HELSINKI
INSTITUTE FOR
INFORMATION
TECHNOLOGY

PROACTIVE REASONING IN MOBILE HCI: TACKLING UNCERTAINTY AND APPROPRIATE TASK SUPPORT

Antti Salovaara

November 14, 2004

HIIT
TECHNICAL
REPORT
2004-2

Proactive Reasoning in Mobile HCI: Tackling Uncertainty and Appropriate Task Support

Antti Salovaara
Helsinki Institute for Information Technology
P.O. Box 9800, 02015 HUT, Finland
antti.salovaara@hiit.fi

HIIT Technical Reports 2004-2
ISSN 1458-9478

PROACTIVE REASONING IN MOBILE HCI: TACKLING UNCERTAINTY AND APPROPRIATE TASK SUPPORT

Antti Salovaara

Helsinki Institute of Information Technology, Helsinki, Finland

Abstract

Mixed-initiative user interfaces provide a way to reason about user's time-varying goals using decision theory to decide when a system should invoke actions to support the user. Using this approach as a starting point, this paper extends the theory with hierarchical task analysis, showing ways to address questions of (1) how to define what actions the system should have in its repertoire, (2) how support could be provided in multiple ways in a situation, depending on system's certainty of user's current task, and (3) how the topology of a probabilistic reasoning graph should be derived. A conceptual evaluation is carried out against concrete observations from elevator maintenance work. The analysis suggests that the extended framework can be best used in such mobile contexts where activity is goal-directed and orderly, and system's actions will not change user's activity remarkably.

1 Introduction

Understanding the purpose of user's actions is crucial to any context-aware system that is intended for providing real-time support for the user. Research conducted this far has shown that this goal is extremely difficult to achieve in a larger scale than possibly in a command line keystroke level. The area of interest in context-aware computing and mobile HCI is however much bigger than that, and thus harder to tackle.

The problem of understanding user's activity shows various paths as a solution: (1) designing work so that the user will act more predictably, (2) making the activity more understandable to the computer through well-designed dialogue with the user, or (3) integrating uncertainty of user's intentions to the design of context-aware systems. While the first alternative may be possible in some settings, it tends to make users subordinate to the computing system and lose the benefits of flexibility in human behavior. The second approach requires user engagement and is not possible in unobtrusive context-aware systems. Thus, the last approach seems the most appropriate one.

In this paper, uncertainty will be tackled in the framework of mixed-initiative user interfaces [8], which is a decision-theoretic approach to selecting right type of proactive task support in real-time use situations. The original treatment by Horvitz shows how a system can decide whether to invoke an action or not, as will be shown in a short recapitulation in this paper. The theory has not yet been fully developed, though. For instance, it does not address clearly how the user should be supported in more than one way in a situation, or how to know what supporting actions should be linked to each user goal or task. In addition, the decision-theoretic approach has not yet been introduced to mobile HCI research (or context-aware computing away from the desktop).

Therefore, this paper will extend the theory by analyzing its connections to hierarchical task analysis (HTA) (*e.g.*, [17]). The connection to HTA comes from linking specificity levels of supporting actions with the user's task description hierarchy. This may potentially provide a way to define how specific support in each situation can be allowed without disturbing the user, while the certainty of his or her present task is continuously changing. After having developed the model, its feasibility is evaluated against one use context (elevator maintenance work) and two observed use situations where worker's unexpected behavior would pose problems to a context-aware system.

2 How Mixed-Initiative User Interfaces Address Uncertainty

Mixed-initiative user interfaces, presented by Eric Horvitz, aim for coupling interface agents and direct manipulation through collaboration between the agent and the user. One of the primary questions is the consideration of costs and benefits of automated actions to the user, and to this, Horvitz proposes decision-making as a function of inferred probabilities of the user's goals. As a starting point, he lists 12 principles that address the critical factors behind successful decision-making, touching issues such as user attention and efficient mechanisms for establishing and maintaining mutual understanding of the present task. The ones related to probabilities and of interest in this paper are replicated below ([8], pp. 159-160, original emphases).

(2) Considering uncertainty about user's goals. Computers are often uncertain about the goals and current focus of attention of a user. In many cases, systems can benefit by employing machinery for inferring and exploiting the uncertainty about a user's intentions and focus.

(4) Inferring ideal action in light of costs, benefits, and uncertainties. Automated actions taken under uncertainty in a user's goals and attention are associated with context-dependent costs and benefits. The value of automated services can be enhanced by guiding their invocation with a consideration of the expected value of taking actions.

