A Component-Based Ambient Agent Model for Assessment of Driving Behaviour

Tibor Bosse, Mark Hoogendoorn, Michel C.A. Klein, and Jan Treur

Vrije Universiteit Amsterdam, Department of Artificial Intelligence,
de Boelelaan 1081a, 1081 HV Amsterdam, The Netherlands
{tbosse, mhoogen, mcaklein, treur}@cs.vu.nl
http://www.cs.vu.nl/~{tbosse, mhoogen, mcaklein, treur}

Abstract. This paper presents an ambient agent-based model that addresses the assessment of driving behaviour. In case of negative assessment, cruise control slows down and stops the car. The agent model has been formally specified in a component-based manner in the form of an executable specification that can be used for prototyping. A number of simulation experiments have been conducted. Moreover, dynamic properties of components at different aggregation levels and interlevel relations between them have been specified and verified.

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], [9]. Applications make use of possibilities to acquire sensor information about humans and their functioning, and knowledge for analysis of such information. Based on this, ambient devices can (re)act by undertaking actions in a knowledgeable manner that improve the human's, safety, health, performance, and wellbeing.

The focus of this paper is on driving behaviour. 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 wider applicable option. A component-based ambient agent-based model is presented 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.

F.E. Sandnes et al. (Eds.): UIC 2008, LNCS 5061, pp. 229–243, 2008.

The ambient agent-based system model that is presented and analysed here includes four types of agents: sensoring agents, monitoring agents, a driver assessment agent, and a cruise control agent (see also Figure 1). Models for all of these types of agents have been designed as specialisations of a more general component-based Ambient Agent Model. Within the model of the driver assessment agent, a model of a driver's functioning is used expressing that an impaired internal state leads to observable behaviour showing abnormal steering operation and unfocused gaze. The monitor agent model includes facilities to automatically analyse incoming streams of information by verifying them on temporal patterns that are to be monitored (for example, instable steering operation over time). The design has been formally specified in the form of a component-based executable agent-based model that can be used for prototyping. A number of simulation experiments have been conducted. Moreover, dynamic properties of components at different aggregation levels and interlevel relations between them have been formally specified and verified against these traces.

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 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, and finally Section 8 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. The modelling approach uses the Temporal Trace Language TTL for formal specification and verification of dynamic properties [3], [7]. This predicate logical language supports formal specification and analysis of dynamic properties, covering both qualitative and quantitative aspects. TTL is built on atoms referring to states, time points and traces. 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 STATES(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 $\gamma : T \rightarrow STATES(Ont)$, i.e., a sequence of states γ_t ($t \in T$) in STATES(Ont). The set of *dynamic properties* DYNPROP(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 state(γ, t). These states can be related to state properties via the formally defined satisfaction relation |=, comparable to the Holds-predicate in the Situation Calculus [8]: state(γ, t) |= 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.

To specify simulation models and to execute these models, the language LEADSTO, an executable sublanguage of TTL, is used (cf. [4]). The basic building blocks of this language are causal relations of the format $\alpha \rightarrow\!\!\!\!\rightarrow_{e, f, g, h} \beta$, 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.

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., [5], [6]), 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 *communication*. Interaction between an agent and a world component can be either *observation* or *action* performance; cf. [5]. 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. In Figure 1 an overview of the system is shown. Table 1 shows the structure in terms of different types of components and interactions.

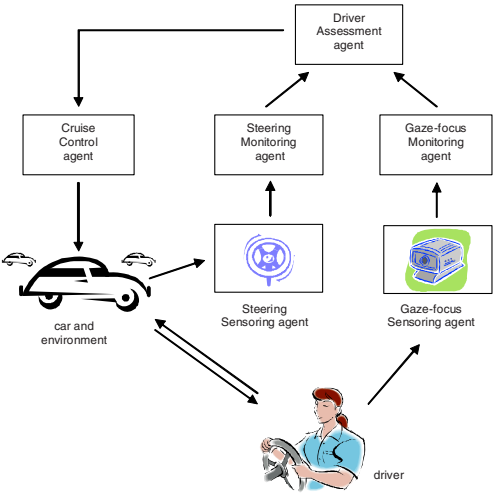

Fig. 1. Ambient Driver Support System

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.

