An Ambient Intelligent Agent with Awareness of Human Task Execution

Fiemke Both[1], Mark Hoogendoorn[1], Andy van der Mee[2], and Michael de Vos[2]
[1]*Vrije Universiteit Amsterdam, Department of Artificial Intelligence*
De Boelelaan 1081a, 1081 HV Amsterdam, The Netherlands
{fboth, mhoogen}@few.vu.nl
[2]*Force Vision Lab, Barbara Strozzilaan 362a, 1083 HN Amsterdam, The Netherlands*
{andy, michael}@forcevisionlab.nl

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 focusing mechanism to guide and accelerate the reasoning process in concluding the path most likely selected 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 new workflow representation is introduced. The approach is evaluated by means of an extensive case study.

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 [10], and Human Aware Computing [7]. 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 have a clear view of what the human is doing at particular time points, and knowledge of what support could be given. This information can be derived from knowledge about the active workflow (or possibly workflows) the human is currently involved in. Ideally, the agent would be able to observe all activities of the human, and simply select the correct path in the workflow. However, not all activities the human undertakes might be observable, or the observations could be highly uncertain. Therefore, many of the available paths in the workflow could still be possible, resulting in a very complex reasoning task to monitor the human activities.

In [4] an approach has been proposed that allows an agent to reason about workflows to monitor the human activities, and use background knowledge to focus this reasoning process This focused reasoning is proposed to avoid an explosion of the complexity of the reasoning process, there could for instance be loops within the workflow. What knowledge could be used for this focusing was however left unattended. [6] addressed one option to focus the reasoning process, namely to use the competences of the operator.

In this paper, a more advanced focusing mechanism is proposed that takes a variety of factors into account, namely: (1) the structure of the workflow; (2) competences; (3) previous experiences of the human with the workflow, and (4) background knowledge about the tasks in the workflow (e.g. what resources are needed to perform a task). In order to come to such an approach, the workflow representation as used in [4;6] has been extended. Furthermore, a series of rules to perform the focusing of the reasoning process of the agent have been identified. These rules have been

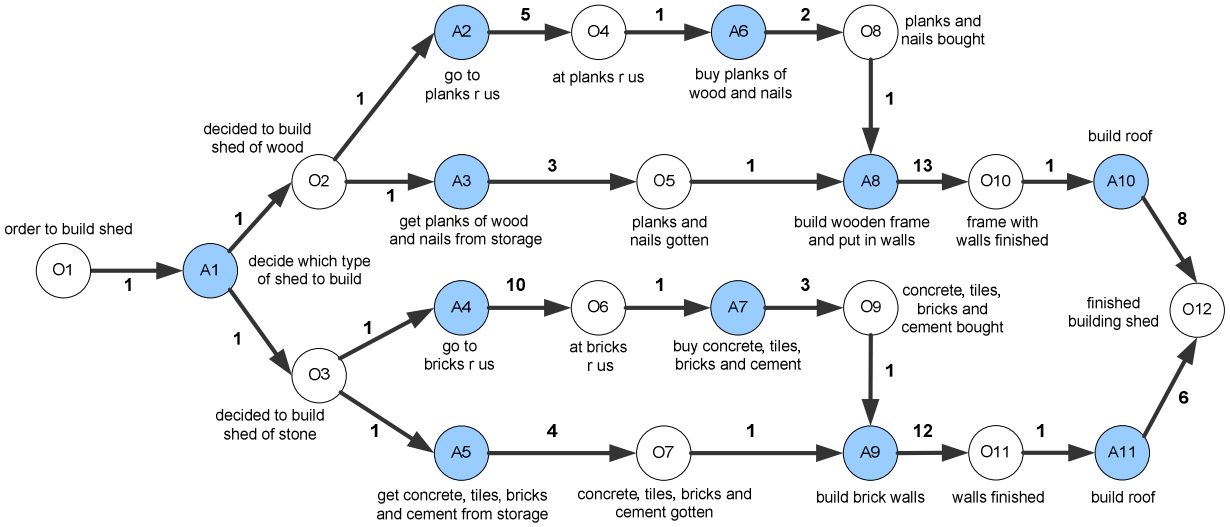

Figure 1. Example workflow: Building a Shed

specified in such a way, that the approach can easily be extended with additional information sources.

