviour

Tibor Bosse^{a*}, Mark Hoogendoorn^a, Michel C.A. Klein^a, and Jan Treur^a

^a *Vrije Universiteit Amsterdam, Department of Artificial Intelligence, de Boelelaan 1081, 1081 HV Amsterdam, The Netherlands*

Abstract. In ambient intelligent systems, monitoring of a human could consist of more complex tasks than merely identifying whether a certain value of a sensor is above a certain threshold. Instead, such tasks may involve monitoring of complex dynamic interactions between human and environment. In order to enable such more complex types of monitoring, this paper presents a generic agent-based framework. The framework consists of support on various levels of system design, namely: (1) the top level, including the interaction between agents, (2) the agent level, providing support on the design of individual agents, and (3) the level of monitoring complex dynamic behaviour, allowing the specification of the aforementioned complex monitoring properties within the agents. The approach is exemplified by a large case study concerning the assessment of driving behaviour, and is applied to two smaller cases as well (concerning fall detection of elderly, and assistance of naval operations, respectively), which are briefly described. These case studies have illustrated that the presented framework enables developers within ambient intelligence to build systems with more expressiveness regarding their monitoring focus. Moreover, they have shown that the framework is easy to use and applicable in a wide variety of domains.

Keywords: ambient agent model, human behaviour, dynamics

* Corresponding author. E-mail: t.bosse@vu.nl.

1. Introduction

Recent developments within Ambient Intelligence provide new technological possibilities to contribute to personal care for safety, health, performance, and wellbeing; cf. [1], [2], [33], [29], [34]. Applications usually acquire sensor information about humans and their functioning over time, and exploit this information in order to decide which type of support should be provided. However, if the underlying mechanisms to make such decisions are limited to, for example, knowledge of the type ‘if the value from a sensor exceeds a certain threshold, then perform a certain intervention’, the ambient support system has limited intelligence. In many application areas the dynamics of the human functioning over time plays an important role. For example, if the human experiences incidentally some stressful moments, this may not be any reason to intervene immediately, but if this repeats itself as a pattern, then it may be important to be noticed and analysed by the ambient system, as this may lead to a negative accumulation. To address such dynamical patterns the ambient system may need to use more detailed models about the dynamics of human functioning, and sophisticated *techniques to analyse* the sensor information over time. In that case more appropriate support can be provided, tailored to a specific course of affairs (cf. [18]). Based on this, indeed ambient devices can (re)act by undertaking actions in a knowledgeable manner that improve the human’s, safety, health, performance, and wellbeing.

The current paper presents a generic approach to develop such more knowledgeable ambient devices. The approach consists of a number of steps for the development of such devices. The first step concerns identifying the agents that play a role in the system being developed (cf. [9]). In addition, standard ontology, and reasoning and communication patterns are presented to enable interaction between these agents. In the second step of the approach, designs for the individual agents are addressed by means of a generic agent model in which certain standard components are specified (cf. [9]). Here, standard ontology for the internals of the agent and more detailed specification of the reasoning patterns are presented. The main contribution of the paper is the third step. In this step, the issue of specifying complex dynamic properties (the techniques to analyse human behaviour) is addressed. Hereby, complex patterns over time can be monitored, and support methods are triggered upon identification of a problem.

To illustrate the approach, the paper uses a system to monitor driving behaviour as a main case study; see also [11], [17], [25], [26], [30], [31], [39]. In this domain, circumstances may occur in which a person’s internal state is affected in such a way that driving is no longer safe. For example, when a person has taken drugs, either prescribed by a medical professional, or by own initiative, the driving behaviour may be impaired. For the case of alcohol, specific tests are possible to estimate the alcohol level in the blood, but for many other drugs such tests are not available. Moreover, a bad driver state may have other causes, such as highly emotional events, or being sleepy. Therefore assessment of the driver’s state by monitoring the driving behaviour itself and analysing the monitoring results is a case study which represents a broader case of monitoring applications. A component-based ambient agent-based model is presented (based upon the generic approach) to assess a person’s driving behaviour, and in case of a negative assessment to let cruise control slow down and stop the car. The approach was inspired by a system that is currently under development by Toyota. This ambient system that in the near future will be incorporated as a safety support system in Toyota’s prestigious Lexus line, uses sensors that can detect the driver’s steering operations, and sensors that can detect the focus of the driver’s gaze. The generic approach is taken as a basis to design the system. Simulation runs have been performed using the model, and dynamic properties of components at different aggregation levels as well as interlevel relations between them have been formally specified and verified against these traces. Besides the main case study, two other cases were investigated as well, namely monitoring of elderly falling, and monitoring of attention of naval operators.

The paper is organised as follows. First, the modelling approach is introduced in Section 2. In Section 3 the global structure of the agent-based model is introduced, whereas Section 4 presents a generic ambient agent model. Specialisations of this generic agent model for the specific agents within the ambient driver assessment system are introduced in Section 5, and in Section 6 simulation results using the entire model are described. Section 7 shows the results of verification of properties against the simulation traces. In Section 8 more case studies are addressed, and finally Section 9 is a discussion.

2. Modelling Approach

This section briefly introduces the modelling approach used. To specify the model conceptually and formally, the agent-oriented perspective is a suitable choice, since it enables the modeller to treat both humans and ambient devices as intelligent, autonomous entities (agents), and to describe their behaviour at a conceptual and formal level in terms of intuitive concepts such as observations, actions, communications, and beliefs. However, to model the dynamic aspects of systems involved in real world processes such as human driving behaviour from an agent perspective, an expressive modelling language is needed. On the one hand, qualitative aspects have to be addressed, such as observations that the engine is running, positions of the steering wheel, and decisions to slow down the car. On the other hand, quantitative aspects have to be addressed. For example, the focus of the driver's gaze can best be described by real numbers. Another requirement of the chosen modelling language is its suitability to express on the one hand the basic mechanisms of a process (for the purpose of simulation), and on the other hand more global properties of the process (for the purpose of logical analysis and verification). For example, basic mechanisms of a driver support system involve patterns to assess dangerous situations, or to determine various support measures, whereas examples of global properties are statements like "whenever the driver exposes symptoms of impaired behaviour, then within 30 seconds the car will slow down".

The Temporal Trace Language TTL for formal specification and verification of dynamic properties [4], [22] fulfills all of these desiderata. This hybrid temporal predicate logical language supports formal specification and analysis of dynamic properties, covering a combination of qualitative and quantitative aspects. This combination allows the modeller to exploit both logical and numerical methods for analysis and simulation. Moreover it can be used to express dynamic properties at different levels of aggregation, which makes it well suited both for simulation and logical analysis. The TTL language is

based on the assumption that dynamics can be described as an evolution of *states* over time. The notion of state as used here is characterised on the basis of an ontology defining a set of physical and/or mental (state) properties that do or do not hold at a certain point in time. A specific state is characterised by dividing the set of state properties into those that hold, and those that do not hold in the state. Examples of state properties are 'agent A observes that the engine is running', or 'the human's gaze is focussed at the location with coordinates $\{0.34, 0.85\}$ '. Real value assignments to variables are also considered as possible state property descriptions.

To describe dynamic properties of complex processes such as human driving behaviour, explicit reference is made to *time* and to *traces*. Thus, TTL is built on atoms referring to states, time points and traces. Formally, a *state* of a process for (state) ontology Ont is an assignment of truth values to the set of ground atoms in the ontology. The set of all possible states for ontology Ont is denoted by $\text{STATES}(\text{Ont})$. To describe sequences of states, a fixed *time frame* T is assumed which is linearly ordered. A *trace* γ over state ontology Ont and time frame T is a mapping $\gamma : T \rightarrow \text{STATES}(\text{Ont})$, i.e., a sequence of states γ_t ($t \in T$) in $\text{STATES}(\text{Ont})$. The set of *dynamic properties* $\text{DYN-PROP}(\text{Ont})$ is the set of 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 \models , comparable to the Holds-predicate in the Situation Calculus [32]: $\text{state}(\gamma, t) \models p$ denotes that state property p holds in trace γ at time t .

Based on these statements, dynamic properties can be formulated in a sorted first-order predicate logic, using quantifiers over time and traces and the usual first-order logical connectives such as $\neg, \wedge, \vee, \Rightarrow, \forall, \exists$. A special software environment has been developed for TTL, featuring a Property Editor for building TTL properties and a Checking Tool that enables formal verification of such properties against a set of traces.

1. *Equality of traces*

$$\forall \gamma 1, \gamma 2 \ [\forall t \ [\text{state}(\gamma 1, t) = \text{state}(\gamma 2, t)] \Rightarrow \gamma 1 = \gamma 2]$$
2. *Equality of states*

$$\forall s 1, s 2 \ [\forall a : \text{STATPROP} \ [\text{truthvalue}(s 1, a) = \text{truthvalue}(s 2, a)] \Rightarrow s 1 = s 2]$$
3. *Truth value in a state*

$$\text{holds}(s, p) \Leftrightarrow \text{truthvalue}(s, p) = \text{true}$$
4. *State consistency axiom*

$$\forall \gamma, t, p \ [\text{holds}(\text{state}(\gamma, t), p) \Rightarrow \neg \text{holds}(\text{state}(\gamma, t), \text{not}(p))]$$
5. *State property semantics*
 - a. $\text{holds}(s, (p 1 \wedge p 2)) \Leftrightarrow \text{holds}(s, p 1) \ \& \ \text{holds}(s, p 2)$
 - b. $\text{holds}(s, (p 1 \vee p 2)) \Leftrightarrow \text{holds}(s, p 1) \ | \ \text{holds}(s, p 2)$
 - c. $\text{holds}(s, \text{not}(p 1)) \Leftrightarrow \neg \text{hold}(s, p 1)$
6. *For any constant variable name x from the sort SVARS:*

$$\text{holds}(s, (\exists x, F)) \Leftrightarrow \exists x' : \text{SGTERMS} \ \text{holds}(s, G)$$

$$\text{holds}(s, (\forall x, F)) \Leftrightarrow \forall x' : \text{SGTERMS} \ \text{holds}(s, G)$$

with G, F terms of sort STATPROP, where G is obtained from F by substituting all occurrences of x by x'.
7. *Partial order axioms for the sort TIME*
 - a. $\forall t \ t \leq t$ (Reflexivity)
 - b. $\forall t 1, t 2 \ [t 1 \leq t 2 \wedge t 2 \leq t 1] \Rightarrow t 1 = t 2$ (Anti-Symmetry)
 - c. $\forall t 1, t 2, t 3 \ [t 1 \leq t 2 \wedge t 2 \leq t 3] \Rightarrow t 1 \leq t 3$ (Transitivity)
8. *Axioms for the sort VALUE: the same as for the sort TIME and standard arithmetic axioms.*
9. *Axioms which relate the sorts TIME and VALUE*
 - a. $(t + v 1) + v 2 = t + (v 1 + v 2)$
 - b. $(t \cdot v 1) \cdot v 2 = t \cdot (v 1 \cdot v 2)$
10. *Finite variability property (for any trace γ).*
This property ensures that a trace is divided into intervals such that the overall state is stable within each interval, i.e., each state property changes its truth value at most a finite number of times:

$$\forall t 0, t 1 \ [t 0 < t 1 \Rightarrow \exists \delta > 0 \ [\forall t \ [t 0 \leq t \ \& \ t \leq t 1] \Rightarrow$$

$$\exists t 2 \ [t 2 \leq t \ \& \ t < t 2 + \delta \ \& \ \forall t 3 \ [t 2 \leq t 3 \ \& \ t 3 \leq t 2 + \delta] \Rightarrow \text{state}(\gamma, t 3) = \text{state}(\gamma, t)]]$$

