Formal Design and Analysis of an Ambient Multi-Agent System Model for Medicine Usage Management

Mark Hoogendoorn, Michel Klein, and Jan Treur

Vrije Universiteit Amsterdam, Department of Artificial Intelligence
De Boelelaan 1081a, 1081 HV Amsterdam, The Netherlands
{mhoogen, michel.klein, treur}@cs.vu.nl

Abstract. A formally specified multi-agent-based model for medicine usage management is presented and formally analysed. The model incorporates an intelligent ambient agent model that has an explicit representation of a dynamic system model to estimate the drug concentration in the patient's body by simulation, and is able to analyse whether the patient intends to take the medicine too early or too late.

1. Introduction

The fast developing area of Ambient Intelligence (Aarts *et al*., 2001; Aarts *et al*., 2003; Riva *et al*., 2005) is an interesting application area for agent-based methods. In this paper a formally analysed ambient multi-agent system model for the domain of medicine usage management is presented. The main problem addressed in this health domain is to achieve in a non-intrusive, human-like manner that patients for whom it is crucial that they take medicine regularly, indeed do so. Examples of specific relevant groups include independently living elderly people, psychiatric patients or HIV-infected persons. One of the earlier solutions reported in the literature provides the sending of automatically generated SMS reminder messages to a patient's cell phone at the relevant times; e.g., (Safren *et al*., 2003). A disadvantage of this approach is that patients are disturbed often, even if they do take the medicine at the right times themselves, and that after some time a number of patients start to ignore the messages. More sophisticated approaches make use of a recently developed automated medicine box that has a sensor that can detect whether a medicine is taken from the box, and can communicate this to a server; cf. SIMpill (Green, 2005). This paper explores and analyses possibilities to use automated devices such as an automated medicine box, servers and cell phones as non-human agents, in addition to human agents such as the patient and a supervising doctor. The aim is to obtain a form of medicine usage management that on the one hand achieves that the patient maintains the right amount of medicine, whereas the human factors are also incorporated in an adequate manner, e.g. that a patient is only disturbed if it is really required, thus providing a human-like ambience.

The ambient multi-agent system model for medicine usage management as discussed was formally specified in an executable manner and formally verified using dedicated tools. The system hasn't been actually deployed in a real life situation. The model developed can be used as a blueprint for specific applications in the domain of medicine usage management. The model incorporates an intelligent ambient agent model that has an explicit representation of a dynamic system model to estimate the concentration of the

medicine in the patient's body by simulation, and is able to analyse whether the patient intends to take medicine too early or too late. For actual deployment, the model might have to be extended with other aspects of the patient, e.g. his physical activity and his food consumption, which all has impacts on the drug doses,

In this paper, Section 2 describes the modelling approach. In Section 3 the Multi-Agent System is introduced, whereas Section 4 presents the specification at the multi-agent system level. The specification of the ambient agent is presented in Section 5. Furthermore, Section 6 presents simulation results, Section 7 formal analysis of these results. Finally, Section 8 is a discussion.

2. Modelling Approach

This section briefly introduces the modelling approach used to specify the multi-agent model. Two different aspects are addressed. First of all, the process aspects, and secondly, the information and functionality aspects. Thereafter, the language used for execution of a model and the specification and verification of dynamic properties is briefly introduced.

2.1 Process and Information/Functionality Aspects

Processes are modelled as components in the generic model. A component can either be an active process, namely an *agent*, or a source that can be consulted, which is a *world component*. In order to enable interaction between components, *interaction links* between such components can be specified. From the perspective of information, *interfaces* of components are specified by *ontologies*. Using ontologies, the *functionalities* of components in order to perform the tasks of the component can be specified as well.

2.2 Specification Languages

In order to execute and verify multi-agent models, an expressive language is needed. To this end, the language called TTL is used (Jonker and Treur 2002a; Bosse *et al*., 2006). 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: 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 formal manner in a sorted first-order predicate logic, using quantifiers over time and traces and the usual first-order logical connectives such as $\neg$, $\wedge$, $\vee$, $\Rightarrow$ $\forall$, $\exists$. A special software environment has been developed for TTL, featuring both a Property Editor

for building and editing TTL properties and a Checking Tool that enables formal verification of such properties against a set of (simulated or empirical) traces.