(8) Scoping precision of service to match uncertainty, variation in goals. We can enhance the value of automation by giving agents the ability to gracefully de-grade the precision of service to match current uncertainty. A preference for "doing less" but doing it correctly under uncertainty can provide user's with a valuable advance towards a solution and minimize the need for costly undoing and backtracking.

In mixed-initiative user interfaces, the fundamental tool for measuring action's helpfulness to a user is to look at its expected utility. The purpose is to estimate the benefit of invoking a certain action, given the agent's beliefs about user's goals. Expected utility is different in each situation, that is, it changes continuously.

At a closer look, Horvitz's decision-theoretic treatment is following. Having observed the evidence E (*e.g.*, a sequence of user actions, data about the context, and so on) for user having a goal G, let us assume that the agent now knows the likelihood $p(G \mid E)$ for user having that goal. Agent should now know whether to invoke an action A or not. There are four decision alternatives that the agent needs to consider, each associated with its own utility u to the user (utility being a function of how much the task can be progressed, how much the user is distracted, side-effects, and so on):

- $u(A,G)$: utility of agent invoking action A, having inferred user's goal G correctly.
- $u(A,G)$: utility of agent invoking action A, having inferred user's goal G incorrectly.
- $u(A,G)$: utility of agent not invoking action A, having inferred user's goal G correctly.
- $u(A,G)$: utility of agent not invoking action A, having inferred user's goal G incorrectly.

Now the expected utility $eu(A \mid E)$ of invoking the action A becomes a sum of the related utilities, weighted with the probabilities of user having (G), or not having (G), the goal in light of observed evidence E:

$$eu(A \mid E) = u(A,G)\, p(G \mid E) + u(A\, G)\, p(G \mid E) \quad (1)$$

Since $p(G|E) = 1–p(G|E)$, the equation can be rewritten as

$$eu(A \mid E) = u(A,G)\, p(G \mid E) + u(A,G)\, [1 - p(G \mid E)] \quad (2)$$

For the actions that the agent has at hand, a separate expected utility is computed to choose the most appropriate action for the situation. But there still remains the question of whether the action should be invoked at all. That is, in what situations should now the agent invoke the seemingly most appropriate action?

The diagram in Figure 1 shows Horvitz's decision-theoretic answer to the question, from a viewpoint of one specific situation with respect to an agent's single action and user goal. In each situation E is different, and the utilities u are unique for each action–goal pair. Thus a separate analysis is needed for each situation and action. The two lines in the figure show the expected utilities of the two alternatives (*i.e.*, whether to act or not) open to the agent.

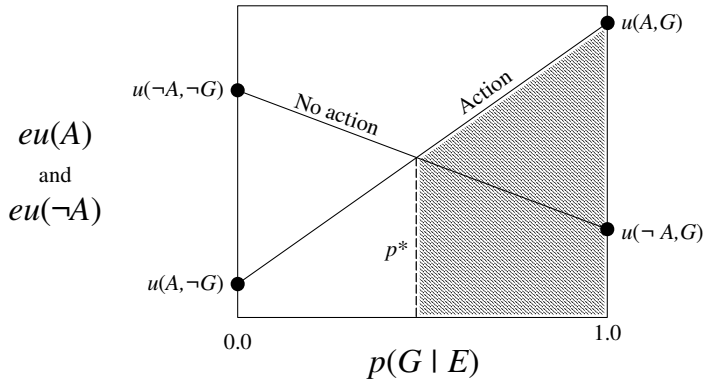

Figure 1. A diagram depicting the decision whether to act or not to act in a given situation. The shaded area, determined by the threshold probability *p**, shows when the agent should act (adapted from [8]).

Essentially, the diagram shows how to make the yes/no decision for an action. The solution lies in estimating $p(G \mid E)$, that is, the confidence of how sure the agent is about the goal, given the evidence. In the case of Figure 1, if the confidence is high enough, the action should be taken. The probability p^* at the intersection of the two lines dictates the threshold whether to act or not.

After having presented this model, Horvitz shows how the principles were put into use in an anthropomorphic MS Outlook assistant that helps organizing information within incoming emails. For instance, when the agent finds a likely proposal for an appointment in a mail, it suggests placing the event into a calendar. If the agent is uncertain about the exact time, it only opens the correct calendar page, but otherwise may put the event into its place and wait for user's approval.

