# An Intelligent Agent Model with Awareness of Workflow Progress

Fiemke Both[a], Mark Hoogendoorn[a], Andy van der Mee[b], Jan Treur[a] and Michael de Vos[b]
[a]*Department of Artificial Intelligence, Vrije Universiteit Amsterdam, De Boelelaan 1081a, 1081 HV Amsterdam, The Netherlands*
[b]*Force Vision Lab, Barbara Strozzilaan 362a, 1083 HN Amsterdam, The Netherlands*

**Abstract.** To support human functioning, ambient intelligent agents require knowledge about the tasks executed by the human. This knowledge includes design-time information like: (i) the goal of a task and (ii) the alternative ways for a human to achieve that goal, as well as run-time information such as the choices made by a human during task execution. In order to provide effective support, the agent must know exactly what steps the human is following. However, if not all steps along the path can be observed, it is possible that the agent cannot uniquely derive which path the human is following. Furthermore, in order to provide timely support, the agent must observe, reason, conclude and support within a limited period of time. To deal with these problems, this paper presents a generic focused reasoning mechanism to enable a guided selection of the path which is most likely followed by the human. This mechanism is based upon knowledge about the human and the workflow to perform the task. In order to come to such an approach, a reasoning mechanism is adopted in combination with the introduction of a new workflow representation, which is utilized to focus the reasoning process is an appropriate manner. The approach is evaluated by means of an extensive case study.

Keywords: ambient agent, workflow model, task progress, awareness, model-based reasoning

## 1. Introduction

Nowadays, a number of research fields focus on technology that can support humans in their everyday activities. These research fields include e.g. Ambient Intelligence [2, 3], Ubiquitous Computing [16], and Human Aware Computing [11]. The support provided to the human could for instance take the form of a personal assistant agent that monitors the activities of the human and supports the execution thereof when necessary.

In order for the support during such activities to be possible, the agent should not only have information about the task being performed and what kind of support can be given, but should also be aware of what the human is doing at particular time points. This awareness enables the ambient agent system to provide dedicated support based on the task at hand and the state of the human, regardless of whether the support is aimed at optimal task performance or at optimally supporting the human. For example, the ambient agent system can decide not to disturb the human during a very intensive task, or provide task dependent support in case a task is taking much longer than it should. A registration system that on the fly records at which points in time a certain (sub)task is started and at which time it is finished is not always available. Direct communication about this with the human often would disturb the human's process. One of the challenges for ambient intelligent agents is to obtain such information about a human's progress by an analysis based on the observation information that is available and a workflow model.

Often workflow models are nondeterministic, and the human's choice for a certain branch may depend on circumstances and on his or her preferences, which may be unknown. Therefore it may not be clear at forehand which of the branches of such a workflow model are actually followed, and at least some observation information is needed.

In this paper an ambient agent model is presented that is able to perform model-based analysis using available workflow models and available observation information, to obtain more detailed information on the human's progress in task execution. The (possibly partial) observation information may come from sensors detecting location, posture, movements of the user, or when a computer system is used. The model incorporates formally specified model-based reasoning methods as presented in [4]. As part of these reasoning methods, a generic focussing approach is applied that uses background knowledge to focus and control this reasoning process. This allows the ambient agent to limit the amount of processing and information within itself and at the same time provides heuristics to use additional knowledge for determining the most likely path. An important aspect of the approach developed is that model-based reasoning methods, as known within Artificial Intelligence and usually applied to causal models, are now applied to workflow models. This allows the addition of more information that can be utilized to guide the reasoning process. To this end, a representation format for workflow models was chosen that unifies with usual formats for model-based reasoning.

This paper is organized as follows. In Section 2 the workflow representation is discussed. Thereafter, Section 3 introduces the formal modelling language that has been used, whereas Section 4 the reasoning approach is presented. Section 5 introduces the focusing mechanisms. In Section 6 simulation results are shown, which are verified in Section 7. Section 8 describes related work and finally, Section 9 is a discussion.

## 2. Workflow Representation

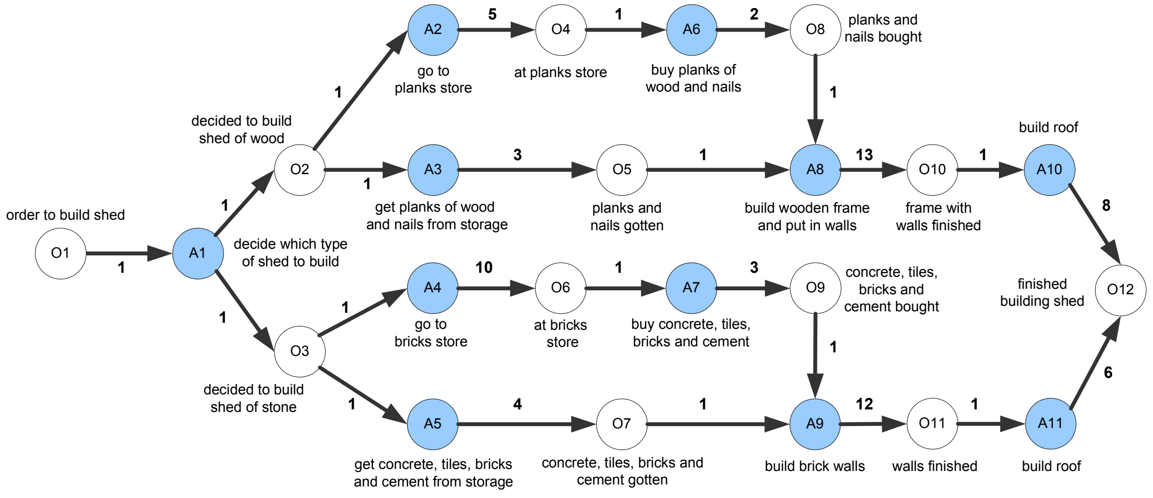
In order to represent the workflow, first of all, a graphical representation is used to represent the dependencies between the various states in the workflow. Thereafter, an ontology is introduced that specifies additional information about the states within the workflow.