To logically specify simulation models and to execute these models in order to get simulation traces, the language LEADSTO, an executable subset of TTL, is used; cf. (Bosse *et al*., 2007). The basic building blocks of this language are temporal (or 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.

with α and β 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. Overview of the Multi-Agent System

Figure 1 presents an overview of the entire system as considered. Two world components are present in this system: the medicine box, and the patient database; the other components are agents. The top right corner shows the patient, who interacts with the medicine box, and communicates with the patient cell phone. The (ambient) Medicine Box Agent monitors whether medicine is taken from the box, and the position thereof in the box. In case, for example, the patient intends to take the medicine too soon after the previous dose, it finds out that the medicine should not be taken at the moment (i.e., the sum of the estimated current medicine level plus a new dose is too high), and communicates a warning to the patient by a beep. Furthermore, all information obtained by this agent is passed on to the (ambient) Usage Support Agent. All information about medicine usage is stored in the patient database by this agent. If the patient tried to take the medicine too early, a warning SMS with a short explanation is communicated to the cell phone of the patient, in addition to the beep sound already communicated by the Medicine Box Agent.

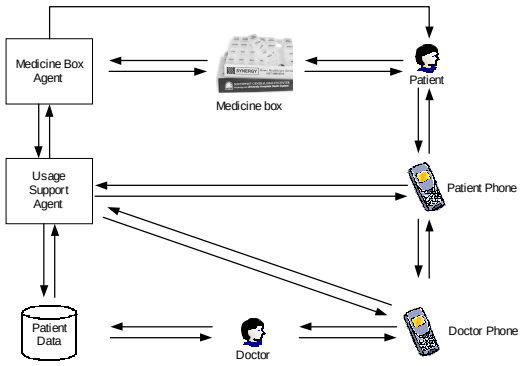

Fig. 1. Multi-Agent System: Overview

On the other hand, in case the Usage Support Agent finds out that the medicine is not taken early enough (i.e., the medicine concentration is estimated too low for the patient and no medicine was taken yet), it can take measures as well. First of all, it can warn the patient by communicating an SMS to the patient cell phone. This is done soon after the patient should have taken the medicine. In case the patient still does not take medicine (for example after a number of hours), the agent can communicate an SMS to cell phone of the

appropriate doctor. The doctor can look into the patient database to see the medicine usage, and in case the doctor feels it is necessary to discuss the state of affairs with the patient, he or she can contact the patient via a call using the doctor cell phone to the patient cell phone.

4. Specification of the Multi-Agent Level

In order for the various components shown in Figure 1 to interact in an appropriate manner, generic temporal rules at the global level are specified that model the interaction between the various components. First of all, Table 1 shows the ontology needed to express such rules. Using the ontology, the following generic temporal rules were specified.

Action Propagation from Agent to World
∀X:AGENT ∀W:WORLD ∀A:ACTION
output(X)|performing_in(A, W) ∧ can_perform_in(X,A,W)
⟶» input(W)|performing(A)

Observation Focus Propagation: from Agent to World
∀X:AGENT ∀W:WORLD ∀I:INFO_EL
output(X)|observation_focus_in(I, W) ∧ can_observe_in(X,I,W)
⟶» input(W)|observation_focus(I)

Observation Result Propagation from World to Agent
∀X:AGENT ∀W:WORLD ∀I:INFO_EL
output(W)|observation_result_from(I, W) ∧ can_observe_in(X,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) ∧ can_communicate_with_about(X,Y,I)
⟶» input(Y)|communicated_from_to(I,X,Y)

5. Specification of the Ambient Agents

The Ambient Agent Model used is based on a combination of the Generic Agent Model GAM described in (Brazier *et al*., 2000), and the generic process control model in (Jonker and Treur, 2002b). To express the agent's internal states and processes, the ontology shown in Table 2 was specified.

Table 1. 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_focus_in(I:INFO_EL, W:WORLD)	observation focus is I in W
observation_result_from(I:INFO_EL, W:WORLD)	observation result from W is I
observed_result_from(I:INFO_EL, W:WORLD)	the observed 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
can_observe_in(X:AGENT, I:INFO_EL,	agent X can observe I within world component W

W:WORLD)	
can_perform_in(X:AGENT, A:ACTION, W:WORLD)	agent X can perform action A within world component W
can_communicate_with_about(X:AGENT , Y:AGENT, I:INFO_EL)	agent X can communicate with agent Y about info element I