Box 1 Specific aspects of semantics of TTL

As TTL uses order-sorted predicate logic as a point of departure, it inherits the standard semantics of this variant of predicate logic. That is, the semantics of TTL is defined in a standard way, by interpretation of sorts, constants, functions and predicates, and a variable assignment. However, in addition the semantics involves some specialised aspects. As a number of standard sorts are present, the elements of these sorts are limited to instances of specified terms in these sorts, as is usual, for example, in logic programming semantics. For example, for the sort TIME it is assumed that in its semantics its elements consist of the time points of the fixed time frame chosen. Moreover, for the sort TRACE, it is assumed that in its semantics its elements consist of a (limited) number of elements named by constants. Furthermore, for the

sort STATPROP for state properties it is assumed that in its semantics its elements consist of the set of terms denoting the propositions built in a chosen state language (this is called reification). In Box 1 an overview is shown of the special elements added to the semantics inherited from order-sorted predicated logic (see also [36]).

A full description of all technical details of TTL's semantics is beyond the scope of the current paper. For this purpose, see [36].

To specify executable temporal models and to use these models for simulation and analysis, the language LEADSTO, an executable sublanguage of TTL, is used (cf. [3]). The basic building blocks of this language are direct temporal (e.g., causal) relations of the format $\alpha \rightarrow_{e, f, g, h} \beta$, which means:

if state property α holds for a certain time interval with duration g ,
 then after some delay (between e and f) state property β will hold for a certain time interval of length h .

where α and β are state properties of the form ‘conjunction of literals’ (where a literal is an atom or the negation of an atom), and e, f, g, h non-negative real numbers. Note that to specify internal agent processes, such direct temporal relations can be (and often are) used on different types of representations, for example, also to perform reflective reasoning in different directions over time. As a very simple example of abductive reasoning

$\text{observes}(A, Y) \ \& \ \text{believes}(A, \text{causes}(X, Y)) \ \rightarrow \text{believes}(A, X)$
 expresses that when at some point in time agent A observes Y and believes that X causes Y , then at some next point in time A will believe X .

3. Global Structure

For the global structure of the model, first a distinction is made between those components that are the *subject* of the system (e.g., a patient to be taken care of), and those that are *ambient*, supporting components. Moreover, from an agent-based perspective (see, e.g., [9], [10]), a distinction is made between active, *agent* components (human or artificial), and passive, *world* components (e.g., part of the physical world or a database). Agents may interact through

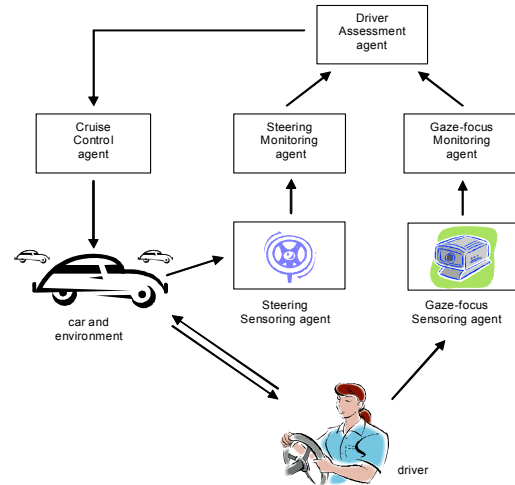


Fig. 1. Ambient Driver Support System

communication. Interaction between an agent and a world component can be either *observation* or *action* performance; cf. [9]. An action is generated by an agent, and transfers to a world component to have its effect there. An *observation result* is generated by a world component and transferred to the agent.

The interaction between components of the global structure of an ambient system can be illustrated by the case study about a system for monitoring driving behaviour. In Figure 1 an overview of this system is shown. Table 1 shows the structure in terms of different types of components and interactions.

Table 1. Components and Interactions of the Ambient Driver Support System

subject components	subject agent	subject world component
	human driver	car and environment
subject interactions	observation and action by subject agent in subject world component	
	driver observes car and environment driver operates car and gaze	
ambient components	ambient agents	
	steering and gaze-focus sensing agent; steering and gaze-focus monitoring agent; driver assessment agent, cruise control agent	
ambient interactions	communication between ambient agents	
	steering sensing agent communicates to steering monitoring agent gaze-focus sensing agent communicates gaze focus to gaze-focus monitoring agent eye-focus monitoring agent reports to driver assessment agent unfocused gaze steering monitoring agent reports to driver assessment agent abnormal steering driver assessment agent communicates to cruise control agent state of driver	
interactions between subject and ambient components	observation and action by ambient agent in subject world component	
	steering sensing agent observes steering wheel gaze-focus sensing agent observes driver gaze cruise control agent slows down or stops engine	

Table 2. Ontology for Interaction at the Global Level

SORT	Description
ACTION	an action
AGENT	an agent
INFO_EL	an information element, possibly complex (e.g., a conjunction of other info elements)
WORLD	a world component
Predicate	
	Description
performing_in(A:ACTION, W:WORLD)	action A is performed in W
observation_result_from(I:INFO_EL, W:WORLD)	observation result from W is I
observed_result_from(I:INFO_EL, W:WORLD)	observation result from W was I
communication_from_to(I:INFO_EL, X:AGENT, Y:AGENT)	information I is communicated by X to Y
communicated_from_to(I:INFO_EL, X:AGENT, Y:AGENT)	information I was communicated by X to Y

3.1. State Ontologies Used at the Global Level

For the information exchanged between components at the global level, ontologies have been specified. This has been done in a universal order-sorted predicate logic format that easily can be translated into more specific ontology languages. Table 2 provides an overview of sorts and predicates used in interactions at the global level.

3.2. Temporal Relations at the Global Level

Interaction between global level components is defined by the following specifications. These general patterns facilitate the propagation of an action, observation or communication between agents or world components. In such specifications, the prefix input, output or internal is used for state properties. This is an indexing of the language elements to indicate that it concerns specific variants of them either present at the input, output or internally within the agent, whereas for the state properties themselves the same representations can be used (without introducing overloading). The indexation is required to distinguish between the role of the state properties in the reasoning process. In the specification below, the symbol \rightarrow is used as shorthand notation for $\rightarrow_{0, 0, 1, 1}$.

Action Propagation from Agent to World Component

$\forall X:AGENT \forall W:WORLD \forall A:ACTION$ output(X)|performing_in(A, W)
 \rightarrow input(W)|performing_in(A, W)

Observation Result Propagation from World to Agent

$\forall X:AGENT \forall W:WORLD \forall I:INFO_EL$
output(W)|observation_result_from(I, W)
 \rightarrow input(X)|observed_result_from(I, W)

Communication Propagation Between Agents

$\forall X, Y:AGENT \forall I:INFO_EL$
output(X)|communication_from_to(I, X, Y) \rightarrow
input(Y)|communicated_from_to(I, X, Y)

4. Component-Based Ambient Agent Model

After the discussion of the global structure of ambient systems, this section focuses on the general Ambient Agent Model (AAM) used for the ambient agents within the system. These agents are assumed to maintain knowledge about certain aspects of human functioning, and information about the current state and history of the world and other agents. Based on this knowledge they are able to have some understanding of the human processes, and can behave accordingly. In Section 5 it is shown how the Ambient Agent Model AAM has been specialised to obtain models for the four types of ambient agents in the system, i.e. the steering monitor agent, the gaze monitor agent, the driver assessment agent, and the cruise control agent.

4.1. Components within the Ambient Agent Model

Based on the component-based Generic Agent Model (GAM) presented in [9], a model for ambient agents (AAM) was designed. Within AAM, as in GAM 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 processes involved in controlling the agent (e.g., determining, monitoring and evaluating its own goals and plans) but also the processes of maintaining a self model are the task of the component *Own Process Control*. In the component *Agent Specific Task*, specific tasks of the agents can be modelled. A graphical overview of this com-

ponent-based agent model GAM (taken from [9]) is shown in Figure 2. In this figure, the rounded rectangles denote components, the smaller rectangles on both sides of the components denote input and output interfaces, and the arrows denote information links.

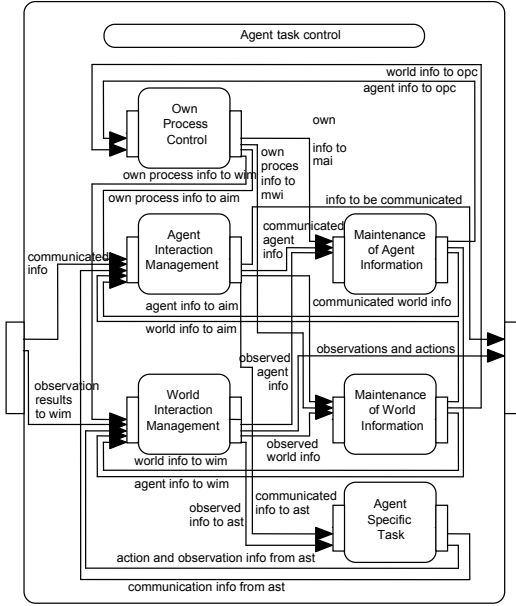


Fig. 2. Overview of the component-based agent model GAM (taken from [9]).