### *2.1. Graphical Representation*

In the graphical representation of the workflow, states are represented by means of nodes (i.e. circles), and the transitions between states by means of arrows. The nodes within the workflow can be either grey, representing activities, or white, representing the result of an activity. The arrows are labeled with the time it takes to move from the state of the source of the arrow to the next state. When the source state is an activity, the label represents the time it takes to execute the activity. Figure 1 shows an example graphical representation of building a shed.

### *2.2. Specification of Additional Information*

As already stated in the introduction, in order to guide the reasoning process more effectively, more information about the states in the workflow is required. Therefore, an ontology is presented here that specifies these additional elements. First of all, the predicates for the representation of the workflow itself are specified as shown in Table 1. Note that a state can occur at a specific time point; this represents the starting time of an activity if the state is an activity. The duration of the activity is represented in the predicate leads_to_after.



order to build shed
O1
1
A1
decide which type of shed to build
1
O2
decided to build shed of wood
1
O3
decided to build shed of stone
1
A2
go to planks store
5
O4
at planks store
1
A6
buy planks of wood and nails
2
O8
planks and nails bought
1
A3
get planks of wood and nails from storage
3
O5
planks and nails gotten
1
A8
build wooden frame and put in walls
13
O10
frame with walls finished
1
A10
build roof
8
O12
finished building shed
1
A4
go to bricks store
10
O6
at bricks store
1
A7
buy concrete, tiles, bricks and cement
3
O9
concrete, tiles, bricks and cement bought
1
1
A5
get concrete, tiles, bricks and cement from storage
4
O7
concrete, tiles, bricks and cement gotten
1
A9
build brick walls
12
O11
walls finished
1
A11
build roof
6

**Figure 1**. Example workflow: building a shed

**Table 1.** Sorts and predicated for workflow

| Sort / Predicate | Explanation |
|---|---|
| STATE | An identifier of a state |
| TIME | A time point |
| DURATION | A duration |
| leads_to_after: STATE x STATE x DURATION | The first state specified leads to the second state in the workflow with a delay of DURATION |
| at: STATE x TIME | The state occurs at time point TIME |
| is_activity: STATE | A state describing an activity (i.e. a grey node) |

**Table 2.** Extended workflow representation

| Sort / Predicate | Explanation |
|---|---|
| RESOURCE | A resource identifier |
| COMPETENCE_TYPE | The type of competence |
| COMPETENCE_TYPE_LEVEL | The level of a certain competence type, which can a real number on the interval [0,1] |
| COMPETENCE_LEVEL | An overall competence level for a state (i.e. combining resources with human competences), represented by a real number on the interval [0,1] |
| RESOURCE_CONDITION | The condition of the resource, indicates by a real number on the interval [0,1] |
| requires_competence_type: RESOURCE x COMPETENCE_TYPE | A resource requires a certain competence type in order to control it. |
| resulting_competence_level_for: RESOURCE x COMPETENCE_TYPE_LEVEL x RESOURCE_CONDITION x STATE x COMPETENCE_LEVEL | Given that the resource can be controlled, a certain competence level for the required competence type in combination with the condition of the resource results in a certain competence level for a state. |
| requires_competence_level: STATE x COMPETENCE_LEVEL | A state requires a certain competence level. |

Besides this basic information, additional information can be specified related to the workflow, as shown in Table 2. The basic intuition behind the predicates introduced in the table is the following. First of all, in order for a human to use a certain resource, he needs to be able to control it (i.e. have the right competence type). For instance in the case of building a shed, in order to hammer a nail into a plank, you need to have the competence to use a hammer. In case this competence is indeed present, the resulting effectiveness of the combination of the resource and the human for a certain state in the workflow depends on two factors: (1) the level the human has of the required competence (e.g. an experienced house builders versus an incidental house builder), and the condition of the resource (is the hammer any good). The resulting competence level can then be matched with the required competence level of the state. Of course, a human can also have a competence for a particular state without using a specific resource.

## 3. Formal Languages Used

In order to execute and verify human-like ambience models, the expressive Temporal Trace Language (TTL) is used [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* $\gamma$ over state ontology Ont and time frame T is a mapping $\gamma : T \rightarrow$ STATES(Ont), i.e., a sequence of states $\gamma_t$ ($t \in T$) 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 $\gamma$ over state ontology Ont, the state in $\gamma$ at time point t is denoted by state($\gamma$, 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($\gamma$, t) |= p denotes that state property p holds in trace $\gamma$ 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 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. Especially the possibility to use variables and quantifiers, also over numbers, makes TTL more useful in practical applications, compared to, for example, propositional and modal temporal languages.

### *3.1. Executable Format.*

To specify simulation models and to execute these models, the language LEADSTO [8], an executable sublanguage of TTL, is used. The basic building blocks of this language are causal relations of the format $\alpha \twoheadrightarrow_{e, f, g, h} \beta$, which means:

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

where $\alpha$ and $\beta$ 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. Also for the language LEADSTO, the possibility to use variables, (e.g., over numbers), makes it more useful for practical application, compared to, for example, propositional or qualitative causal modelling languages. For more information about the hybrid quantitative and qualitative approach of the TTL and LEADSTO languages, see [7, 8].

## 4. Model-Based Reasoning

A first step in deriving what task an agent is currently doing is by specifying the workflow of the agent. The workflow is represented in the ambient agent as a collection of rules that express which state follows which state in the form of beliefs representing the informational state of the agent. The term belief has been used here to make explicit that this knowledge can have a degree of uncertainty. Both the beliefs about the states (since there is usually no complete information, these states do not hold with complete certainty) and the knowledge about the relations (since these are assumptions of the relations the agent considers correct in the domain). The set of rules are of the form

belief(leads_to_after(state1, state2, duration)).