3 Elaboration of the Model

As can be understood from the analysis, choosing the most appropriate action at each moment can be generally an intractable problem. It would require continuous matching of all the possible user goals *G* and agent actions *A* against the observed evidence *E*, over and over again, since the evidence changes from a situation to another. Therefore in practice, the number of possible goals and associated actions to consider should be somehow limited.

In his principle no. 8 Horvitz suggests that the precision of service (*i.e.*, system's autonomous actions) should be adapted to the uncertainty about user's present task. That is, for each situation, there should be a set of alternative ways to provide help. The more certain the agent is of user's goal, the more specific support should be provided. When the confidence is low, actions should be helpful in a more general level, in order to decrease the amount of inappropriate, task-specific obtrusive actions. This orders the actions according to their specificity. Let us now analyze how this hierarchical ordering could be benefited.

3.1 Connection to Hierarchical Task Analysis

By far, the probabilistic treatment within mixed-initiative user interfaces framework describes how to decide whether to act or not, but it leaves open the question of how to define what are the possible agent's actions in each situation, and how they are related to uncertainties.

As stated in the previous section, the actions form a hierarchy. To progress in the analysis, it is helpful to think about them in relation to user's task hierarchy. The most specific agent's actions are related to user's detailed-level tasks, and the more general actions to higher-level tasks. If user's activity is seen as a task hierarchy, this forms a natural mapping from agent's task support actions to user's activity.

Hierarchical task analysis (HTA) (*e.g.*, see [17]) as a user research method describes activity as a goal–sub-goal structure. One or multiple plans are associated with goals, each plan describing a sequence of occurrences that need to take place in order to the goal be achieved. The meaning of the term task in this framework is rather vague [17]. In this paper, it is used as a behavioral concept: what is done in practice to achieve a goal. As opposed to a goal, it is something that the agent can perceive. Therefore, from now on in this paper, we will be talking about tasks T rather than goals G because of terminological safety.

Benefits of combining HTA with mixed-initiative user interfaces' probabilistic reasoning can be summarized as follows:

- Having modeled user's activity in detailed level, it becomes easier to say what user's actions (*i.e.*, observed evidence E) are related to which user's task T. That is, it provides the background for defining the probabilities $p(T \mid E)$ that will govern how the agent will be interpreting user's activity. One of the best predictors of user's future activity is very likely user's previous task. This information can also be included in E.
- HTA helps to tell what specific needs the user has related to each task. With this information, it is easier to associate agent actions A with user's tasks T and assign the respective utilities $u(A,T)$ heuristically.
- Knowing the task structure simplifies agent's work of maintaining hypotheses of what the user is most probably doing at each moment, and considering what actions the agent should invoke. This helps in achieving computational tractability, compared to the basic solution that was described in the beginning of this elaboration section.

We now turn to look what the third benefit would bring about and how it could be achieved.

3.2 Flexible Interpretation of User's Activity Using HTA

At any given time, user is performing actions in task hierarchy's leaf level operations. The agent, trying to interpret the task from the evidence it perceives of the activity, has a set of hypotheses $p(T \mid E)$ for tasks T to choose from in different levels in the task hierarchy. Since detailed-level tasks are likely to require more evidence in order to $p(T \mid E)$ be high, a path reaching from the most general task in the hierarchy to an actual leaf-level operation yields a sequence of probabilities in descending order.

If an agent is maintaining an interpretation about what the user is doing at each time, it needs a threshold probability for deciding how detailed interpretation it is willing to adopt, given the observed evidence about the activity. The threshold determines how deep in the hierarchy

the agent can go before the confidence levels $p(T \mid E)$ of the hypotheses drop so low that committing to believe that the user really is doing the tasks cannot be trusted anymore.

Flexible interpretation can now be achieved as a combination of two different ways in this probabilistic framework. First, as time passes and new information about user's activity is accumulated to E, some other task in another place in the task hierarchy may become more probable than the present one, based on how $p(T \mid E)$ for different tasks T change. This way the agent can change its interpretation to the most probable task. It may be a good idea to constrain how far in the hierarchy from the present point of interpretation the new one is allowed to lie, to ensure continuity and reduce the amount of hypotheses to consider.

Secondly, the new evidence may show that the interpretation in the hierarchy may change along the same path to more detailed-level or higher-level descriptions of the same task. The interpretation can be strengthened by committing to a more detailed task, or weakened, which takes it towards more general tasks. By being able to back up to the topmost task and this way reaching any task in the hierarchy, the agent rarely gets trapped in any certain sub-tree.

A potential benefit of having such an interpretation comes from communication possibilities with the user: if the agent can tell what task it believes user is doing, the agent's actions become more intelligible to the user. This is one way how an agent can explain its reasoning to the user.