Adopting this component-based agent model GAM, the *ambient agent model* (AAM) can be obtained as a refinement in the following manner (not shown in Figure 2). The component *Maintenance of Agent Information* has three subcomponents in AAM. The subcomponent *Maintenance of a Dynamic Agent Model* maintains the causal and temporal relationships for the subject agent's functioning. The subcomponent *Maintenance of an Agent State Model* maintains a snapshot of the (current) state of the agent. As an example, this may model the gaze focussing state. The subcomponent *Maintenance of an Agent History Model* maintains the history of the (current) state of the agent. This may for instance model gaze patterns over time.

Similarly, the component *Maintenance of World Information* has three subcomponents for a dynamic world model, a world state model, and a world history model, respectively. Moreover, the component *Agent Specific Task* has the following three subcomponents: *Simulation Execution* extends the informa-

tion in the agent state model based on the internally represented dynamic agent model for the subject agent's functioning, *Process Analysis* assesses the current state of the agent, and *Plan Determination* determines whether action has to be undertaken, and, if so, which ones (e.g., to determine that the cruise control agent has to be informed).

Finally, as in the model GAM, 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.

4.2. State Ontologies within Agent and World

To express the information involved in the agent's internal processes, the ontology shown in Table 3 was specified.

Table 3. Ontology used within the Ambient Agent Model

Ontology element	Description
belief(I:INFO_EL)	information I is believed
world_fact(I:INFO_EL)	I is fact true in the world
has_effect(A:ACTION, I:INFO_EL)	action A has effect I
leads_to_after(I:INFO_EL, J:INFO_EL, D:REAL)	state property I leads to state property J after D
at(I:INFO_EL, T:TIME)	property I holds at time T

As an example $\text{belief}(\text{leads_to_after}(I:\text{INFO_EL}, J:\text{INFO_EL}, D:\text{REAL}))$ is an expression based on this ontology which represents that the agent has the knowledge that state property I leads to state property J with a certain time delay specified by D.

4.3. Generic Temporal Relations within AAM

Similar to the global structure, a number of generic temporal relations are needed for the functioning of the Ambient Agent Model.

Belief Generation based on Observation and Communication

$$\forall X:\text{AGENT}, I:\text{INFO_EL}, W:\text{WORLD}$$

$$\text{input}(X)|\text{observed_result_from}(I, W) \wedge \text{internal}(X)|\text{belief}(\text{is_reliable_for}(W, I)) \rightarrow \text{internal}(X)|\text{belief}(I)$$

$$\forall X, Y:\text{AGENT}, I:\text{INFO_EL}$$

$$\text{input}(X)|\text{communicated_from_to}(I, Y, X) \wedge \text{internal}(X)|\text{belief}(\text{is_reliable_for}(X, I)) \rightarrow \text{internal}(X)|\text{belief}(I)$$

Here, the first specification is a generic pattern for the component *World Interaction Management*, and the second for the component *Agent Interaction Management*. Explicit mechanisms to derive the truth values of different *is_reliable_for* relations (i.e., to determine which sources are reliable for which type of information) are beyond the scope of this paper. However, one straightforward way to do this would

be to use an experience-based trust mechanism, e.g., as put forward in [23]. When the sources are assumed always reliable, the conditions on reliability can be left out.

Belief Generation based on Simulation

$\forall X:\text{AGENT} \forall I,J:\text{INFO_EL} \forall D:\text{REAL} \forall T:\text{TIME}$
 $\text{internal}(X)|\text{belief}(\text{at}(I, T)) \wedge \text{internal}(X)|\text{belief}(\text{leads_to_after}(I, J, D))$
 $\rightarrow \text{internal}(X)|\text{belief}(\text{at}(J, T+D))$

The last generic pattern within the agent’s component *Simulation Execution* specifies how a dynamic model that is explicitly represented as part of the agent’s knowledge (within its component *Maintenance of Dynamic Models*) can be used to perform simulation, thus extending the agent’s beliefs about the world state at different points in time. This can be considered an internally represented deductive causal reasoning method. Another option is a multiple effect abductive causal reasoning method:

Belief Generation based on Multiple Effect Abduction

$\forall X:\text{AGENT} \forall I,J1, J2:\text{INFO_EL} \forall D:\text{REAL} \forall T:\text{TIME}$
 $J1 \neq J2 \wedge \text{internal}(X)|\text{belief}(\text{at}(J1, T)) \wedge$
 $\text{internal}(X)|\text{belief}(\text{leads_to_after}(I, J1, D)) \wedge$
 $\text{internal}(X)|\text{belief}(\text{at}(J2, T)) \wedge$
 $\text{internal}(X)|\text{belief}(\text{leads_to_after}(I, J2, D)) \rightarrow$
 $\text{internal}(X)|\text{belief}(\text{at}(I, T-D))$

Note that the reliability of this last pattern is dependent on the number of other potential causes of J1 and J2, which is domain-specific. For the general case, a variant of the pattern could be established, incorporating a comparison between the observed and maximally possible number of causes and effects, as well as a confidence threshold to decide whether or not to apply the abduction.

In [35], another formal approach (based on event calculus) is presented to draw conclusions about humans’ states (in particular intentions) on the basis of observed behaviour and background knowledge. This approach can be seen as an elaboration of some of the specifications presented in this section.

4.4. Generic Temporal Relations within a World

For World Components the following specifications indicate the actions’ effects and how observations provide their results. The first specification defines that actions carried out in the world will lead to a changed state of the world when there is a pattern that specifies an effect of the action. The *observation_focus_in* predicate in the second and third specification is used to be able to do specific observations of one aspect of a world component. For example, in the running case study, it is used to specify that the steering wheel of the car will be observed. This prevents

that all facts within a world component are outputted as observation result.

Action Execution and Observation Result Generation in a World

$\forall W:\text{WORLD_COMP} \forall A:\text{ACTION} \forall I:\text{INFO_EL}$
 $\text{input}(W)|\text{performing_in}(A, W) \wedge \text{internal}(W)|\text{has_effect}(A,I)$
 $\rightarrow \text{internal}(W)|\text{world_fact}(I)$
 $\forall W:\text{WORLD_COMP} \forall I:\text{INFO_EL}$
 $\text{input}(W)|\text{observation_focus_in}(I, W) \wedge \text{internal}(W)|\text{world_fact}(I) \rightarrow$
 $\text{output}(W)|\text{observation_result_from}(I, W)$
 $\forall W:\text{WORLD_COMP} \forall I:\text{INFO_EL}$
 $\text{input}(W)|\text{observation_focus_in}(I, W) \wedge \text{internal}(W)|\text{world_fact}(\text{not}(I))$
 $\rightarrow \text{output}(W)|\text{observation_result_from}(\text{not}(I), W)$

5. Instantiations of the Ambient Agent Model

This section provides instantiations of the Ambient Agent Model for, respectively, Ambient Monitor Agents, a Driver Assessment Agent, and a Cruise Control Agent.

5.1. Ambient Monitor Agents

As a refinement of the Ambient Agent Model AAM, an Ambient Monitoring Agent Model AMAM has been designed, and instantiated for steering monitoring and gaze monitoring. The role of the two monitoring agents is to detect via the gaze of the driver or the steering behaviour that the driver might be impaired. These agents relate temporal patterns of gaze, resp. steering to qualifications of abnormality. Note that the qualifications of abnormality are specific to driving behaviour; for example, a typical temporal pattern of gaze that is classified as abnormal is a situation in which the driver stares in the distance for a long period. A typical abnormal steering pattern is a situation in which the steer continuously moves from left to right (and back) during a certain period.

Table 4 indicates the components within these monitoring agents.

Table 4. Ambient Monitor Agent Model: Components

Maintenance of Agent and World Information	
maintenance of history models	model of gaze/steering patterns over time
Agent Specific Task	
process analysis	determine whether a gaze/steering pattern is abnormal
plan determination	for abnormality state decide to communicate to driver assessment agent
Agent Interaction Management	prepare communication to driver assessment agent

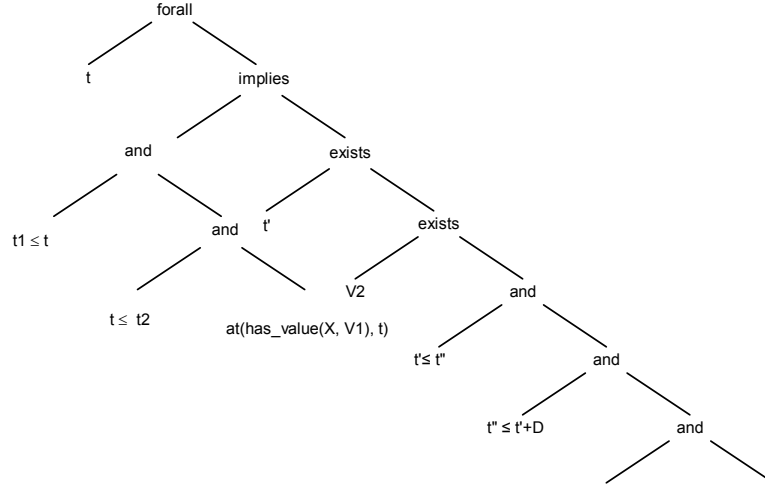


Fig. 3. Nested property structure depicted as a tree.

A monitor agent receives a stream of information over time, obtained by observation of a world component or by communication from other agents. Typical sources of information are world parts equipped with sensor systems or sensing agents that interact with such world parts. Any monitoring agent has some properties of the incoming information stream that are to be monitored (*monitoring foci*), e.g., concerning the value of a variable, or a temporal pattern to be detected in the stream. This idea of checking certain properties of an agent system at runtime has similarities to the approach proposed in [14]. As output a monitoring agent generates communication that a certain monitoring focus holds. Table 5 explains the predicates used for the specification of the monitoring process.