Table 2. Ontology Used within the Ambient Agent Model

Predicate	**Description**
belief(I:INFO_EL)	information I is believed
world_fact(I:INFO_EL)	I is a world fact
has_effect(A:ACTION, I:INFO_EL)	action A has effect I
Function to INFO_EL	**Description**
leads_to_after(I:INFO_EL, J:INFO_EL, D:REAL)	state property I leads to state property J after duration D
at(I:INFO_EL, T:TIME)	state property I holds at time T
critical_situation(I:INFO_EL)	the situation concerning I is critical

An example of an expression that can be made by combining elements from this ontology is

belief(leads_to_after(I:INFO_EL, J:INFO_EL, D:REAL))

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. Using this ontology, the functionality of the agent has been specified by generic and domain-specific temporal rules.

5.1 Generic Temporal Rules

The functionality within the Ambient Agent Model has the following generic specifications.

∀X:AGENT, I:INFO_EL, W:WORLD
input(X)|observed_result_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)
∀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))

When the sources are assumed always reliable, the conditions on reliability can be left out. The last rule specifies how a dynamic model that is represented as part of the agent's knowledge can be used by the agent to perform simulation, thus extending its beliefs about the world at different points in time.

For the world components the following generic formal specifications indicate how actions get their effects and how observations provide their results:

∀W:WORLD_COMP ∀A:ACTION ∀I:INFO_EL
input(W)|performing_in(A, W) ∧ internal(W)|has_effect(A,I)
⟶⟶ internal(W)|world_fact(I)

∀W:WORLD_COMP ∀I:INFO_EL
input(W)|observation_focus_in(I, W) ∧ internal(W)|world_fact(I)
⟶⟶ 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))
⟶⟶ output(W)|observation_result_from(not(I), W)

5.2 Domain-Specific Temporal Rules

Domain-specific rules are both shown for the Medicine Box Agent and the Usage Support Agent.

Medicine Box Agent. The Medicine Box Agent has functionality concerning communication to both the patient and the Usage Support Agent. First of all, the observed usage of medicine is communicated to the Usage Support Agent in case the medicine is not taken too early, as specified in MBA1.

MBA1: Medicine usage communication
If the Medicine Box Agent has a belief that the patient has taken medicine from a certain position in the box, and that the particular position contains a certain type of medicine M, and taking the medicine does not result in a too high medicine concentration of medicine M within the patient, then the usage of this type of medicine is communicated to the Usage Support Agent. Formally:

internal(medicine_box_agent)|belief(medicine_taken_from_position(x_y_coordinate(X,Y))) ∧
internal(medicine_box_agent)|belief(medicine_at_location(x_y_coordinate(X, Y), M)) ∧
internal(medicine_box_agent)|belief(medicine_level(M, C)) ∧
max_medicine_level(maxB) ∧ dose(P) ∧ C + P ≤ maxB
$\rightarrow\!\!\!\!\rightarrow_{0,0,1,1}$ output(medicine_box_agent)|communication_from_to(
medicine_used(M), medicine_box_agent, usage_support_agent)

In case medicine is taken out of the box too early, a warning is communicated by a beep and the information is forwarded to the Usage Support Agent (MBA2 and MBA3).

MBA2: Too early medicine usage prevention
If the Medicine Box Agent has the belief that the patient has taken medicine from a certain position in the box, that this position contains a certain type of medicine M, and taking the medicine results in a too high medicine concentration of medicine M within the patient, then a warning beep is communicated to the patient.

internal(medicine_box_agent)|belief(medicine_taken_from_position(x_y_coordinate(X,Y))) ∧
internal(medicine_box_agent)|belief(medicine_at_location(x_y_coordinate(X, Y), M)) ∧
internal(medicine_box_agent)|belief(medicine_level(M, C)) ∧
max_medicine_level(maxB) ∧ dose(P) ∧ C + P > maxB
$\rightarrow\!\!\!\!\rightarrow_{0,0,1,1}$ output(medicine_box_agent)|communication_from_to(
sound_beep, medicine_box_agent, patient)

MBA3: Early medicine usage communication
If the Medicine Box Agent has a belief that the patient was taking medicine from a certain position in the box, and that the particular position contains a certain type of medicine M, and taking the medicine would result in a too high concentration of medicine M within the patient, then this is communicated to the Usage Support Agent.

internal(medicine_box_agent)|belief(medicine_taken_from_position(x_y_coordinate(X,Y))) ∧
internal(medicine_box_agent)|belief(medicine_at_location(x_y_coordinate(X, Y), M)) ∧
internal(medicine_box_agent)|belief(medicine_level(M, C)) ∧
max_medicine_level(maxB) ∧ dose(P) ∧ C + P > maxB
$\rightarrow\!\!\!\!\rightarrow_{0,0,1,1}$ output(medicine_box_agent)|communication_from_to(
too_early_intake_intention, medicine_box_agent, usage_support_agent)

Usage Support Agent. The Usage Support Agent's functionality is described by three sets of temporal rules. First, the agent maintains a dynamic model for the concentration of medicine in the patient over time in the form of a belief about a *leads to* relation.