For finding appropriate actions for supporting the user, a threshold probability may not be necessary, however, since the decision of how to support user can also be based on expected utilities $eu(A)$. Since one of the components for calculating the expected utility is the task's probability $p(T \mid E)$, highly improbable tasks will get a low expected utility, and will thus not be preferred to be invoked. This replaces the need for a designated probability threshold.

Deciding about what actions to invoke is discussed next.

3.3 Supporting User in an Appropriate Way

Figure 2 shows conceptually how the same agent action can help user in multiple tasks. There, action A is associated to a more general level task than B, C and D, which are for more specific support. Each action is associated with a task in a suitable hierarchy level, following the understanding the designer had about user's varying needs in the work process. So, the designer of the system can predefine these actions. Now, in Figure 2, depending on the amount of confidence of user's task, the agent can provide help on two different levels. If $p(T_{1.1} \mid E)$ for task $T_{1.1}$ is high enough, the agent can invoke both A, B and C to help user. If not, the agent may play it safe, invoking only action A.

If computational complexity is not a problem, the suitable actions to support user can be carried out by first finding the most probable leaf-level task (*i.e.*, a task in the bottom level of the hierarchy) that the user is doing. This can be done either (1) with an exhaustive search over all the leaf-level tasks, or (2) progressively by first finding the most probable top-level task, of which the most probable sub-task is searched, and this way proceeding deeper until the agent has a hypothesis for the leaf-level task.

When the task hypothesis is known, a decision must be made what actions to invoke to help in the task. This is done by defining a threshold level for the expected utility that an action

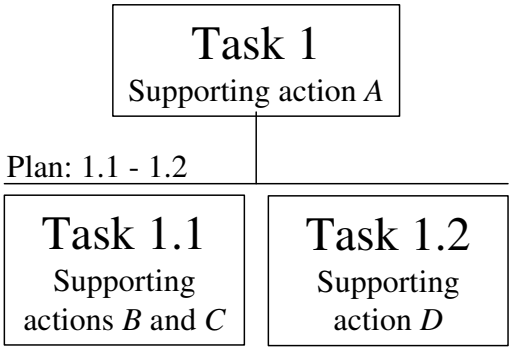

Figure 2. A simple conceptual task hierarchy, each task associated with one or more agent's actions to support the user.

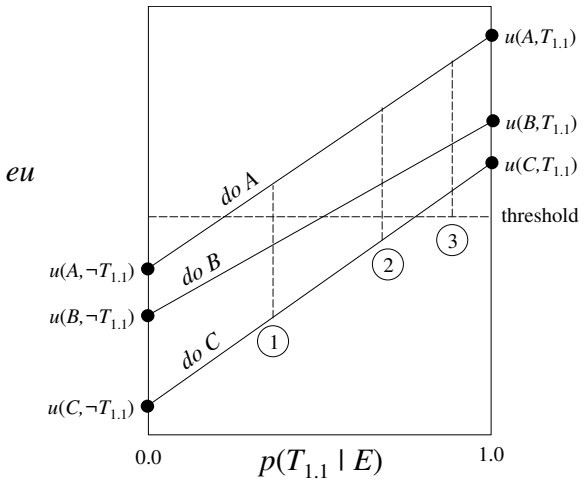

Figure 3. A re-investigation of Figure 1 with multiple actions to consider at the same time, all of them relevant to task 1.1.

must exceed before the agent invokes it. As suggested by Pattie Maes [12], the level could be adjusted by the user, to adapt the agent for personal preferences.

Figure 3 shows how invoking the correct actions would be done in task $T_{1.1}$ (which is a leaf-level task). Let us assume that the agent has found $T_{1.1}$ to be the most probable user's task in either of the two ways: with exhaustive or progressive search. Depending on how confident the agent is of user performing $T_{1.1}$, it may now invoke action *A* only, or actions *A* and *B*, or actions *A* and *B* and *C*. These corresponding possibilities are marked with 1, 2 and 3 in the figure, respectively.

In case 1, agent's confidence of task $T_{1.1}$ is low. The result is that the expected utilities for each action are low as well, which can be seen by noting that the dashed line intersects the lines for actions *B* and *C* below the threshold level. Therefore, in this case only *A* is seen useful for the user.

Cases 2 and 3 show how actions *B* and *C* differ from each other. In case 2, the confidence $p(T_{1.1} \mid E)$ is big enough for the agent daring to invoke also action *B*, and only in case 3 the confidence is sufficient to make all the actions' expected utilities high enough to be invoked.