Table 5. Predicates used in the monitoring process

Ontology element	Description
monitor_focus (P:PROP)	property P is a focus of the monitoring process
has_expression (E:EXP, P:PROP)	expression E is a named abstraction of property P
in_focus (P:PROP)	property P is a focus of the verification process
verification (P:PROP, pos)	property P is positively verified

A monitor focus can be a state property or a dynamic property. An example of a simple type of state property to be used as a monitor focus is a state property that expresses that the value of a certain variable X is between two bounds LB and UB: $\exists V [\text{has_value}(X, V) \wedge LB \leq V \wedge V \leq UB]$. In prefix notation, this can be expressed as follows: $\text{exists}(V, \text{and}(\text{has_value}(X, V),$

$\text{and}(LB \leq V, V \leq UB)))$. It is possible to obtain abstraction by using (meaningful) names of properties. For example, $\text{stable_within}(X, LB, UB)$ can be used as an abstract name for the example property expressed above by specifying:

$\text{has_expression}(\text{stable_within}(X, LB, UB), \text{exists}(V, \text{and}(\text{has_value}(X, V), \text{and}(LB \leq V, V \leq UB))))$

The fact that a property $\text{stable_within}(X, LB, UB)$ is a monitor focus for the monitor agent is expressed by: $\text{monitor_focus}(\text{stable_within}(X, LB, UB))$. An example of a monitor property is:

$\forall t [t1 \leq t \wedge t \leq t2 \wedge \text{at}(\text{has_value}(X, V1), t) \rightarrow \exists t', V2 [t \leq t' \leq t+D \wedge V2 \neq V1 \wedge \text{at}(\text{has_value}(X, V2), t')]]$

This property expresses that between $t1$ and $t2$ the value of variable X is changing all the time, which can be considered as a type of instability of that variable. This dynamic property is expressed in prefix notation as:

$\text{forall}(t, \text{implies}(\text{and}(t1 \leq t, \text{and}(t \leq t2, \text{at}(\text{has_value}(X, V1), t))), \text{exists}(t', \text{exists}(V2, \text{and}(t \leq t', \text{and}(t' \leq t+D, \text{and}(V2 \neq V1, \text{at}(\text{has_value}(X, V2), t'))))))$

This expression can be named, for example, by $\text{instable_within_duration}(X, D)$. It is assumed that the monitor focus on which output is expected is an input for the agent, communicated by another agent. This input is represented in the following manner.

$\text{communicated_from_to}(\text{monitor_focus}(F), A, B)$
 $\text{communicated_from_to}(\text{has_expression}(F, E), A, B)$

Note that it is assumed here that the ontology elements used in the expression E here are elements of the ontology used for the incoming stream of information. Moreover, note that for the sake of simplicity, sometimes a prefix such as $\text{input}(X)$, which indi-

cates in which agent a state property occurs, is left out.

Within AMAM's World Interaction Management component, observation results get a time label: $observed_result_in(I, W) \wedge current_time(T) \rightarrow belief(at(I, T))$. Similarly, within the Agent Interaction Management component communicated information is labeled: $communicated_from_to(I, X, AMAM) \wedge current_time(T) \rightarrow belief(at(I, T))$. The time-labeled consequent atoms $belief(at(I, T))$ are transferred to the component Maintenance of Agent History and stored there.

Within the component Process Analysis two specific subcomponents are used: Monitoring Foci Determination, and Monitor Foci Verification.

5.1.1. Monitoring Foci Determination

In this component the monitor agent's monitoring foci are determined and maintained: properties that are the focus of the agent's monitoring task. The overall monitoring foci are received by communication and stored in this component. However, to support the monitoring process, it is useful when an overall monitor focus is decomposed into more refined foci: its constituents are determined (the subformulas) in a top-down manner, following the nested structure as depicted in Figure 3.

This decomposition process was specified in the following manner:

$$\begin{aligned} monitor_focus(F) &\rightarrow in_focus(F) \\ in_focus(E) \wedge is_composed_of(E, C, E1, E2) &\rightarrow \\ &in_focus(E1) \wedge in_focus(E2) \end{aligned}$$

Here $is_composed_of(E, C, E1, E2)$ indicates that E is an expression obtained from subexpressions $E1$ and $E2$ by a logical operator C (i.e., and, or, implies, not, forall, exists).

5.1.2. Monitoring Foci Determination

The process to verify whether a monitoring focus holds, makes use of the time-labeled beliefs that are maintained. If the monitoring focus is an atomic property $at(I, T)$ of the state of the agent and/or world at some time point, beliefs about these state properties are involved in the verification process:

$$in_focus(E) \wedge belief(E) \rightarrow verification(E, pos)$$

Verification of more complex formulae is done by combining the verification results of the subformulae following the nested structure as shown in Figure 3 in a bottom-up manner:

$$\begin{aligned} in_focus(and(E1, E2)) \wedge verification(E1, pos) \wedge verification(E2, pos) &\rightarrow \\ \rightarrow verification(and(E1, E2), pos) & \\ in_focus(or(E1, E2)) \wedge verification(E1, pos) \rightarrow & \\ verification(or(E1, E2), pos) & \\ in_focus(or(E1, E2)) \wedge verification(E2, pos) \rightarrow & \end{aligned}$$

$$\begin{aligned} verification(or(E1, E2), pos) & \\ in_focus(implies(E1, E2)) \wedge verification(E2, pos) \rightarrow & \\ verification(implies(E1, E2), pos) & \\ in_focus(implies(E1, E2)) \wedge not\ verification(E1, pos) \rightarrow & \\ verification(implies(E1, E2), pos) & \\ in_focus(not(E)) \wedge not\ verification(E, pos) \rightarrow & \\ verification(not(E), pos) & \\ in_focus(exists(V, E)) \wedge verification(E, pos) \rightarrow & \\ verification(exists(V, E), pos) & \\ in_focus(forall(V, E)) \wedge not\ verification(exists(V, not(E), pos)) \rightarrow & \\ verification(forall(V, E), pos) & \end{aligned}$$

The negative outcomes $not\ verification(E, pos)$ of verification can be obtained by a Closed World Assumption on the $verification(E, pos)$ atoms. If needed, from these negations, explicit negative verification outcomes can be derived:

$$not\ verification(E, pos) \rightarrow verification(E, neg)$$

The following relates verification of an expression to its name:

$$verification(E, S) \wedge has_expression(F, E) \rightarrow verification(F, S)$$

Eventually, when a monitoring property E has been satisfied that is an indication for a certain type of abnormal behaviour of the driver, the Monitoring agent will indeed believe this; for example, for the Steering Monitoring Agent:

$$\begin{aligned} verification(E, pos) \wedge & \\ internal(monitoring_agent) \wedge belief(is_indication_for(E, I)) & \\ \rightarrow internal(monitoring_agent) \wedge belief(I) & \end{aligned}$$

5.2. Driver Assessment Agent

As another refinement of the Ambient Agent Model AAM, the Driver Assessment Agent Model DAAM; see Table 6 for an overview of the different components. For the Driver Assessment Agent, a number of domain-specific patterns have been identified in addition to the generic patterns specified for the Ambient Agent Model presented in Section 4. Some of the key patterns are expressed below. First of all, within the Driver Assessment Agent an explicit representation is present of a dynamic model of the driver's functioning. In this model it is represented how an impaired state has behavioural consequences: abnormal steering operation and gaze focusing.

Table 6. Driver Assessment Agent Model: Components

Maintenance of Agent and World Information	
maintenance of dynamic models	model relating impaired state to abnormal steering behaviour and gaze focusing
maintenance of state models	model of internal state, abnormality of gaze of driver, and of steering wheel
Agent Specific Task	
process analysis	determine impaired driver state by multiple effect abduction
plan determination	for impaired driver state decide to communicate negative assessment to cruise control agent
Agent Interaction Management	
	receive and prepare communication (from monitor agents, to cruise control agent)

The dynamic model is represented in component *Maintenance of Dynamic Models* by:

```

inter-
nal(driver_assessment_agent)|belief(leads_to_after(impaired_state,
abnormal_steering_operation, D))
inter-
nal(driver_assessment_agent)|belief(leads_to_after(impaired_state,
unfocused_gaze, D))

```

The Driver Assessment Agent receives information about abnormality of steering and gaze from the two monitoring agents. When relevant, by the multiple effect abductive reasoning method specified by the generic temporal pattern in Section 4, the Driver Assessment Agent derives a belief that the driver has an impaired internal state. This is stored as a belief in the component *Maintenance of an Agent State Model*. Next, it is communicated to the Cruise Control Agent that the driver assessment is negative.

5.3. Cruise Control Agent

The Cruise Control Agent Model CCAM is another agent model obtained by specialisation of the Ambient Agent Model AAM. It takes the appropriate measures, whenever needed. Within its Plan Determination component, the first temporal specification expresses that if it believes that the driver assessment is negative, and the car is not driving, then the ignition of the car is blocked:

```

internal(cruise_control_agent)|belief(driver_assessment(negative)) ^
internal(cruise_control_agent)|belief(car_is_not_driving)
→
output(cruise_control_agent)|performing_in(block_ignition,
car_and_environment)

```

If the car is already driving, the car is slowed down:

```

internal(cruise_control_agent)|belief(driver_assessment(negative)) ^
internal(cruise_control_agent)|belief(car_is_driving)
→
output(cruise_control_agent)|performing_in(slow_down_car,
car_and_environment)

```

6. Simulation Results

Based upon temporal specifications as described in the previous section, an overall specification within the LEADSTO software environment (cf. [3]) has been made and simulation runs of the system have been generated, of which an example trace is shown in Figure 4. In the figure, the left side indicates the atoms that occur during the simulation whereas the right side indicates a time line where a dark box indicates the atom is true at that time point and a grey box indicates false. Note that in the trace merely the outputs and internal states of the various components are shown for the sake of clarity.

The driver starts the car and accelerates, resulting in a driving car.

```

internal(car_and_environment)|world_fact(car_driving)

```

After a short time, between time points 10 and 20, the driver shows signs of inadequate behaviour: the gaze becomes unfocused and steering instable. Over short time intervals an alternation occurs of:

```

output(driver)|performing_in(steer_position(centre),
car_and_environment)
output(driver)|performing_in(steer_position(left),
car_and_environment)
output(driver)|performing_in(steer_position(right),
car_and_environment)

```

On the other hand, the gaze focus becomes fixed for long time intervals:

```

output(driver)|observation_result_from(gaze_focus(far_away),
driver)

```

The temporal sequences of these observed steering positions and gaze focus are communicated moment by moment by the respective sensing agent to the corresponding monitoring agent. The following dynamic monitor property is used as monitor focus within the Steering Monitoring Agent:

$$\forall t [t1 \leq t \wedge t \leq t2 \wedge \text{belief}(\text{at}(\text{steer_position}(\text{centre}), t)) \rightarrow \exists t' t \leq t' \leq t+D \wedge \text{not belief}(\text{at}(\text{steer_position}(\text{centre}), t'))]$$

This property expresses that between $t1$ and $t2$, whenever the steer is in a central position, there is a slightly later time point at which it is not in a central position (in other words, the driver keeps on moving the steer). This dynamic property is expressed in prefix notation as:

```

forall(t, implies(and(t1 ≤ t, and(t ≤ t2,
belief(at(steer_position(centre), t))))), exists(t', and(t ≤ t',
and(t' ≤ t+D, not(belief(at(steer_position(centre), t')))))

```



Fig. 4. Example Simulation Trace

In LEADSTO this property was expressed as:

```
is_composed_of(gp(1), forall, t, gp(2, t))
is_composed_of(gp(2, t), implies, gp(3, t), gp(8, t))
is_composed_of(gp(3, t), and, gp(4, t), gp(5, t))
has_expression(gp(4, t), t1≤t)
is_composed_of(gp(5, t), and, gp(6, t), gp(7, t))
has_expression(gp(6, t), t≤t2)
has_expression(gp(7, t),
  belief(at(steer_position(centre), t)))
is_composed_of(gp(8, t), exists, t', gp(9, t, t'))
is_composed_of(gp(9, t, t'), and,
  gp(10, t, t'), gp(11, t, t'))
has_expression(gp(10, t, t'), t≤t')
is_composed_of(gp(11, t, t'), and,
  gp(12, t, t'), gp(13, t, t'))
has_expression(gp(12, t, t'), t'≤sum(t, D))
is_composed_of(gp(13, t, t'), not,
  gp(14, t, t'), gp(14, t, t'))
has_expression(gp(14, t, t'),
  belief(at(steer_position(centre), t')))
```

Note that during the process within the Steering Monitoring Agent the overall monitoring focus given by this dynamic property is decomposed into a number of smaller expressions (using the predicate `is_composed_of`). The top level expression (that is checked by the Steering Monitoring Agent) is called `gp(1)`. The atomic expressions have the form of a belief that a state property holds at a certain time point (e.g., `belief(at(steer_position(centre), t))`), or of an inequality (e.g., `t≤t2`).