USA1: Maintain dynamic model
The Usage Support Agent believes that if the medicine level for medicine M is C, and the usage effect of the medicine is E, then after duration D the medicine level of medicine M is C+E minus G*(C+E)*D with G the decay value.

internal(usage_support_agent)|belief(leadsto_to_after(
medicine_level(M, C) ∧ usage_effect(M, E) ∧ decay(M, G),
medicine_level(M, (C+E) - G*(C+E)*D), D)

In order to reason about the usage information, this information is interpreted (USA2), and stored in the database (USA3).

USA2: Interpret usage

If the agent has a belief concerning usage of medicine M and the current time is T, then a belief is generated that this is the last usage of medicine M, and the intention is generated to store this in the patient database.

internal(usage_support_agent)|belief(medicine_used(M)) ∧
internal(usage_support_agent)|belief(current_time(T))

→→0,0,1,1
internal(usage_support_agent)|belief(last_recorded_usage(M, T) ∧
internal(usage_support_agent)|intention(store_usage(M, T))

USA3: Store usage in database

If the agent has the intention to store the medicine usage in the patient database, then the agent performs this action.

internal(usage_support_agent)|intention(store_usage(M, T))

→→0,0,1,1 output(usage_support_agent)|performing_in(store_usage(M, T), patient_database)

Finally, temporal rules were specified for taking the appropriate measures. Three types of measures are possible. First, in case of early intake, a warning SMS is communicated (USA4). Second, in case the patient is too late with taking medicine, a different SMS is communicated, suggesting to take the medicine (USA5). Finally, when the patient does not respond to such SMSs, the doctor is informed by SMS (USA6).

USA4: Send early warning SMS

If the agent has the belief that an intention was shown by the patient to take medicine too early, then an SMS is communicated to the patient cell phone that the medicine should be put back in the box, and the patient should wait for a new SMS before taking more medicine.

internal(usage_support_agent)|belief(too_early_intake_intention)

→→0,0,1,1
output(usage_support_agent)|communication_from_to(put_medicine_back_and_wait_for_signal,
usage_support_agent, patient_cell_phone)

USA5: SMS to patient when medicine not taken on time

If the agent has the belief that the level of medicine M is C at the current time point, and the level is considered to be too low, and the last message has been communicated before the last usage, and at the current time point no more medicine will be absorbed by the patient due to previous intake, then an SMS is sent to the patient cell phone to take the medicine M.

internal(usage_support_agent)|belief(current_time(T3)) ∧
internal(usage_support_agent)|belief(at(medicine_level(M, C), T3)) ∧
min_medicine_level(minB) ∧ C < minB ∧
internal(usage_support_agent)|belief(last_recorded_usage(M, T)) ∧
internal(usage_support_agent)|belief(last_recorded_patient_message_sent(M, T2)) ∧
T2 < T ∧ usage_effect_duration(UED) ∧ T3 > T + UED

→→0,0,1,1 output(usage_support_agent)|communication_from_to(
sms_take_medicine(M), usage_support_agent, patient_cell_phone)

USA6: SMS to doctor when no patient response to SMS

If the agent has the belief that the last SMS to the patient has been communicated at time T, and the last SMS to the doctor has been communicated before this time point, and furthermore, the last recorded usage is before the time point at which the SMS has been sent to the patient, and finally, the current time is later than time T plus a certain delay parameter for informing the doctor, then an SMS is communicated to the cell phone of the doctor that the patient has not taken medicine M.

internal(usage_support_agent)|belief(last_recorded_patient_message_sent(M, T)) ∧
internal(usage_support_agent)|belief(last_recorded_doctor_message_sent(M, T0)) ∧
internal(usage_support_agent)|belief(last_recorded_usage(M, T2)) ∧
internal(usage_support_agent)|belief(current_time(T3)) ∧
T0 < T ∧ T2 < T ∧ max_delay_after_warning(DAW) ∧ T3 > T + DAW

→→0,0,1,1
output(usage_support_agent)|communication_from_to(sms_not_taken_medicine(M),
usage_support_agent, doctor_cell_phone)

6. Simulation Results

In order to show how the above presented system functions, the system has been implemented in a dedicated software environment that can execute such specifications (Bosse *et al*., 2007)]. This section presents some of the simulation results. First, the stochastic model to generate patient scenarios is introduced; thereafter an example simulation trace of the system is shown and explained.