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 sensoring agent; steering and gaze-focus monitoring agent; driver assessment agent, cruise control agent	
ambient interactions	**communication between ambient agents**	
	steering sensoring agent communicates to steering monitoring agent gaze-focus sensoring 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 sensoring agent observes steering wheel gaze-focus sensoring 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
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

3.2 Temporal Relations at the Global Level

Interaction between global level components is defined by the following specifications. In such specifications, for state properties the prefix input, output or internal is used. 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.

Action Propagation from Agent to World Component
∀X:AGENT ∀W:WORLD ∀A:ACTION output(X)|performing_in(A, W) → input(W)|performing_in(A, W)

Observation Result Propagation from World to Agent
∀X:AGENT ∀W:WORLD ∀I:INFO_EL output(W)|observation_result_from(I, W)
→ input(X)|observed_result_from(I, W)

Communication Propagation Between Agents
∀X,Y:AGENT ∀I:INFO_EL output(X)|communication_from_to(I,X,Y) →input(Y)|communicated_from_to(I,X,Y)

4 Component-Based Ambient Agent Model

This section focuses on an Ambient Agent Model (AAM) used for the four types of ambient agents in 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 monitor agents, a driver assessment agent, and a cruise control agent.

4.1 Components within the Ambient Agent Model

Based on the component-based Generic Agent Model (GAM) presented in [5], 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. In the component *Agent Specific Task*, specific tasks can be modelled. Adopting this component-based agent model GAM, the ambient agent model has been obtained as a refinement in the following manner.

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 information 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 belief(leads_to_after(I:INFO_EL, J:INFO_EL, D: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

The temporal relations for the functionality within the Ambient Agent Model are:

Belief Generation based on Observation and Communication
∀X:AGENT, I:INFO_EL, W:WORLD
input(X)|observed_from(I, W) ∧ internal(X)|belief(is_reliable_for(W, I)) ⇛ internal(X)|belief(I)

∀X,Y:AGENT, I:INFO_EL
input(X)|communicated_from_ to(I,Y,X) ∧ internal(X)|belief(is_reliable_for(X, I)) ⇛ internal(X)|belief(I)

Here, the first rule is a generic rule for the component *World Interaction Management*, and the second for the component *Agent Interaction Management*. When the sources are assumed always reliable, the conditions on reliability can be left out.

Belief Generation based on Simulation
∀X:AGENT ∀I,J:INFO_EL ∀D:REAL ∀T:TIME
internal(X)|belief(at(I, T)) ∧ internal(X)|belief(leads_to_after(I, J, D)) ⇛ internal(X)|belief(at(J, T+D))

The last generic rule 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
∀X:AGENT ∀I,J1, J2:INFO_EL ∀D:REAL ∀T:TIME
J1≠J2 ∧ internal(X)|belief(at(J1, T)) ∧ internal(X)|belief(leads_to_after(I, J1, D)) ∧
internal(X)|belief(at(J2, T)) ∧ internal(X)|belief(leads_to_after(I, J2, D)) ⇛ internal(X)|belief(at(I, T-D))

4.4 Generic Temporal Relations within a World

For World Components the following specifications indicate the actions' effects and how observations provide their results.

Action Execution and Observation Result Generation in a World
∀W:WORLD_COMP ∀A:ACTION ∀I:INFO_EL input(W)|performing_in(A, W) ∧ internal(W)|has_effect(A,I)

$\rightarrow\!\!\!\!\rightarrow$ internal(W)|world_fact(I)

∀W:WORLD_COMP ∀I:INFO_EL
input(W)|observation_focus_in(I, W) ∧ internal(W)|world_fact(I) $\rightarrow\!\!\!\!\rightarrow$ output(W)|observation_result_from(I, W)

∀W:WORLD_COMP ∀I:INFO_EL input(W)|observation_focus_in(I, W) ∧ internal(W)|world_fact(not(I)) $\rightarrow\!\!\!\!\rightarrow$ output(W)|observation_result_from(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. Table 4 indicates the components within these monitoring agents. These agents relate temporal patterns of gaze, resp. steering to qualifications of abnormality.

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

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 sensoring 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. As output a monitoring agent generates communication that a certain monitoring focus holds.