This way agent's precision of service can be matched with uncertainty, as was asked in principle no. 8 in mixed-initiative user interfaces. Task-analysis based user research helps in

naming different supporting actions for different specificity levels of activity, allowing for support in different ways in different times. This way, a wide range of different proactive behaviors can be provided (see [15]).

3.4 Differences and Extensions w.r.t. Horvitz's Model

Certain differences have now appeared with respect to the model by Horvitz. First, the designer of the system has defined what actions by the agent are useful for each user's task by applying HTA in user research. Horvitz has not suggested a way for doing this, and therefore in his approach, actions and goals have not been linked together a priori. With the help of task analysis, this is possible, thus allowing different ways to assist user depending on the level of confidence.

Second, utility assessment for each action, $u(A,T)$, is the same as in the original model, except for one exception. Like before, the designer must determine two utilities for each action: $u(A,T)$ for p=0.0 (for those potentially harmful situations when action is invoked although the user is not doing task T), and $u(A,T)$ for p=1.0 (when the agent is absolutely sure that the current task is T). The new constraint, originating from the ordering of actions according to the task hierarchy, is that the lines for actions in the same path in the hierarchy must not intersect each other in the graph. In Figure 3 this means that lines for actions *A*, *B* and *C* must not intersect. Otherwise their positions in the task hierarchy (Figure 2) would become insensible: their mutual order would be different depending on the confidence the agent would have about task $T_{1.1}$.

Third, there is more than one action to consider for each goal, and they form a hierarchy, not a flat set of mutually equal choices.

3.5 Assumptions in the Extended Model

The model developed implies the following underlying assumptions about the nature of user's activity:

- *No multitasking.* Task hierarchy as a representation of activity is suitable only for situations when the user does not interlace actions about multiple activities together. The model cannot tell apart these threads of actions.
- *Activity can be easily perceived.* Following the user using a task hierarchy requires that user leaves a trace of actions that can be used as an evidence E for tracking. This excludes mentally heavily emphasized work from the domains that can be tackled with the model.
- *Work is orderly.* If the work is chaotic or otherwise unpredictable, HTA is not an optimal modeling technique.
- *General-level task support can be inherited to support all the sub-tasks as well.* In the presentation above, the higher-level actions are always invoked to support deeper-level tasks. This may not be the case in all situations. For instance, if agent's higher-level action reads "keep lights on", some sub-task may still require absolute darkness. It may therefore be necessary to add overriding rules to the action repertoire.

Assumptions when considering building context-aware systems based on this model are:

- Task hierarchy must be probabilistic. In order to estimate confidences of the tasks, transitions from one task to another must be determined with probabilities, not with rigid ordered arcs like in traditional HTA.
- Agent does not (try to) change user's work routine remarkably. Following the user based on already modeled work procedures requires that agent's helpful actions do not change them that dramatically that the task hierarchy would not be valid anymore. If many of agent's actions are tips, notifications, and other suggestions of new ways of work, user's activity can be heavily altered by those actions. On the contrary, in the case of subtle unobtrusive support this property is not a problem. In spite of all, it may be necessary to take responses to agent's actions into account in modeling user's future actions. Although the activity would change because of the context-aware system, the service provided by the system exhibits graceful degradation, since reasoning on higher-level tasks is likely to stay valid longer.
- When creating the task hierarchy through user observations, also contextual information related to each task must be collected. This information is needed for developing the probabilistic model and should cover the same information channels that the system can perceive and use in real-time settings. The information may include user's actions, agent's actions that make a change in the context, and any other events in the context from extraneous sources (*e.g.*, weather conditions etc.). Looking this from context modeling perspective, the task hierarchy can then be considered an activity-centered layered model, where each layer describes the context in increasing situation specificity.

In the following sections, the model is evaluated against empirical findings of a work domain that fulfills most of the implied assumptions. Then in the discussion, some of the particularly tricky characteristics in human behavior are discussed.

4 Reasoning about Elevator Maintenance Work

Our team studied elevator maintenance work in a project that aimed for identifying benefits of context awareness and proactive computing in service work by applying user-centered design methods to the problem. For ubiquitous and context-aware computing research, service work is an interesting domain because of the reasons listed below.