6.1. Stochastic Patient Model

To enable creation of simulations, a patient model is used that simulates the behaviour of the patient in a stochastic manner. The model specifies four possible behaviours of the patient, each with its own probability: (1) too early intake, (2) correct intake (on time), (3) responding to an SMS warning that medicine should be taken, and (4) responding to a doctor request by phone. Based upon such probabilities, the entire behaviour of the patient regarding medicine usage can be simulated. In the following simulations, values of respectively 0.1, 0.8, 0.9 and 1.0 have been used.

6.2. Example Trace

Figure 2 shows the medicine level over time in an example of a simulation trace as estimated by the agent based on its dynamic model. Here the x-axis represents time whereas the y-axis represents the medicine level. Note that in this case, the minimum level of medicine within the patient is set to 0.35 whereas the maximum level is 1.5. These numbers are based on the medicine half-life value, that can vary per type of medicine. In the trace in Figure 2, it can be seen that the patient initially takes medicine at the appropriate time, this is done by performing an action:

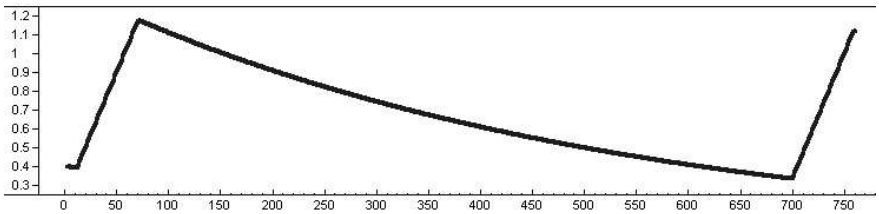

Fig. 2. Medicine level over time

output(patient)|performing_in(take_medicine_from_position(x_y_coordination(1,1)), medicine_box)

This information is stored in the patient database:

input(patient_database)|performing_in(store_usage(recorded_usage(hiv_slowers, 8)), patient_database)

Resulting from this medicine usage, the medicine level increases, as can be seen in Figure 3. In this simulation, the medicine takes 60 minutes to be fully absorbed by the patient. Just 240 minutes after taking this medicine, the patient again takes a pill from the medicine box. This is however too early, and as a result, the Medicine Box Agent communicates a warning beep, which is received by the patient:

input(patient)|communicated_from_to(sound_beep, medicine_box_agent, patient)

The patient does not take the medicine, and waits for an SMS, as the patient is instructed to do. The SMS is received a little while later, stating that the patient should wait with taking the medicine until a new message is sent.

input(patient)|communicated_from_to(put_medicine_back_and_wait_for_signal, patient_cell_phone, patient)

The patient awaits this message, which is sent after the medicine level is considered to be low enough such that medicine can be taken. The Usage Support Agent therefore communicates an SMS to the patient cell phone:

output(usage_support_agent)|communication_from_to(sms_take_medicine(hiv_slowers), usage_support_agent, patient_cell_phone)

This message is received by the patient:

input(patient)|communicated_from_to(sms_take_medicine(hiv_slowers), patient_cell_phone, patient)

The patient does however not respond to the SMS, and does not take its medicine. The usage support agent responds after 60 minutes, and sends an SMS to the doctor of the patient:

input(doctor)|communicated_from_to(sms_not_taken_medicine(hiv_slowers), usage_support_agent, doctor)

The doctor immediately calls the patient, and makes sure the patient understands that it is essential to take the medicine immediately. As a result, the patient takes the medicine:

output(patient)|performing_in(take_medicine_from_position(x_y_coordination(1,1)), medicine_box)

As can be seen in Figure 3, during the entire simulation the patient medicine level never exceeds the maximum level (1.6), and never goes below the minimum required level (0.35).