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: ∃V [has_value(X, V) ∧ LB ≤ V ∧ V ≤ UB]. In prefix notation, this can be expressed as follows: exists(V, and(has_value(X, V), and(LB ≤ V, V ≤ UB))). It is possible to obtain abstraction by using (meaningful) names of properties. For example, stable_within(X, LB, UB) can be used as an abstract name for the example property expressed above by specifying:

has_expression(stable_within(X, LB, UB), exists(V, and(has_value(X, V), and(LB ≤ V, V ≤ UB))))

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

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

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:

```
forall(t, implies(and(t1≤t, and(t≤t2, at(has_value(X, V1), t))),
      exists(t', exists(V2,   and(t≤ t', and(t' ≤ t+D, and(V2 ≠V1, at(has_value(X, V2), t'))))
```

This expression can be named, for example, by 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.

```
communicated_from_to(monitor_focus(F), A, B)
communicated_from_to(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 input(X)|, which indicates 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) $\twoheadrightarrow$ 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) $\twoheadrightarrow$ 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.

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. This decomposition process was specified in the following manner:

monitor_focus(F) $\twoheadrightarrow$ in_focus(F)

in_focus(E) $\wedge$ is_composed_of (E, C, E1, E2) $\twoheadrightarrow$ in_focus(E1) $\wedge$ in_focus(E2)

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).

Monitoring Foci Verification. 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) $\twoheadrightarrow$ verification(E, pos)

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

in_focus(and(E1, E2)) $\wedge$ verification(E1, pos) $\wedge$ verification(E2, pos) $\twoheadrightarrow$ verification(and(E1, E2) , pos)

in_focus(or(E1, E2)) $\wedge$ verification(E1, pos) $\twoheadrightarrow$ verification(or(E1, E2) , pos)

in_focus(or(E1, E2)) ∧ verification(E2, pos) ⟶⟶ verification(or(E1, E2) , pos)
in_focus(implies(E1, E2)) ∧ verification(E2, pos) ⟶⟶ verification(implies(E1, E2), pos)
in_focus(implies(E1, E2)) ∧ not verification(E1, pos) ⟶⟶ verification(implies(E1, E2), pos)
in_focus(not(E)) ∧ not verification(E, pos) ⟶⟶ verification(not(E), pos)
in_focus(exists(V, E)) ∧ verification(E, pos) ⟶⟶ verification(exists(V, E), pos)
in_focus(forall(V, E)) ∧ not verification(exists (V, not(E), pos) ⟶⟶ verification(forall(V, E), pos)

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) ⟶⟶ verification(E, neg)

The following relates verification of an expression to its name:

verification(E, S) ∧ has_expression(F, E) ⟶⟶ 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:

verification(E, pos) ∧ internal(monitoring_agent)|belief(is_indication_for(E, I))
⟶⟶ internal(monitoring_agent)|belief(I)

5.2 Driver Assessment Agent

As another refinement of the Ambient Agent Model AAM, the Driver Assessment Agent Model DAAM; see Table 5 for an overview of the different components. For the Driver Assessment Agent, a number of domain-specific rules have been identified in addition to the generic rules specified for the Ambient Agent Model presented in Section 4. Some of the key rules 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 5. 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 focussing
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:

internal(driver_assessment_agent)|belief(leads_to_after(impaired_state, abnormal_steering_operation, D))
internal(driver_assessment_agent)|belief(leads_to_after(impaired_stste, 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 rule 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 rule specifies 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 rules as described in previous section, a specification within the LEADSTO software environment (cf. [4]) has been made and simulation runs of the system have been generated, of which an example trace is shown in Figure 2. 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 sensoring agent to the

Fig. 2. Example Simulation Trace

corresponding monitoring agent. The following dynamic monitor property is used as monitor focus within the Steering Monitoring Agent:

$$\forall t\ [\ t1 \le t \wedge t \le t2 \wedge \text{belief}(\text{at}(\text{steer\_position}(\text{centre}), t)) \rightarrow \exists t'\ \ t \le t' \le t+D \wedge \text{not belief}(\text{at}(\text{steer\_position}(\text{centre}), t'))$$

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'))))
```