- *Maintenance activities can be made perceivable to machines.* This is not easily possible in many other contexts of work. Elevators are both mechanical and electronic devices containing a wide variety of technologies. In the future, they can be equipped with sensors, which makes tracking operations and task sequences possible.
- *The work spans many contexts.* Because of installations in different buildings, every maintenance visit is carried out in a different physical context. Repair tasks are time-critical work, which poses problems since moving between sites takes up a lot of time. These properties together make service work an interesting field to study also from a mobile HCI point of view.
- *Activity takes place in natural settings.* Bringing context-aware computing research out from laboratory settings is important for the ecological validity of the whole field. Interactive spaces in laboratories are always contextually more or less artificial. By studying also real work we can get ideas on how to cope with the complexity in real life.

- *Service work is a new design domain in context-aware computing.* By far, ubiquitous and context-aware computing has not been studied in relation to service work, although the above-listed reasons make it an appealing research subject. Ethnographic studies on service work have been conducted by Barley [1], Orr [14] and Yamauchi *et al*. [19] and to some extent within mobile HCI research (see [20] for a summary), but they have not been directly connected to context-aware computing.

Work was studied using following techniques:

- *Contextual inquiry.* 3 workers were studied during their daily work. Contextual inquiry is a method combining observation and interviewing in user's real context, and is a part of Contextual Design developed by Beyer and Holtzblatt [2]. In addition to seeing the main activities of service work in the first person, photographs on tools in use could be collected.
- *Photograph-based artifact analysis.* 4 workers were asked to take photos of their tools during their daily work. The films were developed and then an interview was arranged where the workers discussed their shots in pairs. Discussions were taped and transcribed onto paper, and were but once organized as paired interviews [13]. The outcome was then used to supplement direct observations from contextual inquiry.

4.1 Work Structure

Following factors shape the activity in elevator maintenance work:

- Machines are located in different buildings within a territory that the worker is responsible alone. Maintenance requires a lot of driving from a site to another. At the elevator, the work has to be done in the presence of customers. The work context is therefore both mobile and dynamic.
- Safety regulations must be adhered to, to avoid injuries to the worker or the customers when the machine is in use or under maintenance. Elevators have also many in-built mechanisms for coping in faulty situations.
- Visits are further divided into routine maintenance and time-critical repair visits. Elevators are visited routinely to proactively prevent faults that put the system out of order. Routine maintenance visits are scheduled in advance, frequencies depending on the amount of use and conditions at the location. Repair visits, however, are always unpredictable and have to be reacted immediately, often interrupting a routine visit.

Due to these reasons, maintenance visits often follow the same procedure. The worker either selects a site from a list of to-be-visited elevators, or receives a message about a time-critical repair task. He then drives to the location, does either the routine maintenance or solves the fault (which may sometimes be difficult and require spare parts), leaves and sends a report about the visit. Then the sequence is repeated. At the location, the tasks follow a routinized order which increases the quality of service. The work is not however completely prescribed but allows for variation, depending on the situation and amount of hurry. That is, the work is orderly in large scale.

Figures 4 and 5 show a task hierarchy of the work. For the sake of presentation, only those subtrees have been decomposed that are of interest in the following sections. Certain tasks have been equipped with actions that a supporting system could invoke. They will be considered in the use cases below.

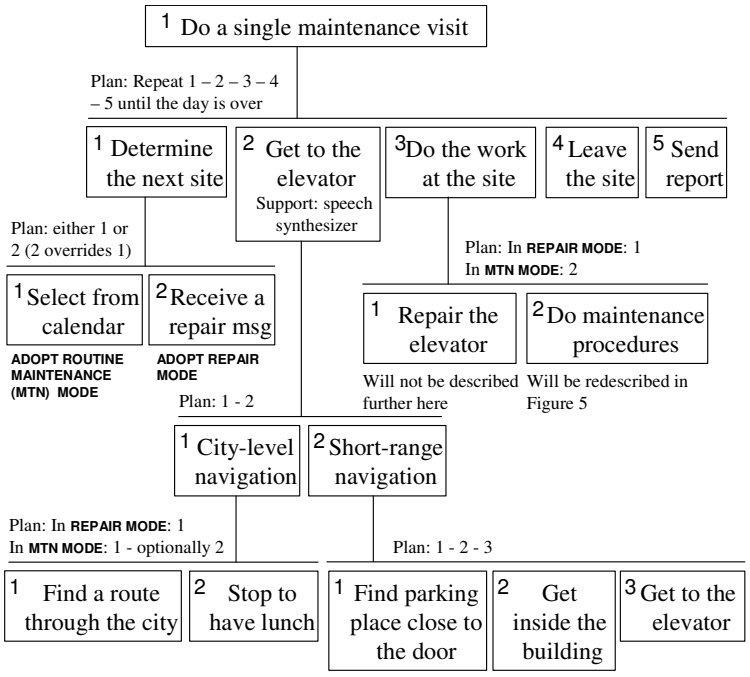

Figure 4. Hierarchical task model for a single maintenance visit.