When the ambient agent believes that the human has performed a specific subtask at some time point, it has a belief of the form belief(at(state, time)). A set of generic reasoning methods has been developed to derive more beliefs based on the leads_to_after rules and one or more beliefs about active states; cf. [4]. Below is a summary of these methods.

### *4.1. Forward Reasoning Methods*

Reasoning methods that reason forward in time are often used to make predictions on future states, or on making an estimation of the current state based on information acquired in the past. The first reasoning method is one that occurs in the literature in many variants in different contexts and under different names. This varies from, for example, computational (numerical) simulation based on difference or differential equations, qualitative simulation, causal reasoning, execution of executable temporal logic formulae, and forward chaining in rule-based reasoning, to generation of traces by transition systems and finite automata. The basic specification of this reasoning method can be expressed as follows (note that for the sake of clarity the subscript below the LEADSTO arrow has been omitted since the same values for e,f,g, and h are used namely 0,0,1,1). The term INFO_EL should in this case be seen as the information type consisting of the states in the workflow.

**Belief Generation based on Positive Forward Simulation**

If it is believed that I holds at T and that I leads to J after duration D, then it is believed that J holds after D.

∀I,J:INFO_EL ∀D:REAL ∀T:TIME
belief(at(I, T)) ∧ belief(leads_to_after(I, J, D))
↠ belief(at(J, T+D))

If it is believed that I1 holds at T and that I2 holds at T, then it is believed that I1 and I2 holds at T.

belief(at(I1,T)) ∧ belief(at(I2, T))
↠ belief(at(and(I1, I2), T))

Note that, if the initial beliefs are assumed correct, belief correctness holds for leads to beliefs, and positive forward correctness of leads to relationships holds, then all beliefs generated in this way are correct. A second way of belief generation by forward simulation addresses the propagation of negations. This is expressed as follows for single source causal relations.

**Belief Generation based on Single Source Negative Forward Simulation**

If it is believed that I does not hold at T and that I leads to J after duration D, then it is believed that J does not hold after D.

∀I,J:INFO_EL ∀D:REAL ∀T:TIME
belief(at(not(I), T)) ∧ belief(leads_to_after(I, J, D))
↠ belief(at(not(J), T+D)))

If it is believed that I1 (resp. I2) does not hold at T, then it is believed that I1 and I2 does not hold at T.

belief((at(not(I1),T))) ↠ belief(at(not(and(I1, I2)), T))
belief(at(not(I2),T)) ↠ belief(at(not(and(I1, I2)), T))

Note that this only provides correct beliefs when the initial beliefs are assumed correct, belief correctness holds for leads to beliefs, and single source negative forward correctness holds for the leads to relationships.

When there are multiple sources leading to information element J and all of the sources are believed to be not true, it can be assumed that J is also not true.

**Belief Generation based on Multiple Source Negative Forward Simulation**

If for any J and time T, for every I that is believed to lead to J after some duration D, it is believed that I does not hold before duration D, then it is believed that J does not hold.

∀I,J:INFO_EL ∀D:REAL ∀T:TIME
∀I, D [ belief(leads_to_after(I, J, D)) →
belief(at(not(I), t-D) ] ↠ belief(at(not(J), T))

If it is believed that I1 (resp. I2) does not hold at T, then it is believed that I1 and I2 does not hold at T.

belief(at(not(I1),T)) ↠ belief(at(not(and(I1, I2)), T))
belief(at(not(I2),T)) ↠ belief(at(not(and(I1, I2)), T))

This provides correct beliefs when the initial beliefs are assumed correct, belief correctness holds for leads to beliefs, and multiple source negative forward correctness holds for the leads to relationships.

### *4.2. Backward reasoning methods*

The basic specification of a backward reasoning method is based on the basic Modus Tollens argument and is specified as follows.

**Belief Generation based on Modus Tollens Inverse Simulation**

If it is believed that J does not hold at T and that I leads to J after duration D, then it is believed that I does not hold before duration D.

∀I,J:INFO_EL ∀D:REAL ∀T:TIME
belief(at(not(J), T)) ∧ belief(leads_to_after(I, J, D))
↠ belief(at(not(I), T-D))

If it is believed that not I1 and I2 holds at T and that I2 (resp. I1) holds at T, then it is believed that I1 (resp. I2) does not hold at T.

belief(at(not(and(I1, I2), T)) ∧ belief(at(I2, T))
↠ belief(at(not(I1), T))
belief(at(not(and(I1, I2), T)) ∧ belief(at(I1, T))
↠ belief(at(not(I2), T))

As another option, an abductive causal reasoning method can be internally represented in a simplified form as follows.

**Belief Generation based on Simple Abduction**

If it is believed that J holds at T and that I leads to J after duration D, then it is believed that I holds before duration D.

∀I,J:INFO_EL ∀D:REAL ∀T:TIME
belief(at(J, T)) ∧ belief(leads_to_after(I, J, D))
↠ belief(at(I, T-D))

If it is believed that I1 and I2 holds at T, then it is believed that I1 holds at T and that I2 holds at T.

$$\begin{aligned}&\text{belief(at(and(I1, I2), T))}\\&\qquad\twoheadrightarrow \text{belief(at(I1,T))} \wedge \text{belief(at(I2, T))}\end{aligned}$$

Note that this method does not necessarily lead to the truth, since I does not need to be the state that led to J. However, I is derived when J is believed to be true, because the goal of the reasoning process is to consider all possible options.