The following dynamic monitor property is used as monitor focus within the Gaze Focus Monitoring Agent:

$$\exists t \forall t' [t \leq t' \leq t+D \rightarrow \text{belief}(\text{at}(\text{gaze_focus}(\text{far_away}), t'))]$$

This property expresses that there is a time period from t to $t+D$ in which the gaze of the driver is focused at a point far away. It is expressed in prefix notation as:

```
exists(t, forall(t', implies(and(t≤t', t'≤t+D), belief(at(gaze_focus(far_away), t'))))))
```

Within the LEADSTO model, this property was expressed as:

```
is_composed_of(gp(15), exists, t, gp(16, t))
is_composed_of(gp(16, t), forall, t', gp(17, t, t'))
is_composed_of(gp(17, t, t'),
  implies, gp(18, t, t'), gp(21, t, t'))
is_composed_of(gp(18, t, t'),
  and, gp(19, t, t'), gp(20, t, t'))
has_expression(gp(19, t, t'), t≤t')
has_expression(gp(20, t, t'), t'≤sum(t, D))
has_expression(gp(21, t, t'),
  belief(at(gaze_focus(far_away), t')))
```

Here, the top level expression (that is checked by the Gaze Focus Monitoring Agent) is called `gp(15)`. Given these monitoring foci, the monitoring agents detect the patterns in this sensor information, classify them as abnormal, and communicate this to the Driver Assessment Agent. By the multiple effect abductive reasoning method, this agent generates the belief that

the driver is having an impaired state, upon which a negative driver assessment is communicated to the Cruise Control Agent. The Cruise Control Agent first slows down the car, and after it stopped, blocks the ignition:

```
output(cruise_control_agent)|performing_in(slow_down_car,
car_and_environment)
output(cruise_control_agent)|performing_in(block_ignition,
car_and_environment)
```

Note that the dynamic monitor properties used within the Steering Monitoring Agent as well as the Gaze Focus Monitoring Agent make use of some temporal parameters ($t1$, $t2$, D) that indicate how long a particular pattern should persist before the behaviour is qualified as abnormal. Finding the most appropriate values for these parameters is an empirical question, which is nontrivial since they are likely to differ across individuals and contexts. In case sufficient training data about correct classification decisions are available, optimal values for these parameters can be estimated by means of Machine Learning techniques, such as Maximum Likelihood-based methods or Simulated Annealing (see [38]). For an example application of these techniques in an Ambient Intelligence context, see [5].

7. Verification of Dynamic Properties

This section addresses specification and verification of relevant dynamic properties of the cases considered, for example, requirements imposed on these systems. The approach used for verification is explained in Section 7.1. The following subsections address dynamic properties at different levels of the system, from the top level of the system as a whole (Section 7.2) to the level of individual components (Section 7.5).

7.1. Verification Approach

To enable the automated verification of dynamic properties specified in TTL against formal traces, a dedicated tool has been developed. This tool (called the TTL Checker Tool [4]) takes a dynamic property and one or more formal traces as input, and checks whether the dynamic property holds for the traces. Note that these checks can be performed irrespectively of who or what produced the formal traces: humans, simulators or an implemented (prototype) system. Thus, the TTL Checker Tool can be used to verify properties of both empirical traces, simulated traces and execution traces.

The Checker was implemented in Prolog, and offers a user-friendly graphical editor to create and edit dynamic properties based on tree structures (by means of graphical manipulation and filling in slots) and a graphical user interface to visualise traces (using XPCE, see the partial screenshot in Figure 4). A query to check some TTL formula against all loaded traces is compiled into a Prolog clause. Compilation is obtained by mapping conjunctions, disjunctions and negations of TTL formulae to their Prolog equivalents, and by transforming universal quantification into existential quantification. Thereafter, if this Prolog clause succeeds, the corresponding TTL formula holds with respect to all traces under consideration.

Since it does not ‘exhaustively’ check all possible traces (as in model checking, see, e.g., [13], [15]), the complexity of the checking algorithm is relatively low. It has an upper bound in the order of the product of the sizes of the ranges of all quantified variables. However, a number of specific optimisations make it possible to check realistic dynamic properties with reasonable performance. In practice, the duration of such checks usually varies from one second to a couple of minutes, depending on the complexity of the formula and the traces under consideration. With the increase of the number of traces, the checking time grows linearly. However, it is polynomial in the number of isolated time range variables occurring in the formula under analysis. For more (quantitative) details about the complexity of the checking algorithm, see [4].

7.2. Properties of the System as a whole

A natural property of the Ambient Driver Support System is that a driver with impaired driving behaviour cannot continue driving. The global property is:

GP1 No driving when symptoms of impaired driving occur

If the driver exposes symptoms that indicate that it is not safe to drive anymore then within 30 seconds the car will not drive and the engine will be off

$$\begin{aligned} &\forall \gamma:\text{TRACE}, t:\text{TIME}, R:\text{REAL} \quad (\text{unfocused_gaze}(t, \gamma) \wedge \\ &\quad \text{abnormal_steering_behaviour}(t, \gamma)) \Rightarrow \\ &\quad \exists t2:\text{TIME} < t:\text{TIME} + 30 \\ &[\text{state}(\gamma, t2, \text{internal}(\text{car_and_environment}))]= \\ &\quad \text{world_fact}(\text{car_not_driving})] \end{aligned}$$

This property makes use of two other properties:

UG Unfocussed gaze has occurred for some time.

In trace γ , during the time period D just before t, the gaze of the driver was focussed at a far distance.

$$\begin{aligned} &\forall t2:\text{TIME} \quad ((t2 \leq t) \wedge (t2 \geq t-D)) \Rightarrow \\ &[\text{state}(\gamma, t2, \text{internal}(\text{driver}))= \text{world_fact}(\text{gaze_focus}(\text{far_away}))] \end{aligned}$$

AS Abnormal steering behaviour has occurred

In trace γ , during a time period P just before t, whenever the steer is in a central position, there is time point within D time steps at which the steer is not in a central position.

$$\begin{aligned} &\forall t:\text{TIME} \quad ((t-P-D < t2) \wedge (t2 < t-D) \wedge \\ &[\text{state}(\gamma, t2, \text{internal}(\text{car_and_environment}))= \\ &\quad \text{world_fact}(\text{steer_position}(\text{centre}))]) \\ &\Rightarrow \exists t3:\text{TIME}, ((t \leq t3) \wedge (t3 \leq t-D) \wedge \\ &\quad \text{not}([\text{state}(\gamma, t3, \text{internal}(\text{car_and_environment}))= \\ &\quad \quad \text{world_fact}(\text{steer_position}(\text{centre}))])) \end{aligned}$$

The global property GP1 has been automatically verified (using the TTL Checker Tool) against the trace shown in the paper. For D a value of 3 has been used, which means that the driver should have an unfocussed gaze for at least 3 time steps, and that steering corrections should occur within 3 time steps. For P a value of 10 has been used, which means that continued steering corrections should be present for at least 10 time steps. Under these conditions, GP1 proved to hold for the generated trace.

7.3. Interlevel Relations Between Properties at Different Aggregation Levels

Following [22], dynamic properties can be specified at different aggregation levels. For the Ambient Driver Support system, three levels are used: properties of system as a whole, properties of subsystems, and properties of agents and the world within a subsystem. In Table 7 it is shown for the Ambient Driver Support System how the property at the highest level relates to properties at the lower levels. The lower level properties in the fourth column are described below.

Table 7. Properties and their interlevel relations

subsystems		components	
sensing	S1	steering, gaze-focus sensing	SSA1, GSA1
monitoring	M1	steering, gaze-focus monitoring	SMA1, GMA1
assessment	A1	driver assessment	DAA1
plan determination	P1	cruise control	CCA1, CCA2
subject process	SP1	driver, car/env	CE1, CE2

The property GP1 of the system as a whole can be logically related to properties of the subsystems (shown in the second column in the table) by the following inter level relation: S1 & M1 & A1 & P1 & SP1 \Rightarrow GP1. This expresses that the system functions well when all of the subsystems for sensing, monitoring, assessment, plan determination and the subject process function well.

7.4. Properties of Subsystems

The properties of subsystems of the Ambient Driver.Support System, as described in Table 7, are summarised below.

S1 Sensing system

If the sensory system receives observation input from the world and driver concerning gaze focus and steering operation, then it will provide as output this information for the monitoring system

M1 Monitoring system

If the monitoring system receives sensor information input concerning gaze-focus and steering operation from the sensing system, then it will provide as output monitoring information concerning qualification of gaze-focus and steering operation for the assessment system.

A1 Assessment system

If this system receives monitoring information concerning specific qualifications of gaze-focus and steering operation, then it will provide as output a qualification of the driver state.

P1 Plan determination system

If the plan determination system receives an overall qualification of the driver state, then it will generate as output an action to be undertaken.

SP1 Subject process

If the subject process receives an action to be undertaken, then it will obtain the effects of these actions.

If an impaired internal driver state occurs, then the driver will operate the steering wheel abnormally and the driver's gaze is unfocused.