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: ∃t ∀t' [t ≤ t' ≤ t+D → belief(at(gaze_focus(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)
```

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.

7.1 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
∀γ:TRACE, t:TIME, R:REAL (unfocused_gaze(t, γ) ∧ abnormal_steering_behaviour(t, γ)) ⇒
∃t2:TIME < t:TIME + 30 [state(γ, t2, internal(car_and_environment))|= world_fact(car_not_driving)]

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.
∀t2:TIME ((t2 <= t) ∧ (t2 >= t-D)) ⇒ [state(γ, t2, internal(driver)|= world_fact(gaze_focus(far_away))]

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.

∀t:TIME ((t-P-D < t2) ∧ (t2 < t-D) ∧
[state(γ, t2, internal(car_and_environment)|= world_fact(steer_position(centre))])
⇒ ∃t3:TIME, ((t <= t3) ∧ (t3 <= t-D) ∧
not ([state(γ, t3, internal(car_and_environment)|= world_fact(steer_position(centre))]))

The global property GP1 has been automatically verified (using the TTL checker tool [3]) 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.

7.2 Interlevel Relations between Properties at Different Aggregation Levels

Following [7], 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 6 it is shown for the Ambient Driver Support System how the property at the highest level relates to properties at the lower levels (see also Figure 2). The lower level properties in the fourth column are described below.

Table 6. Properties and their interlevel relations

subsystems		components	
sensoring	S1	steering, gaze-focus sensoring	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 ⇒ GP1. This expresses that the system functions well when all of the subsystems for sensoring, monitoring, assessment, plan determination and the subject process function well.

7.3 Properties of Subsystems

S1 Sensoring 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 sensoring 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.

7.4 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 sensoring subsystem functions well when each of the sensoring 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 sensoring and monitoring agents (GSA1, GMA1) are similar.

SSA1 Steering Sensoring agent
If the Steering Sensoring 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 Sensoring agent
If the Gaze Sensoring 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 Discussion

The ambient agent-based model introduced in this paper is described at an implementation-independent conceptual design level. 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 model involves both 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. Thus a reusable application model was obtained that can be considered an agent-based Ambient Intelligence system (cf. [1], [2], [9]). It was shown how the different types of agents work together to support safety of the driving and the driver. 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.

References

1. Aarts, E., Collier, R.W., van Loenen, E., de Ruyter, B. (eds.): EUSAI 2003. LNCS, vol. 2875, p. 432. Springer, Heidelberg (2003)
2. Aarts, E., Harwig, R., Schuurmans, M.: Ambient Intelligence. In: Denning, P. (ed.) The Invisible Future, pp. 235–250. McGraw Hill, New York (2001)
3. Bosse, T., Jonker, C.M., Meij, L., van der Sharpanskykh, A., Treur, J.: Specification and Verification of Dynamics in Cognitive Agent Models. In: Nishida, T., et al. (eds.) Proceedings of the Sixth International Conference on Intelligent Agent Technology, IAT 2006., pp. 247–254. IEEE Computer Society Press, Los Alamitos (2006)
4. Bosse, T., Jonker, C.M., van der Meij, L., Treur, J.: A Language and Environment for Analysis of Dynamics by Simulation. International Journal of Artificial Intelligence Tools 16, 435–464 (2007)
5. Brazier, F.M.T., Jonker, C.M., Treur, J.: Compositional Design and Reuse of a Generic Agent Model. Applied Artificial Intelligence Journal 14, 491–538 (2000)
6. Brazier, F.M.T., Jonker, C.M., Treur, J.: Principles of Component-Based Design of Intelligent Agents. Data and Knowledge Engineering 41, 1–28 (2002)
7. Jonker, C.M., Treur, J.: Compositional Verification of Multi-Agent Systems: a Formal Analysis of Pro-activeness and Reactiveness. International Journal of Cooperative Information Systems 11, 51–92 (2002)
8. Reiter, R.: Knowledge in Action: Logical Foundations for Specifying and Implementing Dynamical Systems. MIT Press, Cambridge (2001)
9. Riva, G., Vatalaro, F., Davide, F., Alcañiz, M. (eds.): Ambient Intelligence. IOS Press, Amsterdam (2005)