An example graph of the medicine level of a patient not using the support system presented in this section is shown in Figure 3. Note that this is a simulation over a longer time period (3000 minutes). As can be seen, the intake interval fluctuates a lot and the medicine level does not always stay above the desired level of 0.35.

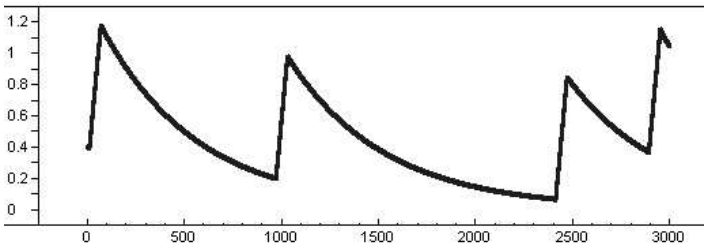

Fig. 3. Medicine level without support system

7. Verification of the System

Requirements that should hold for all simulation traces of the type of system as designed can be specified as dynamic properties. The language TTL (Jonker and Treur 2002a; Bosse et al., 2006)) is used to specify such properties. The dedicated software environment for TTL allows automated verification of these properties against generated traces. To perform such validation, first the aim of the system is to be defined. A human-like ambience system has as goal to support a human's wellbeing and functioning. Here 'supporting' is a broad term which can take many different forms. However, in practice often one of the following types of properties are at issue.

(a) The system tries to keep the value of some state property or variable within a preferred or required range. For example, the medicine level in the blood of a patient should be high enough to be effective, but not too high to cause undesired side effects.

(b) The system tries to prevent a undesired situation to happen. For example, a system tries to prevent that a potential criminal performs a criminal act.

The validation of the usefulness of a human ambience-like system is related to its aims. In general, its validation consists of comparing a scenario where a human ambience system is present with a similar scenario without such a system. The generic hypothesis is that with a human-like ambience system the situation is more often within the preferred range, respectively that undesired situations less frequently happen. This hypothesis can be refined in questions about the scenarios. For the first task the following questions can be asked:

1. **Occurrence of violation**: Is the value of the property in always within the safe boundaries?
2. **Number of violations**: What is the average number of violations of the safe boundaries per scenario?
3. **Duration of violation**: For how long is the property on average not within the safe boundaries?

If a distinction is made between preferred values of a property and required values, these questions can be asked for both types of boundaries. For the second type of task the duration is not relevant, so the only two relevant questions are:

1. **Occurrence of violation**: Does the undesired situation occur?
2. **Number of violations**: How often does the undesired situation occur on average per scenario?

In the formalization of these questions the first three questions are considered a special case of the last two: i.e., the violations of the safe boundaries are seen as a specification of an undesired situation. The consequence of this is that all dynamic properties can be formulated at a general level and the only domain specific knowledge is the specification of the undesired situation. In the formalization P stands for the property of interest, M for the traces for a specific scenario, has_value(P,V) means that V is the value of P, TU stands for the upper boundary of the property, TL for its lower boundary, and S for the undesired situation. Note that for the duration and number of violations, we have to specify a minimum number that should hold. Here $\sum$ case(p, 1, 0) is a special construct in the language that calculates the sum of timepoints for which a state holds. The generic rules are:

1. **Occurrence of violation**:
   $\forall$M:TRACE $\forall$T:TIMEPOINT: state(M, T) |= $\neg$S
2. **Number of violations**:
   $\forall$M:TRACE: $\forall$T1, T2:TIMEPOINT: $\sum$ case(T1$\leq$T & T $\leq$T2 & state(M, T) |= $\neg$S $\wedge$ state(M, T+1) |= S, 1, 0) > MIN_OCCURANCES
3. **Duration of violation**:
   $\forall$M:TRACE $\forall$T1, T2:TIMEPOINT: $\sum$ case(T1$\leq$T & T $\leq$T2 & state(M, T) |= S, 1, 0) > MIN_DURATION

In case of a value that should be within a specific range, the undesired situation S is specified as follows:

$\exists$V:VALUE: has_value(P,V) $\wedge$ (V > TU $\vee$ V < TL)

This has been applied to the medicine box case, where the value of the property 'medicine_level' was checked. The values 0.3 and 1.5 were used for the minimal and maximal required level. The system was validated on 10 traces with the length of one week. Five traces simulated a scenario with support of the system and five a scenario without. In the scenario with support, there were less violations of the required medicine level: on average 1.8 violations, versus 4.2 in the unsupported scenarios. In addition, those violations were significantly shorter: the average time that a patient has a wrong medicine level in his blood is 1.9 hours per week in the supported case, versus 16.3 hours per week in the unsupported case. This clearly illustrates the improvement of the situation of the patient that the system provides.