7.5. Properties of Components

As indicated in Table 6 in the fourth column, each property of a subsystem is logically related to properties of the components within the subsystem. For example, the inter level relation $SSA1 \& GSA1 \Rightarrow S1$ expresses that the sensing subsystem functions well when each of the sensing agents functions well. Similarly, for the monitoring subsystem $SMA1 \& GMA1 \Rightarrow M1$. Properties characterising proper functioning of components are the following. The properties for the other sensing and monitoring agents ($GSA1$, $GMA1$) are similar.

SSA1 Steering Sensing agent

If the Steering Sensing agent receives observation results about steering wheel operation then it will communicate this observation information to the Steering Monitoring agent

SMA1 Steering Monitoring agent

If the Steering Monitoring agent receives observation results about the steering wheel, and this operation is abnormal, then it will communicate to the Driver Assessment Agent that steering operation is abnormal.

GSA1 Gaze Sensing agent

If the Gaze Sensing agent receives gaze observation results

then it will communicate this observation information to the Gaze Monitoring agent

GMA1 Gaze Monitoring agent

If the Gaze Monitoring agent receives gaze observation results, and this shows an abnormal pattern, then it will communicate to the Driver Assessment Agent that gaze is abnormal.

The properties for the Driver Assessment Agent are:

DAA1 Assessment of driving behaviour

If the Driver Assessment Agent receives input that steering operation is abnormal and gaze is unfocused, then it will generate as output communication to the Cruise Control agent that the driver state is inadequate

For the Cruise Control Agent the properties are:

CCA1 Slowing down a driving car

If the Cruise Control agent receives communication to that the driver state is inadequate, and the car is driving, then it will slow down the car.

CCA2 Turning engine off for a non driving car

If the Cruise Control agent receives communication that the driver state is inadequate, and the car is not driving, then it will turn off the engine.

The properties for the Car and Environment are:

CE1 Slowing down stops the car

If the Car and Environment components perform the slowing down action, then within 20 seconds the car will not drive.

CE2 Turning off the engine makes the engine off

If the Car and Environment components perform the turn off engine action, then within 5 seconds the engine will be off.

8. Additional Case Studies

Below, two additional cases are addressed. The first case concerns fall detection for elderly whereas the second case addresses the assessment of human attention for naval operators. Note that for both cases only parts of the case are explained for the sake of brevity.

8.1. Fall Detection of Elderly

For elderly, falls are a major problem, especially when they are still living independently, and do not have people continuously monitoring them. Research has shown that over 30% of the people aged above 75 falls at least one time per year [16]. Of course, warning the appropriate parties in case an elderly has fallen is essential to minimise the consequences of such a fall. Therefore, several technical solutions have been proposed (see e.g. [8], [37], [40]). These solutions vary from using infrared sensors [37] to using accelerometers [8], [40]. One crucial element in these technological solutions is that they are able to distinguish between a fall and everyday activities. Espe-

cially in case of elderly this is a hard problem since they have reduced muscle strength, and for instance tend to ‘fall’ into a couch when they sit down (which should not be seen as a real fall, of course). In the example worked out below, it is assumed that two sensors are available, namely accelerometers, and a positioning sensor indicating where the aged person is at the moment. The structure of the multi-agent system is shown in Figure 5. The interactions between the various agents are shown in more detail in Table 8.

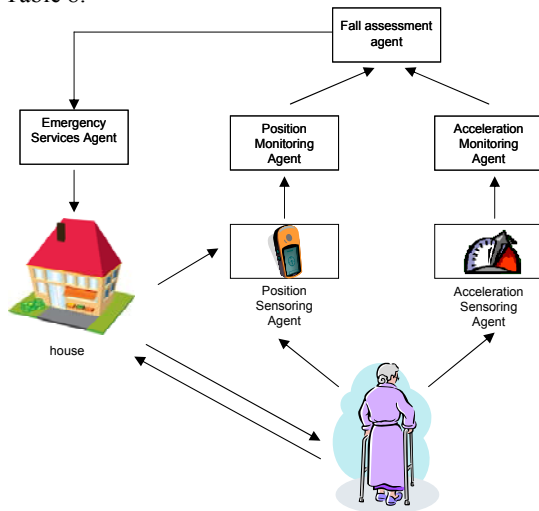


Fig. 5. Elderly Fall Detection System

Table 8. Components and Interactions of the Elderly Fall Detection System

subject components	subject agent	subject world component
	aged person	house
subject interactions	observation and action by subject agent in subject world component	
	aged person can observe the house and its contents	
	aged person can perform actions in the house, such as walking	
ambient components	ambient agents	
	position sensing agent, acceleration sensing agent, position monitoring agent, acceleration monitoring agent, fall assessment agent, emergency services agent	
ambient interactions	communication between ambient agents	

tions	position sensing agent communicates current position and information about objects at current position to position monitoring agent acceleration sensing agent communicates current acceleration to acceleration monitoring agent position monitoring agent reports to fall assessment agent whether the current position of the aged person allows for a fall or not acceleration monitoring agent reports to fall assessment agent whether a fall has been detected fall assessment agent communicates to the emergency services agent the status of the aged person
interactions between subject and ambient components	observation and action by ambient agent in subject world component
	position sensing agent can observe current position of aged person in the house as well as the objects present in the house at that position acceleration sensing agent observes the acceleration of the aged person emergency services agent can bring help into the house

The basic layout of the system is as follows. The aged person is of course the central player in the system, and resides in a house, in which actions are performed such as walking, sitting down, etc. The aged person is continuously observed by two agents, namely the position sensing agent that monitors the current position of the aged person, and also detects objects in the neighbourhood (e.g. a bed, chair, etc.), and the acceleration sensing agent that observes the movements of the aged person. Both pass information on to the accompanying monitoring agent. The acceleration monitoring agent follows the algorithm as proposed in [40]. This algorithm derives a potential fall in case a certain time period with high acceleration (above a certain threshold) has been observed, and thereafter the aged person does not move for a certain period (no accelerations measured). An example of a device which could measure such acceleration is for instance a mobile phone (almost all smart phones currently sold comprise of an accelerometer), but also dedicated devices can be utilised, such as a fall detector incorporating an accelerometer. The information is passed on to the fall assessment agent. For the case of the position monitoring agent, this agent derives that a location is potentially harmful in case there is no object at the current location of the aged person that would explain a high acceleration (e.g. a bed or a chair). This information is communicated to the fall assessment agent as well. The

fall assessment agent combines the information from the two monitoring agents, and in case a potential fall has been detected at a potentially harmful location, the fall assessment agent informs the emergency services agent that a fall has been detected, and that the emergency services should be warned. As a result, these will enter the house of the aged person.

In order to realise such a system, both monitoring agents will have to use specific dynamic monitor properties as their monitor focus (similar to the approach explained for the ambient driver support system in Section 6). As an illustration, for the Position Monitoring Agent, the following dynamic property is used as monitor focus:

$$\forall t \forall x [t1 \leq t \wedge t \leq t2 \wedge \text{belief}(\text{at}(\text{current_location}(x), t)) \rightarrow \text{not belief}(\text{at}(\text{soft_object_at}(x), t))]$$

This property expresses that whenever (between a given $t1$ and $t2$) the aged person is at a certain location x , there is no ‘soft’ object (i.e., an object that would explain a high acceleration) at this location. This dynamic property is expressed in prefix notation as:

```
forall(t, forall(x, implies(and(t1 ≤ t, and(t ≤ t2, belief(at(current_location(x), t))),
not(belief(at(soft_object_at(x), t))))))
```

Similarly, for the Acceleration Monitoring Agent, the following dynamic property is used as monitor focus:

$$[\forall t \forall x [t1 \leq t \wedge t \leq t2 \wedge \text{belief}(\text{at}(\text{acceleration}(x), t)) \rightarrow x > th]] \& [\forall t' \forall x' [t2 \leq t' \wedge t' \leq t3 \wedge \text{belief}(\text{at}(\text{acceleration}(x'), t')) \rightarrow x' = 0]]$$

This property expresses that (for a given $t1$, $t2$, and $t3$), between $t1$ and $t2$ the acceleration speed of the person is above a certain threshold th , and between $t2$ and $t3$ the acceleration speed of the person is 0. This dynamic property is expressed in prefix notation as:

```
and(forall(t, forall(x, implies(and(t1 ≤ t, and(t ≤ t2, belief(at(acceleration(x), t))), x > th))),
forall(t', forall(x', implies(and(t2 ≤ t', and(t' ≤ t3, belief(at(acceleration(x'), t'))), x' > th))))))
```

The complete specification of the system is not further elaborated, but the main idea is that the two monitoring agents continuously evaluate whether the above dynamic properties hold, in a manner similar to the driver case.

8.2. Assistance of Naval Operators

A third application concerns the support of naval operators. In the domain of naval warfare, it is crucial for the crew of the vessels involved to be aware of the situation in the field. Examples of important questions that should be addressed continuously are “in which direction are we heading?”, “are we currently

under attack?”, “are there any friendly vessels around?”, and so on. To assess such issues, one of the crew members is usually assigned the Tactical Picture Compilation Task (TPCT): the task to identify and classify all entities in the environment (e.g., [20]). This is done by monitoring a radar screen in the control room for radar contacts, and reasoning with the available information in order to determine the type and intent of the contacts on the screen. However, due to the complex and dynamic nature of the environment, this person has to deal with a large number of tasks in parallel. Often the radar contacts are simply too numerous and dynamic to be adequately monitored by a single human, which compromises the performance of the task.

For these reasons, it may be useful to offer this human operator some support from an intelligent ambient system, consisting of software agents that assist him in the execution of the Tactical Picture Compilation Task. For example, in case the human is directing its attention on the left part of a radar screen, but ignores an important contact that just entered the radar screen from the right, such a system may alert him about the arrival of that new contact. To be able to provide this kind of intelligent support, the system somehow needs to maintain a model of the cognitive state of the human: in this case the human’s focus of attention. It should have the capability to attribute mental, and in particular attentional (e.g., [21]) states to the human, and to reason about these. Below, an example of such a system is described, based on [6]. Three types of sensor information are assumed. First of all, a sensor is assumed that provides information about the human’s gaze. An example of such a sensor is for instance an eye tracker whereby infrared light is used to determine a human’s gaze. Such devices can be mobile (e.g. installed in a cap) or fixed (for instance mounted below a screen), for examples of these eye trackers, see e.g. www.tobii.com. Secondly, a sensor is assumed which provides information about the human’s actions (e.g., by measuring mouse clicks and keystrokes), and finally a sensor is required which measures the characteristics of stimuli (e.g., the airplanes on the radar screen). The structure of the multi-agent system is shown in Figure 6. The interactions between the various agents are shown in more detail in Table 9.