The same principle can be applied for multiple effects:

**Belief Generation based on Multiple Effect Abduction**

If for any I and time T, for every J for which it is believed that I leads to J after some duration D, it is believed that J holds after duration D, then it is believed that I holds at T.

$$\begin{aligned}&\forall \text{I:INFO\_EL}\ \forall \text{T:TIME}\\&\forall \text{J [belief(leads\_to\_after(I, J, D))} \rightarrow \text{belief(at(J, T+D)) ]}\\&\qquad\twoheadrightarrow \text{belief(at(I, T))}\end{aligned}$$

If it is believed that I1 and I2 holds at T, then it is believed that I1 holds at T and that I2 holds at T.

$$\begin{aligned}&\text{belief(at(and(I1, I2), T))}\\&\qquad\twoheadrightarrow \text{belief(at(I1,T))} \wedge \text{belief(at(I2, T))}\end{aligned}$$

The last reasoning method is context-supported abduction, which uses a conjunction of two states leading to a third state to derive one of the conjunction states.

**Belief Generation based on Context-Supported Abduction**

If it is believed that J holds at T and that I2 holds at T and that I1 and I2 leads to J after duration D, then it is believed that I1 holds before duration D.

$$\begin{aligned}&\forall \text{I,J:INFO\_EL}\ \forall \text{D:REAL}\ \forall \text{T:TIME}\\&\text{belief(at(J, T))} \wedge \text{belief(at(I2, T-D))} \wedge\\&\text{belief(leads\_to\_after(and(I1, I2), J, D))}\\&\qquad\twoheadrightarrow \text{belief(at(I1, T-D))}\end{aligned}$$

If it is believed that I1 and I2 holds at T, then it is believed that I1 holds at T and that I2 holds at T.

$$\begin{aligned}&\text{belief(at(and(I1, I2), T))}\\&\qquad\twoheadrightarrow \text{belief(at(I1,T))} \wedge \text{belief(at(I2, T))}\end{aligned}$$

## 5. Focused Reasoning

When the ambient agent uses these methods to derive more beliefs, the number of beliefs can quickly get out of control. For example, in a workflow with one starting node and several different paths that can be followed after that first node, the ambient agent could derive beliefs about all nodes in the workflow, because there is no reason to select one path and not the other. Therefore, this section introduces a selection mechanism that can control which beliefs are derived and which beliefs are not. For the belief generation reasoning methods this means that an antecedent is added stating which selection criteria must be met. This idea is shown for the reasoning method *positive forward simulation*.

If the belief that I holds at T was selected and it is believed that I leads to J after duration D, and selection criterion s1 holds, then the belief that J holds after D is selected.

$$\begin{aligned}&\forall \text{I,J:INFO\_EL}\ \forall \text{D:REAL}\ \forall \text{T:TIME}\\&\text{selected\_belief(at(I, T))} \wedge \text{belief(leads\_to\_after(I, J, D))}\\&\wedge\ \text{s1}\ \twoheadrightarrow \text{selected\_belief(at(J, T+D))}\end{aligned}$$

Selection criteria needed for controlled belief generation can be specified for example by assuming that the ambient agent has knowledge about which beliefs are relevant, expressed by a predicate in_focus. If this assumption is made, then the selection criterion s1 in the example above can be expressed as in_focus(I), where I is the property for which a belief is considered. The general idea is that if a belief can be generated, it is selected (only) when it is in focus.

In the following sections a generic focusing mechanism is proposed that uses path information, observations and softer criteria such as competence level and use of resources to calculate the likelihood of paths in the workflow. This mechanism is developed as a generic method that can be applied to reasoning about workflows in various domains. Resources for example, used for the execution of a task are in our case study tools for building a shed. In other domains these resources and required competences can be the use of computer software and the skills required for that, or consulting a manual.

First, the path generation method is explained. Then, the focusing mechanism proposed is briefly outlined. Thereafter more details are given how to calculate the specific parameters within the focusing mechanism.

### *5.1. Path Generation*

The first step is to determine all possible paths in the workflow. States can occur in more than one path, because each path starts with the first state and ends with the last state of the workflow. Notice that in the focusing part of the reasoning, the reasoning about these paths is relatively lightweight because the time points within the paths are not considered yet. The paths are generated using the following set of rules. The first rule finds the first state of a workflow, the second rule than keeps adding a state to the path that the last state of the temporary path leads to. The last rule finalizes the path when the last state of a workflow has been added.

**P3: Generate initial path**

If A is a state, and there does not exist a state B from which A can be derived, then A is set as an initial path (starting point).

∀A:STATE ∀D:DURATION
state(A) ∧ ¬∃B:STATE leads_to_after(B, A, D)
↠ temporary_path(A)

**P4: Build up paths**

If P is a temporary path, and the last element is A, and A is known to lead to B which is not part of P yet, then path P with B added is a new temporary path.

∀P:PATH A, B:STATE, D:DURATION
temporary_path(P) ∧ last_element_of(P, A) ∧
leads_to_after(A, B, D) ∧ B ∉ P ↠ temporary_path(P + B)

**P5: Select complete paths**

If P is a temporary path, and there does not exist any state B that can be derived from A (the last element in path P), then this is a complete path.

∀P:PATH A:STATE D:DURATION
temporary_path(P) ∧ last_element_of(P, A) ∧
¬∃ B:STATE leads_to(A, B) ↠ path(P)

The construction described above assumes *or* type structures specified in using the leads_to_after rules, *and* structures could however easily be incorporated.

### *5.2. Focusing Heuristic Function Outline*

Given a certain workflow, all possible paths in the workflow and certain states that have been observed:

**Focusing heuristic function**

1. Select the paths that contain observed states, called consistent paths.
2. From the set of paths that have now been selected another selection is made based upon softer criteria:
    a. The combination of resources used and the competences of the operator
    b. The past paths the operator has taken

    Each of these criteria specifies the likelihood of the various paths with a total of 1 for all paths. Combine them by taking the weighed sum of the softer criteria for each path, and take the one with the highest value, all the states within this highest path are then in focus.

The result of the heuristic function is a ranking of the various paths that are possible (given the available knowledge). Thereafter, the most likely path (i.e. the one with the highest rank) is selected. Upon this path the full reasoning is performed (following Section 4). Below, each of the individual elements of the heuristic function will be discussed in more detail.