This paper is organized as follows. In Section 2 the newly created workflow representation is addressed. Furthermore, the example used as a case study is introduced in that section as well. Thereafter, Section 3 briefly shows the reasoning process itself, as taken from [4]. The focusing mechanism is introduced in Section 4, and experiments using the focusing mechanism are presented in Section 5. Section 6 presents related work, and Section 6 concludes the paper and discusses future work.

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 finish the state of the source of the arrow, and be ready move to the next state. In case a conjunction of states is required, the arrows are combined using an arch. 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.

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. Model-Based Reasoning

Now that it is known how the workflow can be represented, reasoning methods can be applied. First, the method without explicit focusing is addressed. This reasoning mechanism used is taken from [4]. The mechanism works on the basis of a set of rules about states within the workflow leading to other states. Hereby, time is taken as an explicit parameter. A state holding at a particular time point is represented by specifying a belief around the workflow representation shown in Section 2.2: belief(at(state, time)). The same holds for the specification : belief(leads_to_after(state1, state2, duration)). Of course, conjunctions and disjunctions can be represented in the leads_to_after relationship as well. Using these predicates, reasoning constructs involving forward and backward reasoning can be specified. First, the logical format to enable this reasoning is explained, thereafter some example reasoning rules are presented.

3.1. LEADSTO language

The rules within the reasoning mechanism are specified in an executable logical format called LEADSTO [5]. 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. The LEADSTO language features a simulation engine. For more details on the LEADSTO language, see [5].

3.2. Reasoning Rules

For both forward and backward simulation well-known reasoning techniques can be applied. Furthermore, the approach takes a specific focus of the reasoning into account (of which the mechanism to determine the precise focus will be addressed in Section 4). An example of a forward reasoning rule is shown below. Note that the subscript below the LEADSTO arrow has been left out, meaning that the standard parameters 0,0,1,1 are used.

P1: 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.

$\forall$I,J:STATE $\forall$D:DURATION $\forall$T:TIME
belief(at(I, T)) $\wedge$ belief(leads_to_after(I, J, D)) $\wedge$ in_focus(J)
$\rightarrow\!\!\!\!\rightarrow$ belief(at(J, T+D))

Of course, more forward reasoning rules exist, see [4] for more details. For backward reasoning the abduction principle can be applied:

P2: Positive backward simulation
$\forall$I,J:STATE $\forall$D:DURATION $\forall$T:TIME
belief(at(J, T)) $\wedge$ belief(leads_to_after(I, J, D)) $\wedge$ in_focus(I)
$\rightarrow\!\!\!\!\rightarrow$ belief(at(I, T-D))

The results of applying this rule are not guaranteed to be correct since there could be multiple leads_to_after rules that cause J to occur. Again, see [4] for more details and backward simulation rules.

4. Advanced Focusing Methods

Given that it is possible to reason through the workflow in a focused manner using the methods specified above, the focusing itself can be addressed, which is one of the main contributions of this paper. First, the focusing mechanism proposed is briefly outlined. Thereafter more details are given how to calculate the specific parameters within the focusing mechanism.

4.1. Focusing Algorithm Outline

Given a certain workflow, and certain states that have been observed within this workflow:

Focusing algorithm

1. Based upon the structure of the workflow, determine the paths the human could take. Hereby, time is not considered.
2. Select the paths that are consistent with observed states.
3. 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 specify 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

The result of the algorithm 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 3). Below, each of the individual elements of the algorithm will be discussed in more detail.

4.2. Path Generation

The first step is to determine all possible paths. 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.

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).

$\forall$A:STATE $\forall$D:DURATION
state(A) $\wedge$ $\neg\exists$B:STATE leads_to_after(B, A, D)
$\rightarrow$ 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.

$\forall$P:PATH A, B:STATE, D:DURATION
temporary_path(P) $\wedge$ last_element_of(P, A) $\wedge$
leads_to_after(A, B, D) $\wedge$ B $\notin$ P $\rightarrow$ 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.

$\forall$P:PATH A:STATE D:DURATION
temporary_path(P) $\wedge$ last_element_of(P, A) $\wedge$
$\neg\exists$ B:STATE leads_to(A, B) $\rightarrow$ 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.

4.3. Remove Inconsistent Paths

The next step in the algorithm 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.