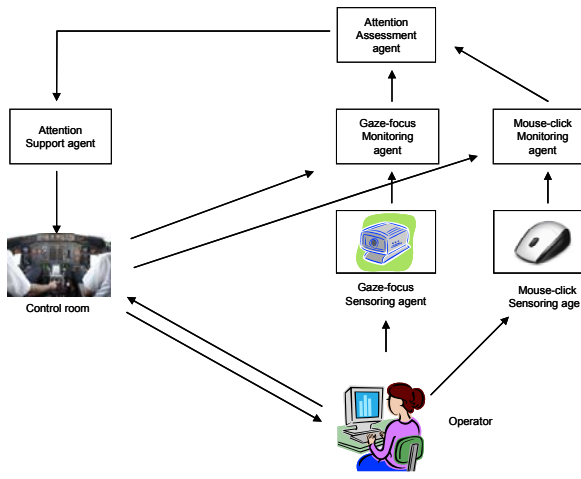


Fig. 6. Naval Operator Support System

Table 9. Components and interactions for a naval operator case

subject components	subject agents		subject world components
	human naval operator		
subject interactions	observation and action by subject agent		
naval operator gaze focuses on radar screen with planes, extracts information from radar screen view, naval operator acts on planes that are dangerous			
ambient components	ambient agents		
gaze-focus sensing agent, mouse-click sensing agent, gaze-focus monitoring agent, mouse-click monitoring agent, attention assessment agent, attention support agent			
ambient interactions	communication between ambient agents		
gaze-focus sensing agent communicates gaze focus to gaze-focus monitoring agent mouse-click sensing agent communicates mouse clicks to mouse-click monitoring agent gaze-focus monitoring agent communicates estimation of locations with too little attention to attention assessment agent mouse-click monitoring agent communicates estimation of locations with too little attention to attention assessment agent attention assessment agent communicates locations with too little attention to attention support agent			
interactions between sub-	communication	observation and action	

subject and ambient	attention support agent communicates overlooked dangerous item to naval operator	gaze-focus sensing agent observes operator gaze mouse-click sensing agent observes operator mouse clicks
---------------------	--	---

The global behaviour of the system is as follows. The naval operator continuously observes the radar screen, trying to identify and classify all items on the screen (using the mouse). The operator's behaviour is observed by two agents, namely the gaze-focus sensing agent that observes his gaze focus, and the mouse-click sensing agent that observes the mouse clicks he performs. Both agents pass information on to the accompanying monitoring agent. Moreover, the two monitoring agents receive domain-specific information from the radar screen (such as the color, speed, and so on, of the items), which provides a cue for which items deserve attention from the operator. The gaze-focus monitoring agent then makes an estimation of which items are overlooked, which is mainly based on the distance between an item and the point of gaze (using an algorithm as proposed in [6]). This estimation is passed on to the attention assessment agent. Similarly, the mouse-click monitoring agent also makes an estimation which items are overlooked, but this time based on the amount of clicks the operator has performed with respect to the items. This information is also communicated to the attention assessment agent. The attention assessment agent combines the information from the two monitoring agents, and makes a final decision about which items receive too little attention from the operator. In case the attention assessment agent decides that a particular item currently receives less attention that it deserves, it communicates the location of this item to the attention support agent. This agent will then determine what is the best way to bring this item under the operator's attention (e.g., by modifying its luminance, or color). Alternatively, in case the operator is already very busy, the attention support agent may also decide to take over the task of classifying the item.

In order to realise this system, again, both monitoring agents will have to use specific dynamic monitor properties as their monitor focus. As an illustration, for the Gaze-Focus Monitoring Agent, the following dynamic property is used as monitor focus:

$$\forall t \forall x \forall y [t1 \leq t \wedge t \leq t2 \wedge \text{belief}(\text{at}(\text{gaze_focus}(x), t)) \wedge \text{belief}(\text{at}(\text{attention_required_at}(y), t)) \rightarrow d(x, y) > th]$$

This property expresses that whenever (between a given t1 and t2) the operator's attention is at a certain

location x , and location y is one of the locations that deserve attention, then the distance between x and y (denoted by $d(x,y)$) is above a certain threshold th . This dynamic property is expressed in prefix notation as:

```
forall(t, forall(x, forall(y,
  implies(and(t1 ≤ t, and(t ≤ t2, belief(at(gaze_focus(x), t)),
    belief(at(attention_required_at(y), t))), d(x,y)>th))))))
```

Similarly, for the Mouse-Click Monitoring Agent, the following dynamic property is used as monitor focus:

```
∀t ∀x [[ t1 ≤ t ∧ t ≤ t2 ∧ at(attention_required_at(x), t) ] → [ ∃t' t1 ≤ t' ∧ t' ≤ t2 ∧ at(mouse_click_at(x), t') ]]
```

This property expresses that for all locations x (between a given $t1$ and $t2$) that deserve attention, the operator clicks on that location at a certain time point t' (between $t1$ and $t2$). This dynamic property is expressed in prefix notation as:

```
forall(t, forall(x, implies(and(t1 ≤ t, and(t ≤ t2,
  belief(at(attention_required_at(x), t))))),
  exists(t', and(t1 ≤ t', and(t' ≤ t2, at(mouse_click_at(x), t'))))))
```

Again, the monitoring agents continuously evaluate whether these dynamic properties hold. In this way, the system is able to analyse information from several sensors and to monitor complex behaviour. The output of each monitoring agent will be a list of (potentially multiple) locations that are overlooked. Next, these lists are communicated to the attention assessment agent, which determines for which location the human will actually receive support.

9. Discussion

The ambient agent-based model introduced in this paper is described at an implementation-independent conceptual design level. In order to handle dynamic patterns in human and environmental processes well, it has facilities built in to represent models for human states and behaviours, dynamic process models, and analysis methods on the basis of such models. The agent model involves both formally specified generic and specific content and provides a detailed component-based executable design for a working prototype system. The specific content, together with the generic methods to operate on it, enables ambient agents to react in a knowledgeable manner.

In the major case study, it was shown how the different types of agents work together to support safety of the driving and the driver; e.g., [11], [17], [25], [26], [30], [31], [39]. Simulation experiments have been conducted and the outcomes have been formally

analysed, thus showing in how far the system indeed supports safety. For the monitoring agents, specific patterns of gaze and steering behaviour were chosen and formalised in a temporal language as monitor foci. However, as the approach is more general, it is easy to use different, more sophisticated monitoring foci. It would be interesting further experimental research to find out which types of observable deviations of driving behaviour can be found as effects of different types of impaired internal states, for example caused by drugs, or by becoming sleepy, and use results of this to obtain more sophisticated monitoring foci and actions.

Two other application scenario's were described to illustrate the generality of the agent model. Both in the case of intelligent fall detection for elderly and attention support for naval officers, it appeared that it is necessary to analyse more complex behaviour than the mere fact that some atomic event has been detected or that the value of a sensor has reached some threshold. For example, for the fall detection system, only measuring a person's location or acceleration speed at one particular time point is not sufficient (see also [40]). Instead, it is needed to measure these states over a number of consecutive time points, and compare them in an intelligent manner, in order to conclude that there is an alarming situation. Similarly, for the naval attention support system, only measuring a person's gaze at one particular time point is not sufficient [6]. Here, again, it is necessary to combine different types of information (e.g., about gaze, task demands, and mouse clicks) over time. The ability of the presented model to specify these complex behaviour patterns is a major difference with existing work on agent models for ambient intelligence applications. For example, [24] also presents a framework that utilizes mobile agents for ambient intelligence in a distributed ubiquitous environment. In this work, however, the focus is not on the properties or behaviour that should be monitored, but on the architecture and the mathematical formulation (using π -calculus) that can be used for evaluation and verification. Similarly, in [12], [41], a multi agent-based framework for a typical ambient intelligence "space" is proposed. It provides a hierarchical system model for an AmI space, a model of the middleware and of the physical structure of the application. However, the specific monitoring objectives are not yet described.

Other approaches do not provide a general model for ambient intelligent applications, but focus on a specific task or domain. For example, the main emphasis in the work presented in [19] is on learning

and adaptation techniques for agents within an ambient intelligence application. Its goal is to provide online, personalised learning of anticipatory adaptive control to realize the ambient intelligence vision in ubiquitous-computing environments. To this aim, an intelligent dormitory has been developed as test-bed and demo environment. In [28], a multi-agent based ambient intelligence application for a specific domain has been developed. The multi-agent system AMICO is able to support and follow users along a manufactory laboratory, offering them the information needed anytime in the most suitable device available. In [27] a framework is presented to enable home healthcare. The framework enables the observation of patients' clinical data via wearable devices, and of movements via sensor networks. Based on these types of information, habits and actions are derived by means of logic programming techniques.

The model presented in this article is domain independent and provides support on various levels of system design, i.e. the overall architecture and interaction between agents, the design of individual agents, and the level of monitoring complex behaviour. Thus it provides a reusable application model that can be considered an agent-based Ambient Intelligence system (cf. [1], [2], [33], [29], [34]).

Future work will include implementation of the conceptual models presented in this paper in a real world environment, and their validation in the context of experiments with human participants. For the case of the naval operator support system, a prototype system has already been implemented. Preliminary experiments point out that the system successfully improves performance of participants that execute a variant of the Tactical Picture Compilation Task [5].