### *5.3. Remove Inconsistent Paths*

The first step in the heuristic function is to remove the paths that are inconsistent with the observations from the set of generated paths. Paths are marked as inconsistent in case they do not contain a state which has been observed.

**P6: Mark path with lacking observation**

In case a state has been observed to have occurred, and this state is not part of a path, then this part is marked as being inconsistent.

∀A:STATE, ∀T:TIME
path(P) ∧ observation_result(at(A, T), pos) ∧ A ∉ P
↠ inconsistent_path(A)

### *5.4. Calculate Soft Criteria*

Of the remaining (consistent) paths, the soft criteria are calculated. In order to enable this calculation, certain information about the human should be known within the personal assistant. In Table 3, the predicates are shown of the knowledge the personal assistant has about the human. Hereby, two are distinguished, namely the combination of resources and competences, and the experience paths of the human (note that for the latter, no separate predicate is introduced). This information can be learned by the personal agent based upon experiences, but also based upon background knowledge about the human (e.g. prior education).

**Table 3.** Knowledge about the human

| Predicate | Explanation |
|---|---|
| human_has_competence_type_for: COMPETENCE_TYPE x COMPETENCE_TYPE_LEVEL | A human is known to have a certain level of competence for a particular competence type. |
| has_basic_competence_level_for: TASK x COMPETENCE_LEVEL | A human has a basic competence level for a certain task, without using resources. |

*Resources combined with competences*

The combination of resources that have been observed being used, and the competences of the human can deliver a lot of information. Basically, using the combination it can be calculated at what competence level the combination is, and what the mismatch of this level with the required level is.

**P7: Calculate competences with resources**

If it has been observed that a resource has been used, and the human has the appropriate competence type for the resource, then the resulting combination results in a particular competence level for a state.

∀T:STATE ∀R:RESOURCE, C:COMPETENCE_TYPE,
CTL:COMPETENCE_TYPE_LEVEL,
RC:RESOURCE_CONDITION,
CL:COMPETENCE_LEVEL
observation_result(resource_used(R), pos) ∧
requires_competence_type(R, C) ∧
human_has_competence_level_for_type(C, CTL) ∧
resource_condition(R, RC)
resulting_competence_level_for(R, CTL, RC, T, CL)
→ competence_for(T, CL)

Furthermore, the human might also be able to perform a certain task without a certain resource. In that case, the basic competence level for the state is assumed to be the competence level of the human for the state.

**P8: Basic competence level without resources**

If a human has a certain basic competence level for a state, then this is also the competence of the human for the state.

∀T:STATE, CL: COMPETENCE_LEVEL
has_basic_competence_level_for(T, CL)
→ competence_for(T, CL)

Thereafter, the value for the competence level for a state can be determined by combining the levels for all resources used for that state. Now the deviation from the required level can be calculated as follows.

$$sumprob\_path_a =$$

$$\sum_{\forall taskt} | requires\_competence\_level(t)_i - highest\_competence\_level(t)|$$

$$competence\_prob\_p_a = 1 - \frac{sumprob\_path_a}{\sum_{path\ x} sumprob\_path_x}$$

*Experience paths*

Determine for all paths how often the human has taken at least one of the paths, then for each path determine the number for that specific path (sum of experiences with all nodes) and divide it by the total number of experiences with the workflow:

$$experience\_prob\_p_a = \frac{\#experiences\_path_a}{\#total\_experiences}$$

These two likelihoods can be combined by using a weighed sum. This way, more criteria can be added easily and the weights thereof can vary per domain.

$$likelihood\_p_a = w_1 \cdot competence\_prob\_p_a + w_2 \cdot experience\_prob\_p_a \text{ where } w_1 + w_2 = 1$$

Thereafter, the path with the highest likelihood can simply be selected. All the states within this path are then in focus (which is the selection criterion s1 as introduced in the beginning of Section 5):

**P9: Generate foci for states**

If a path is in focus, then all elements of that path are also in focus.

∀T:STATE ∀P:PATH
path_in_focus(P) ∧ T ∈ P
→ in_focus(T)

## 6. Case Study

In order to show the functioning of the approach, a number of scenarios have been simulated using the model described above and the workflow model of building a shed (see Figure 1).

In the first scenario, a person starts building a shed at a certain time point. During building the shed, the person does not use any resources that can be observed by the Ambient Agent. The person has the most experience with building a shed out of stone and buying the materials from the store. In scenario 2, the experience of the builder is the same, but s/he uses a hammer and folding rule, which can be observed by the Ambient Agent.

The simulation runs have been performed using the dedicated LEADSTO environment [8]. Figures 2 and 3 show the traces of two simulation runs. The left side of the figure indicates the atoms that occur over time, whereas the right sides indicates a timeline where a dark box indicates that the atom is true, and a grey box indicates false. Note that the time on the x-axis is the simulation time which is not related to the time points in the atoms. The time points in the predicates such as belief(at(A1, 7)) describe the time at which the person is believed to have started a specific task. The timeline at the bottom of the figure is the simulation time and shows the order in which atoms are derived by the Ambient Agent. Also note that in the traces the predicate belief represents the selected beliefs of the agent.