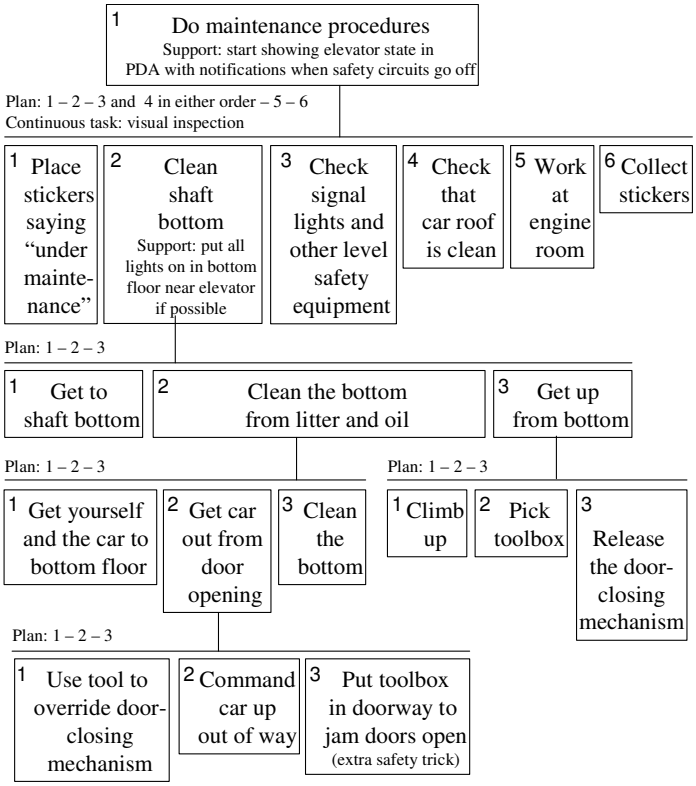

Figure 5. Routine maintenance procedures redescribed from Figure 4.

Let us now suppose that the present work would be supported with a proactively working context-aware system that knows the work structure like in Figure 4 and 5. Its sensor data would consist of data in worker's electronic calendar (which they will eventually have in a PDA), the states of elevator's safety circuits (inspectable already now at the engine room),

and location sensing based on GSM cells and within-building beacons of 1-meter accuracy near the elevator. All of these are quite moderate requirements.

The three cases below are accounts from real observations and have features that would pose problems for a context-aware system interpreting user's activity. An analysis shows how the proposed probabilistic model would cope with the situations. The first case presents a system for supporting worker's awareness of the state of the elevator, and the second a system for facilitating problem-solving.

4.2 Case 1: Unintentional Touch on a Safety Switch

In this case, workers' PDA would have a program for context interpretation, and it would proactively notify user about system-related states probably of interest.

The observed situation was following:

> The worker (named John here) was doing routine maintenance to a two-year old elevator. As he was checking the amount of oil in the guide bars in the shaft bottom, he noticed by looking at the guiding wheel's position that the steering cable (which determines the halting position of the car) had stretched a bit. The actual cable-shortening procedure would have taken about an hour of precious time, so this time John just decided to adjust the guiding wheel into a lower position, thus tightening the cable. John unscrewed the brackets and knocked the wheel a bit with a jaw spanner. The whole procedure took about 5 minutes.
>
> John continued with cleaning and checking lights. Finally he switched the elevator back into operation and ordered it to take him up to the engine room in the top floor. The elevator did not however start moving, and John began to debug for reason: he remembered that he had not touched the safety switches while adjusting the wheel, he checked that no doors were open and that the switches on the top of the car were in the right configuration. These were fast checks that usually rule out the most obvious reasons for the problem.
>
> John climbed the stairs to check error codes in the engine room's control board. A code told that he had anyway touched the safety switches near the wheel. He confirmed the error message and the elevator started working again.

The PDA, equipped with sensor data of worker's location and safety circuit state, would in the beginning of the story have held shaft bottom duties as the strongest interpretation, due to knowing from the calendar that the visit was about routine maintenance and that the worker was at shaft bottom. It would have not anticipated the quick repair task that the worker started doing that caused the safety circuit go off. The event would have been notified to the worker via PDA. Since touching this particular safety switch is not unusual while working in the shaft bottom (since the switch and the worker are close to each other), the obtrusiveness of the notification could have been adjusted appropriately to a mild level.