References

- [1] Aarts, E., Harwig, R., and Schuurmans, M. (2001). Ambient Intelligence. In: P. Denning (ed.), *The Invisible Future*, McGraw Hill, New York, pp. 235-250.
- [2] Aarts, E.; Collier, R.; van Loenen, E.; Ruyter, B. de (eds.) (2003). Ambient Intelligence. Proc. of the First European Symposium, EUSAI 2003. Lecture Notes in Computer Science, vol. 2875. Springer Verlag, 2003, pp. 432.
- [3] Bosse, T., Jonker, C.M., Meij, L. van der, and Treur, J. (2007). A Language and Environment for Analysis of Dynamics by Simulation. *International Journal of Artificial Intelligence Tools*, vol. 16, 2007, pp. 435-464.
- [4] Bosse, T., Jonker, C.M., Meij, L. van der, Sharpanskykh, A., and Treur, J. (2009). Specification and Verification of Dynamics in Agent Models. In: *International Journal of Cooperative Information Systems*, volume 18, 2009, pp. 167-193.
- [5] Bosse, T., Lambalgen, R. van, Maanen, P.P. van, and Treur, J. (2009). Attention Manipulation for Naval Tactical Picture Compilation. In: Baeza-Yates, R., Lang, J., Mitra, S., Parsons, S., and Pasi, G. (eds.), *Proceedings of the 9th IEEE/WIC/ACM International Conference on Intelligent Agent Technology, IAT'09*. IEEE Computer Society Press, 2009, pp. 450-457. Extended version: A System to Support Attention Allocation: Development and Application. *Web Intelligence and Agent Systems Journal*, 2011, to appear
- [6] Bosse, T., Maanen, P.P. van, and Treur, J. (2009). Simulation and Formal Analysis of Visual Attention. *Web Intelligence and Agent Systems Journal*, vol. 7, 2009, pp. 89-105 Preliminary, shorter version in: Nishida, T., Klusch, M., Sycara, K., Yokoo, M., Liu, J., Wah, B., Cheung, W., and Cheung, Y.-M. (eds.), *Proceedings of the Sixth International Conference on Intelligent Agent Technology, IAT'06*. IEEE Computer Society Press, 2006, pp. 255-262.
- [7] Both, F., Hoogendoorn, M., Jaffry, S.W., Lambalgen, R. van, Oorburg, R., Sharpanskykh, A., Treur, J., and Vos, M. de (2009). Adaptation and Validation of an Agent Model of Functional State and Performance for Individuals. In: Yang, J.-J., Yokoo, M.; Ito, T.; Jin, Z.; Scerri, P. (eds.), *Proceedings of the 12th International Conference on Principles of Practice in Multi-Agent Systems, PRIMA'09*. Lecture Notes in Artificial Intelligence, vol. 5925. Springer Verlag, 2009, pp. 595-607.
- [8] Bourke, A.K., O'Brien, J.V., and Lyons, G.M., (2007). Evaluation of a threshold tri-axial accelerometer fall detection algorithm. *Gait and Gesture*, vol. 25, 2007, pp. 194-199.
- [9] Brazier, F.M.T., Jonker, C.M., and Treur, J. (2000). Compositional Design and Reuse of a Generic Agent Model. *Applied Artificial Intelligence Journal*, vol. 14, 2000, pp. 491-538.
- [10] Brazier, F.M.T., Jonker, C.M., and Treur, J. (2002). Principles of Component-Based Design of Intelligent Agents. *Data and Knowledge Engineering*, vol. 41, 2002, pp. 1-28.
- [11] Brookhuis, K.A., and De Waard, D. (2001). Assessment of drivers' workload: Performance and subjective and physiological indexes, in Stress, Workload, and Fatigue, P.A. Hancock and P.A. Desmond, Editors. Lawrence Erlbaum Associates, Mahwah, NJ, 2001, pp. 321-333.
- [12] Chen, R., Hou, Y., Huang, Z., Zhang, Y., Li, H. (2007) Framework for Local Ambient Intelligence Space: The Aml-Space Project. In: *Proceedings of the Computer Software and Applications Conference, 2007 (COMPSAC 2007)*. Volume 2, 24-27 July 2007, 95 - 100.
- [13] Clarke, E.M., Grumberg, O., and Peled, D.A. (2000). *Model Checking*. MIT Press.
- [14] Costantini, S., Dell'Acqua, P., Pereira, L.M., and Tsintza, P. (2008). Specification and Dynamic Verification of Agent Properties. In: Fisher, M., Sadri, F., and Thielscher, M. (eds.), *Proceedings of the Ninth International Workshop on Computational Logic in Multi-Agent Systems, CLIMA-IX*.
- [15] Dima, C. (2009). Revisiting Satisfiability and Model-Checking for CTLK with Synchrony and Perfect Recall. In: Fisher, M., Sadri, F., and Thielscher, M. (eds.), *Proceedings of the Ninth International Workshop on Computational Logic in Multi-Agent Systems, CLIMA-IX*.
- [16] Duthie, E. (1989), *Falls*. Medical Clinics of North America, vol. 73, 1989, pp. 1321-1335.
- [17] Engström, J., Johansson, E. and Östlund, J. (2005). Effects of visual and cognitive load in real and simulated motorway driving. *Transportation Research Part F*, vol. 8, (2005), pp. 97-120.
- [18] Gibbs, W.W. (2005) Considerate computing. *Scientific American* 292(1):54-61.
- [19] Hagrais, H., Callaghan, V., Colley, M., Clarke, G., Pounds-Cornish, A., and Duman, H. 2004. Creating an Ambient

- Intelligence Environment Using Embedded Agents. *IEEE Intelligent Systems* 19, 6 (Nov. 2004), 12-20.
- [20] Heuvelink, A., Both, F. (2007). Boa: A cognitive tactical picture compilation agent. In: *Proceedings of the 2007 IEEE/WIC/ACM International Conference on Intelligent Agent Technology, IAT 2007*. IEEE Computer Society Press, 2007, pp. 175-181.
- [21] Itti, L. and Koch, C. (2001). Computational Modeling of Visual Attention, *Nature Reviews Neuroscience*, Vol. 2, No. 3, 2001, pp. 194-203.
- [22] Jonker, C.M., and Treur, J. (2002). Compositional Verification of Multi-Agent Systems: a Formal Analysis of Proactiveness and Reactiveness. *International Journal of Cooperative Information Systems*, vol. 11, 2002, pp. 51-92.
- [23] Jonker, C.M., and Treur, J. (1999). Formal Analysis of Models for the Dynamics of Trust based on Experiences. In: F.J. Garijo, M. Boman (eds.), *Multi-Agent System Engineering, Proceedings of the 9th European Workshop on Modelling Autonomous Agents in a Multi-Agent World, MAAMAW'99*. Lecture Notes in AI, vol. 1647, Springer Verlag, Berlin, 1999, pp. 221-232.
- [24] Lee, Y., Khorasani, E. S., Rahimi, S., and Gupta, B. (2008). A generic mobile agent framework for ambient intelligence. In *Proceedings of the 2008 ACM Symposium on Applied Computing (Fortaleza, Ceara, Brazil, March 16 - 20, 2008)*. SAC '08. ACM, New York, NY, 1866-1871.
- [25] Mehler, B., Reimer, B., Coughlin, J.F., and Dusek, J.A.. (2009). The impact of incremental increases in cognitive workload on physiological arousal and performance in young adult drivers. *Transportation Research Record*, vol. 2138, 2009, pp. 6-12.
- [26] Mehler, B., Reimer, B. & Coughlin, J.F. (2010). Physiological Reactivity to Graded Levels of Cognitive Workload across Three Age Groups: An On-Road Evaluation. *Proceeding of the 54th Annual Meeting of the Human Factors and Ergonomics Society*, pp. 2062-2066.
- [27] Mileo, A., Merico, D., and Bisiani, R. (2007). CyberCare: Reasoning about Patient's Profile in Home Healthcare. In: *Proceedings of the Symposium on "Artificial Societies for Ambient Intelligence", ASAmI'07, 2007*.
- [28] Perez, M.A., Susperregi, L., Maurtua, I., Ibarguren, A., Sierra, B. Software Agents for Ambient Intelligence based Manufacturing, *Proceedings of the IEEE Workshop on Distributed Intelligent Systems: Collective Intelligence and Its Applications*, p.139-144, June 15-16, 2006.
- [29] Ramos, C., Augusto, J.C., and Shapiro, D. (2008). Ambient Intelligence - the Next Step for Artificial Intelligence (guest editors' introduction to the special issue on ambient intelligence), *IEEE Intelligent Systems*, vol. 23, issue 2, pp.15-18, Mar/Apr 2008.
- [30] Reimer, B., Mehler, B., Coughlin, J.F., Godfrey, K.M., and Tan, G. (2009). An On-Road Assessment of the Impact of Cognitive Workload on Physiological Arousal in Young Adult Drivers. In: *Proceedings of the First International Conference on Automotive User Interfaces and Interactive Vehicular Applications (AutomotiveUI 2009)*, 2009, pp. 115-118.
- [31] Reimer, B., Mehler, B., Wang, Y. & Coughlin, J.F. (2010). The Impact of Systematic Variation of Cognitive Demand on Drivers' Visual Attention across Multiple Age Groups. *Proceeding of the 54th Annual Meeting of the Human Factors and Ergonomics Society*, pp. 2052-2056.
- [32] Reiter, R. (2001). *Knowledge in Action: Logical Foundations for Specifying and Implementing Dynamical Systems*. MIT Press, 2001.
- [33] Riva, G., F. Vatalaro, F. Davide, M. Alcañiz (eds.) (2005). *Ambient Intelligence*. IOS Press.
- [34] Sadri, F. and Stathis, K. (2009). Artificial Societies for Ambient Intelligence (guest editors' introduction to the special issue on artificial societies for ambient intelligence), *IEEE Intelligent Systems*, April 2009.
- [35] Sadri, F. (2010). Intention Recognition with Event Calculus Graphs. In: *Proceedings of the Fourth International Workshop on Human Aspects in Ambient Intelligence, HAI'10*. IEEE Computer Society Press, 2010, to appear.
- [36] Sharpanskykh, A., and Treur, J., A Temporal Trace Language for Formal Modelling and Analysis of Agent Systems. In: Dastani, M., Hindriks, K.V., and Meyer, J.J.Ch. (eds.), *Specification and Verification of Multi-Agent Systems*. Springer Verlag, 2010, pp. 317-352.
- [37] Sixsmith, A. and Johnson, N., (2004). A Smart Sensor to Detect the Falls of the Elderly. *IEEE Pervasive Computing*, vol. 3, 2004, pp. 42-47.
- [38] Sorenson, H.W. (1980). *Parameter estimation: principles and problems*. Marcel Dekker, Inc., New York.
- [39] Tawari, A. and Trivedi, M. M. (2010). Speech based emotion classification framework for driver assistance system, in *IEEE Intelligent Vehicles Symp*, 2010, pp. 174 - 178
- [40] Ven, P. v.d., Bourke, A.L., Nelson, J., and Laughin, G.O., (2008). A Wearable Wireless Platform for Fall and Mobility Monitoring. In: *PETRA 2008*, 2008.
- [41] Zhang, Y., Hou, Y., Huang, Z., Li, H., Chen, R. (2006). A Context-Aware Aml System Based on MAS Model, pp.703-706, 2006 International Conference on Intelligent Information Hiding and Multimedia Signal Processing (IIH-MSP'06).