In the first scenario, the ambient agent of the person building a shed has observed that this person was going to build a shed at time point 6 (observation_result(at(O1, 6))). Using this observation and knowledge about the competences, resources, and experiences of the human, the ambient agent can start reasoning about the most likely path in the workflow. During the focus generation phase, the agent selects all four paths based on one belief about state O1 (see selected_path(p(...)) for all four paths). Then, because no use of resources has been observed, the basic competences of the activities are used to determine the distance from the required competence levels following the formula of P8 in Section 5.4. In Figure 2, the required, combined, and differed competence levels are only shown for activities A1 and A4. Using the formula from Section 5.4, the ambient agent can calculate the competence

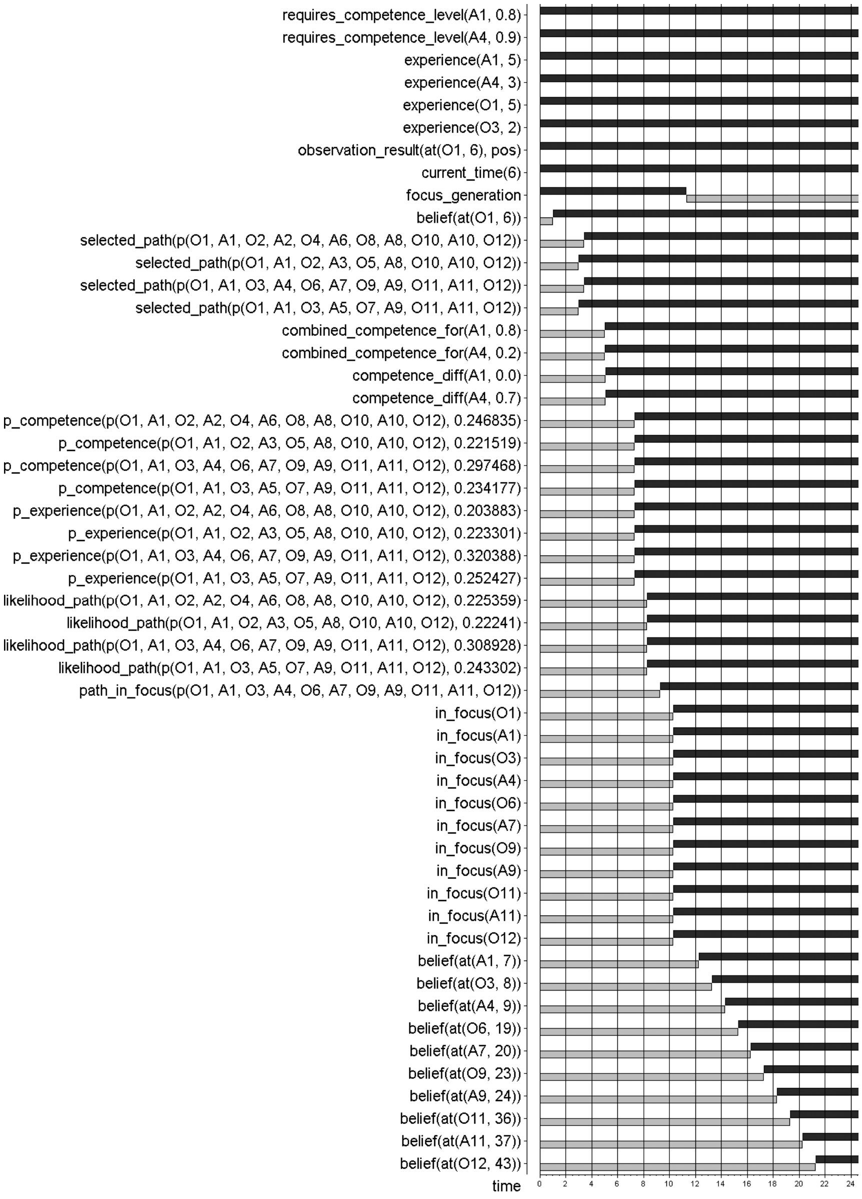


**Figure 2.** Simulation trace of scenario 1.

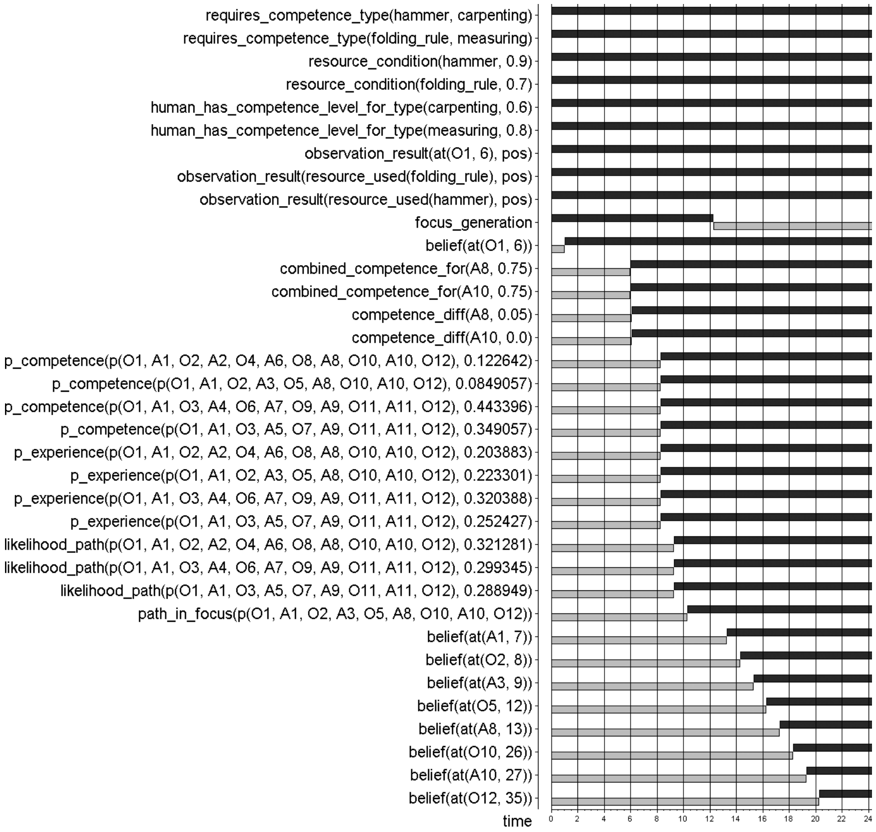


**Figure 3.** Simulation trace of scenario 2.

difference of all paths (p_competence(PATH, P)) Note that the numbers shown are the values before they have been subtracted from 1 as specified in the formula in Section 5.4. This means that the lower the number, the better this path matches with the required competence. The same can be done for experience using the formula for experience in Section 5.4 (p_experience(PATH, P)). Here, a higher number represents more experience and thus more likely to be chosen again by the human. The combined likelihoods for all paths are calculated and the path with the highest score is selected (likelihood_path(PATH, L) and path_in_focus(PATH)). The agent focuses on all states of this path and finishes the focus generation phase. During the belief generation phase, the agent reasons with the belief about O1 and the foci. The agent has determined the most likely path, namely the third path consisting of buying bricks and building a brick shed.