8. Discussion

In this paper, a multi-agent system model was presented that supports the users of medicine in taking their medicine at the appropriate time. The system has been specified using a formal modelling approach which enables the specification of both quantitative as well as qualitative aspects (Jonker and Treur 2002a; Bosse *et al*., 2006). To specify the model, both generic and domain specific temporal rules have been used, enabling reuse of the presented model. Evaluation of the model has been conducted by means of simulation runs using a stochastic model for patients. The simulation results have been evaluated using formal verification techniques, whereby it was shown that usage of the system indeed results in better medicine intake adherence.

The presented multi-agent system model fits well in the recent developments in Ambient Intelligence (Aarts *et al*., 2001, 2003; Riva *et al*., 2005). Furthermore, it also shows that multi-agent system technology can be of great benefit in health care applications, as also acknowledged in (Moreno and Nealon, 2004). More approaches to support medicine usage of patients have been developed. Both in (Greene, 2005) as well as (Floerkemeier and Siegemund, 2003) models are presented that do not simply always send an SMS that medicine should be taken such as proposed by (Safren *et al*., 2003). Both approaches only send SMS messages in case the patient does not adhere to the prescribed usage. The model presented in this paper however adds an additional dimension to such a support system, namely the explicit representation and simulation of the estimated medicine level inside the patient. Having such an explicit model enables the support agent to optimally support the patient.

References

Aarts, E.; Collier, R.; van Loenen, E.; Ruyter, B. de (eds.) (2003). *Ambient Intelligence. Proc. EUSAI 2003*. Lecture Notes in Computer Science, vol. 2875. Springer Verlag, 2003, pp. 432.

Aarts, E., Harwig, R. , and Schuurmans, M. (2001), Ambient Intelligence. In: P. Denning (ed.), *The Invisible Future*, McGraw Hill, pp. 235-250.

Bosse, T., Jonker, C.M., Meij, L. van der, Sharpanskykh, A, and Treur, J. (2006). 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'06*. IEEE Computer Society Press, 2006, pp. 247-254.

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*. To appear, 2007. Shorter, preliminary version in: Eymann, T., et al. (eds.), *Proc. of MATES'05*. Lecture Notes in Artificial Intelligence, vol. 3550. Springer Verlag, 2005, pp. 165-178.

Bosse, T., Memon, Z.A., and Treur, J. (2007). A Two-level BDI-Agent Model for Theory of Mind and its Use in Social Manipulation. In: P. Olivier, C. Kray (eds.), *Proceedings of the Artificial and Ambient Intelligence Conference, AISB'07*. AISB Publications, 2007, pp. 335-342.

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.

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.

Floerkemeier, C., Siegemund, F. (2003). Improving the Effectiveness of Medical Treatment with Pervasive Computing Technologies. *Workshop on Ubiquitous Computing for Pervasive Healthcare Applications at Ubicomp 2003*, Seattle, October 2003

Green D.J. (2005). Realtime Compliance Management Using a Wireless Realtime Pillbottle – A Report on the Pilot Study of SIMPILL. In: *Proc. of the International Conference for eHealth, Telemedicine and Health, Med-e-Tel'05*, 2005, Luxemburg.

Jonker, C.M., and Treur, J. (2002a). Compositional Verification of Multi-Agent Systems: a Formal Analysis of Pro-activeness and Reactiveness. *International Journal of Cooperative Information Systems*, vol. 11, 2002, pp. 51-92.

Jonker, C.M., and Treur, J. (2002b). A Compositional Process Control Model and its Application to Biochemical Processes. *Applied Artificial Intelligence Journal*, vol. 16, 2002, pp. 51-71.

Moreno, A., Nealon, J.L. (eds.), *Applications of Software Agent Technology in Health Care Domain*. Birkhäuser Basel, 2004.

Riva, G., F. Vatalaro, F. Davide, M. Alcañiz (eds). (2005). *Ambient Intelligence*. IOS Press, 2001.

Safren, S.A., Hendriksen, E.S., Desousa, N., Boswell, S.L., Mayer, K.H., (2003). Use of an on-line pager system to increase adherence to antiretroviral medications. In: *AIDS CARE*, vol. 15, pp. 787 – 793