$\forall$A:STATE, $\forall$T:TIME
path(P) $\wedge$ observation_result(at(A, T), pos) $\wedge$ A $\notin$ P
$\rightarrow$ inconsistent_path(A)

4.4. 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 state without a certain resource.

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.

$$sumprob\_path_a = \sum_{\forall task_t} | requires\_competence\_level(t)_i - highest\_competence\_level(t) |$$

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

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 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.

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

Thereafter, the path with the highest likelihood can simply be selected.

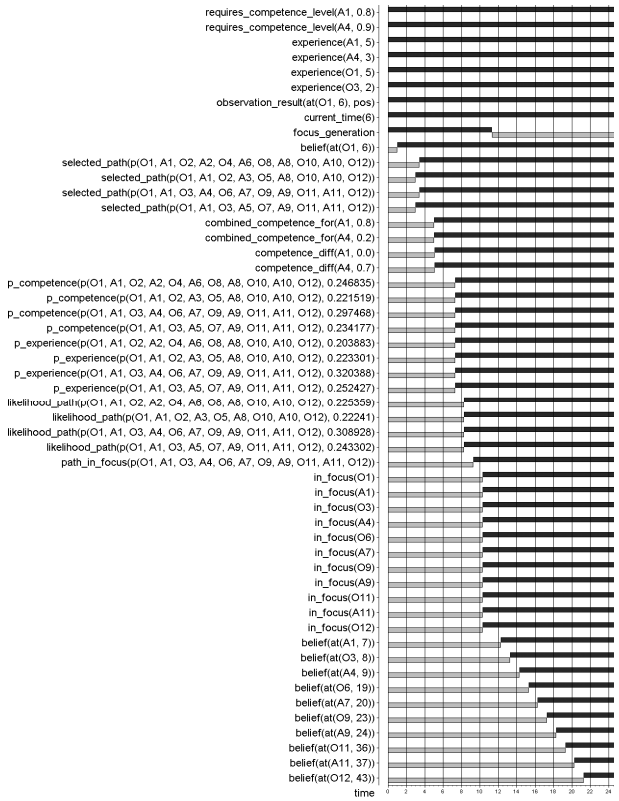

Figure 2. Simulation trace

5. 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). The simulation runs have been performed using the dedicated LEADSTO environment [5]. Below, only one simulation run is shown for the sake of brevity. Figure 2 shows the trace of this simulation run. 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. 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. 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. In Figure 1, the required, combined, and differed competence levels are only shown for activities A1 and A4. Using the formula from section 4.4, the ambient agent can calculate the competence 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 4.4. 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 4.4 (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 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.

6. Related Work

For the representation of workflows, a variety of approaches have been developed, see [1] for a comparison between several of these approaches. 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 [9] 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 logical approach is taken. [8] 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.

7. Conclusions and Future Work

In this paper, a reasoning method has been presented that enables an agent to derive what a human is doing. Having such knowledge allows an agent to support humans in the best possible way. In order to come to such an approach, a simple workflow representation has been introduced, for which reasoning methods have been used as introduced in [4]. Since the workflows of a human could potentially consist of a lot of paths, and not every element in those paths could be observable, a focusing of this reasoning is needed. Hereby, heuristic knowledge about the order in which the paths should be passed can aid to set an appropriate focus. 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. The approach presented in [6] is taken as a basis for the focusing. In this paper however, the information expressed in the workflow is into account, as well as the information about the experiences. Simulation results have shown that the focusing mechanism indeed functions to derive what the human is using, and the revision mechanism does so as well.

In future work, the aim is to evaluate how effective the reasoning method with the focusing mechanism is in a practical setting. Furthermore, the focus in this paper has mainly been on deriving what the human is doing, and not so much on what support could be given. This is part of future work as well.

8. 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.). *Proceedings of EUSAI 2003*. LNCS, vol. 2875. Springer Verlag, 2003, pp. 432.

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

[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] 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.

[6] Both, F., Hoogendoorn, M., and Treur, J., An Ambient Agent Model Exploiting Workflow-Based Reasoning to Recognize Task Progress. In: *Proc. of AmI 2008*. LNCS, Springer Verlag, 2008, to appear.

[7] 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

[8] 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.

[9] Qin, X., and Lee, W., Attack Plan Recognition and Prediction Using Causal Networks. In: *Proc. of the 20th Conf. on Computer Security Appl.*, 2004, pp. 370- 379.

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