In the second scenario, see Figure 3, the ambient agent has more information about the resources the human has used during building the shed. The agent has observed the use of a hammer and a folding rule. This results in different competence values for the tasks that can be performed using a hammer and folding rule (activities A8 and A10). The agent focuses on a different path from scenario 1, and determines that the human has been following the second path: building a wooden shed with materials available in storage.

## 7. Dynamic Properties and their Verification

For the model a number of overall properties have been identified and formally specified using the Property Editor of TTL (an explanation of TTL is given in Section 3). These properties have been automatically verified against the simulation traces of the two scenarios using the Checking Tool of TTL:

**P1: Observed facts will become selected beliefs**

If at time point t a new observation result exists, and the current set of beliefs does not comply to this observed information, then eventually the set of derived beliefs will change such that it complies to the observation result.

```
∀γ:TRACE, t:TIME, I:STATE, T:TIME
[ [ state(γ, t) |= observation_result(at(I, T), pos) &
   ¬state(γ, t-1) |= observation_result(at(I, T), pos) &
   ¬state(γ, t) |= belief(at(I, T)) ]
        ⇒    ∃t2:TIME ≥ t state(γ, t2) |= belief(at(I, T)) ]
```

Note that this property assumes that observation results are always trusted. In case this assumption does not hold a special type of belief such as observed_belief can be introduced instead of the actual observation result.

The property has been shown to be satisfied for the traces of the two scenarios.

**P2: Derived beliefs are within one path in the workflow model**

When a set of beliefs is derived, then this set contains at most one execution path of the workflow.

```
∀γ:TRACE, t:TIME, I1, I2:STATE, T1, T2:TIME
[ state(γ, t) |= belief(at(I1, T1)) &
  state(γ, t) |= belief(at(I2, T2)) &
  I1 ≠ I2 & T2 > T1]
        ⇒  ∃P:PATH [ state(γ, t) |= element_of(I1, P) &
                         state(γ, t) |= element_of(I2, P) ]
```

This property is satisfied for both traces as well.

**P3: Most plausible beliefs are generated first**

If one path is more likely to be followed than another path according to the background information, and there are no observed beliefs that distinguish either path, then this path will be explored first by deriving beliefs along that path.

```
∀γ:TRACE, t:TIME, P:PATH, V:VALUE
   [[ state(γ, t) |= likelihood_path(P, V) &
     state(γ, t) |= path_in_focus(P) ]
    ⇒ [ ∀I:STATE, t0:TIME
         [ state(γ, t0) |= observation_result(I, pos) ⇒
           state(γ, t) |= element_of(I, P) ] ] &
        ∀P2:PATH ≠ P, V2:VALUE
        [ state(γ, t) |= likelihood_path(P2, V2) & V2 > V
          ⇒ ∃I2:STATE, t':TIME< t [
            state(γ, t') |= observation_result(I2, pos) &
            state(γ, t) |= ¬element_of(I2, P2)]
```

This property was shown to be satisfied for all traces.

**P4: Only beliefs consistent with observations are derived**

If beliefs are derived, then these beliefs need to be consistent with the observations.

```
∀γ:TRACE, t:TIME, I:STATE, T:TIME
[ state(γ, t) |= belief(at(I, T))  ⇒
   ¬∃t':TIME [ t' < t &
          state(γ, t) |= observation_result(at(I, T), neg)] ] ]
```

This property is satisfied for the two traces, indicating that the beliefs derived are always accurate compared to the observations.

**P5: Immediate correct derivation**

If a set of beliefs is derived at time t, then there does not exist a time t' > t where the derived belief is not consistent with the observations.

```
∀γ:TRACE, t:TIME, I:INFO_EL, T:TIME
[ state(γ, t) |= belief(at(I, T)) ⇒
  ¬∃t2:TIME > t observation_result(at(I, T), neg) ]
```

This property is also shown to be satisfied.

## 8. Related Work

For the representation of workflows, a variety of approaches have been developed. For instance, in [5], [9] and [10] examples of such representations are shown. In [1] a comparison between several of these approaches is discussed. The approach presented in this paper has been tailored towards providing information for the reasoning process, and therefore has a different focus for which it has been developed, leading to different design choices. In [15] an approach to recognize plan execution states is also presented. The current activities of the human are derived by means of probabilistic methods. In this paper however, a more logical approach is taken. Due to the hybrid quantitative and qualitative approach, it is also possible to combine the logics with probabilities. [13] introduces an approach to focus reasoning processes using case-based reasoning methods. The paper does however not use domain knowledge (e.g. resources) such as done in this paper.

In other work such as [12] and [15] also temporal relationships between activities are exploited to recognize plan execution states, based on relational Markov networks and causal networks, respectively. These papers address the theme from a probabilistic angle, whereas the current paper addresses it from a logical reasoning perspective. For future work it will be interesting to explore how such probabilistic approaches can be integrated within a logical reasoning approach.

In [17] approach is presented in which simulations are used to predict near-future behaviour. However, only observations of historic behaviour are used while in the current work also data about the human is analysed for the reasoning process.

## 9. Discussion and Conclusion

This paper addressed one of the challenges for ambient intelligent agents (e.g., [2; 3; 14]), to support a human in demanding tasks, namely to be aware of which (sub)tasks the human is doing, and how much progress is made, without direct communication about this.

A formally specified executable ambient agent model was introduced that is able to perform model-based analysis using available workflow models, available observation information and experiences of the person executing the workflow. Thus it obtains awareness of the human's progress in task execution. Hereby, a generic focused reasoning method has been used (based upon the formally specified model-based reasoning methods from [4]), whereby the most likely path is evaluated first, avoiding having to pass all possible paths, hence, improving the scalability. How much benefit the proposed mechanism brings depends severely on the workflow at hand, as well as how good the knowledge is that drives the focusing. Therefore, it is hard to give theoretical results. Simulation experiments for a case study concerning the question what the human is doing at the current time point have been shown. These simulation runs have been used to verify particular requirements related to the process. In Section 7 a number of such (required) formally specified dynamic properties of the overall process and their verification were discussed. For specification of these properties the expressive predicate logic temporal language TTL was used [7]. As this language allows the use of variables (also over numbers) and quantifiers, it is more useful for practical applications than, for example, propositional and modal temporal languages. The simulation results have shown that the ambient agent indeed satisfied the expressed requirements.

Of course, more interesting questions concerning the workflows can be posed, for instance: what has the human been doing until now? The ambient agent might want to find out which path the human has taken to reach his/her goal in order to store this information as experience for example, or to store this as information for future reference (when the same workflow needs to be executed advice can be given about the previous solution). Yet another question that can be addressed is: what should the human be doing in the future? If the ambient agent needs to determine the shortest path between two states, it can use a breadth-first search mechanism. The ambient agent determines the next location that can be reached in all paths after a fixed amount of time (1 minute for example). The path that reaches the last node first is the shortest one.

In the current approach it is assumed that the observations are deterministic. Another option would be to obtain observations with a certain probability. The approach presented in this paper can handle such probabilistic observations by simply assuming the one with the highest probability. More advanced methods are future work. Furthermore, for future work the aim is to evaluate how effective the reasoning method with the focusing mechanism is in a practical setting.

## References

[1] Aalst, W.M.P. van der, Hofstede, A.H.M. ter, Kiepuszewski, B., and Barros, A.P., Workflow Patterns. *Distr. and Parallel Databases*, vol. 14, 2003, pp. 5-51.

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

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

[4] Bosse, T., Both, F., Gerritsen, C., Hoogendoorn, M., and Treur, J., Model-Based Reasoning Methods within an Ambient Intelligent Agent Model. In: *Constructing Ambient Intelligence: AmI-07 Workshops Proceedings*. CCIS, vol. 11, Springer Verlag, 2008, pp. 352-370.

[5] Gottschalk, F., van der Aalst, W.M.P., Jansen-Vullers, M.H. and La Rosa, M. (2008) Configurable Workflow Models. *International Journal of Cooperative Information Systems* Vol. 17, no. 2, pp. 177-221.

[6] Bosse, T., Hoogendoorn, M., Klein, M., and Treur, J., A Component-Based Ambient Agent Model for Assessment of Driving Behaviour. In: F.E. Sandnes et al. (eds), *Proceedings of the 5th International*

*Conference on Ubiquitous Intelligence and Computing, UIC'08*. Lecture Notes in Computer Science, vol. 5061. Springer Verlag, pp. 229-243. (2008)

[7] Bosse, T., Jonker, C.M., Meij, L. van der, Sharpanskykh, A., and Treur, J., Specification and Verification of Dynamics in Agent Models. *International Journal of Cooperative Information Systems*, vol. 18, 2009, pp. 167-193.

[8] Bosse, T., Jonker, C.M., Meij, L. van der, and Treur, J. A Language and Environment for Analysis of Dynamics by Simulation. *International Journal of Artificial Intelligence Tools*, vol. 16, 2007, pp. 435-464. (2007)

[9] Ellis, C.A. and Nutt, G.J.. Modelling and Enactment of Workflow Systems. In M. Ajmone Marsan, editor, Application and Theory of Petri Nets, Lecture Notes in Computer Science, vol. 691. pp. 1-16. Springer Verlag, Berlin (1993)

[10] Georgakopoulos, D., Hornick, M., and Sheth, A.. An Overview of Workow Management: From Process Modeling to Workow Automation Infrastructure. Distributed and Parallel Databases, 3:119-153. (1995)

[11] Jaimes, A.; Sebe, N., Gatica-Perez, D. Human-Centered Computing: A Multimedia Perspective. In: *Proc. of the 14th ACM Int. Conf. on Multimedia, ACM Press*, 2006, pp. 855-864

[12] Liao, L., Fox, D., and Kautz, H., Location-Based Activity Recognition using Relational Markov Networks. In: Proceedings of the Nineteenth International Conference on Artificial Intelligence, IJCAI'05. (2005)

[13] Portinale, L., Torasso, P., Ortalda, C., and Giardino, A., Using case-based reasoning to focus model-based diagnostic problem solving. In: *Topics in Case-Based Reasoning*, LNAI 839, 1994, pp. 325 – 337.

[14] Riva, G., F. Vatalaro, F. Davide, M. Alcañiz (eds.) Ambient Intelligence. IOS Press (2005)

[15] Qin, X., and Lee, W., Attack Plan Recognition and Prediction Using Causal Networks. In: Proc. of the 20th Annual Conference on Computer Security Applications, pp. 370- 379. (2004)

[16] Weiser, M., Some computer science issues in ubiquitous computing. *ACM SIGMOBILE Mobile Computing and Communications Review*, vol. 3, 1999, pp. 1559- 1662.

[17] A. Rozinat, M.T. Wynn, W.M.P. van der Aalst, A.H.M. ter Hofstede, and C. Fidge. Workflow Simulation for Operational Decision Support Using Design, Historic and State Information. In M. Dumas, M. Reichert, and M.C. Shan, editors, *International Conference on Business Process Management (BPM 2008)*, volume 5240 of Lecture Notes in Computer Science, pages 196{211. Springer-Verlag, Berlin, 2008.