4.3 Case 2: Deciding to Visit an Unexpected Site

Here, the system would be installed in the worker's car, to be used while driving. Knowing the site to visit next from the calendar data, the system would provide suitable information from the machine's log (which already now is transmitted via Internet to a central server) using a speech synthesizer and statistical analysis of likely most helpful information. This kind of information given in advance would help in debugging the fault as soon as the worker would reach the site.

The worker had agreed to meet people at a construction site at 13.00 but it was only 12.15. He noticed that he could change the route a bit and visit an office building whose elevator had required extra care during the last month. There was just enough time to do a new routine maintenance. Originally, he had planned to visit this elevator in the end of the week.

When finding out from GSM positioning data that the route in the city deviates too much from the expected track, the system providing information through loudspeakers should ask whether it should terminate its recounting, or change the elevator of interest.

In the beginning of the story, the most likely interpretation would be that the worker is doing the city-level navigation task, heading to the construction site. At some point the confidence for the navigation task would drop and the expected utility of providing information about right elevator would sink below the threshold level. The user could be asked whether he still would like to listen. With GSM-based positioning, inferring the correct elevator would probably not be possible before the worker would move to short-range navigation task, so shutting down the synthesizer is the most realistic scenario.

The analyses her show that the extended model seems to be feasible for reasoning about the user's situation and providing context-aware support.

5 DISCUSSION

The problem of understanding human behavior computationally can be a result from (1) the difficult-to-analyze characteristics in human behavior, (2) theoretical challenges in developing suitable computable mathematical models, and (3) practical difficulties in collecting noise-free, accurate and informative real-time data in the use context. In the following sub-sections, the first two of the aspects are discussed more thoroughly. The third one is out of scope of this paper, since it is mostly a technological problem and does not touch user modeling.

5.1 Intricacies in Human Behavior

Human behavior is known to exhibit following characteristics, among others:

Multi-tasking: The user may be interlacing multiple threads of activity that may be directed to achieve both the same or different goals.

- *Situatedness:* The context poses both a constraint and a resource for carrying out the actions in different ways in different situations. The plan for producing the actions is fixed neither, but acts also as a resource for the user, determining the activity only in a large scale [18].
- *Opportunism and creativity:* Humans are creative in finding new opportunities in doing things more efficiently. They may find new uses for a tool or change plans abruptly if new ones seem easier and more suitable [5].
- *Ambiguity:* The same action sequence can mean different things at different times, and vice versa: different strategies can be used to achieve the same goal [11].

If the activity includes cooperation with other people, as very often is the case, a social layer is added to the problem. In that case a big part of information created in the activity is based on language and subtle cues and is thus difficult to capture and analyze by machines. When in collaboration, people also establish a continuous error correction procedure between each

other to ensure intersubjective understanding of each other's intentions [4]. This is also very difficult to implement in computers.

From these, let us analyze multitasking more closely, since it is clearly in conflict with HTA's single task decomposition approach, and is thus specifically interesting to the model presented in this paper.

5.2 Multi-Tasking at a Closer Inspection

What makes multi-tasking difficult for activity interpretation is that it interlaces evidence about multiple tasks onto the same timeline. If the multiple tasks have not been expected to occur simultaneously and if they do not have strong interdependencies that would bring order between the events, trying to fit the observations to a single task hierarchy will not yield good results.

As a general solution, asking the user about his or her task is out of question because of many reasons. First, it is obtrusive and therefore against the purpose of context-aware systems. Second, the context may make it impossible to get an answer from the user. Third, task switches can be so fast-paced that communication is not possible.

Multi-tasking can be seen as a process that adds a new layer in front of the proposed probabilistic approach, by mixing the inputs. If the observations could be pre-processed and directed to task-specific threads, the interpretation problem would be transformed back to its original setting, to be separately carried out for each thread. There are however certain problems. In real life, a user can use a single action to progress multiple tasks at the same time. In addition, users start new tasks and terminate old ones asynchronously, and the changes in the number of threads can be difficult to tell, making it hard to direct events neatly. Let us however continue the investigation by omitting these possibilities from consideration.

Cognitive psychology tells that user cannot pay attention to but one complex task at a time. Therefore, knowing where attention was focused by the time of the action can help to tell what the task was at that time. There are some ways to do this:

- Equip tools with sensors that can provide real-time data about their use. The tool is a mediator between the user and the object of activity, and thus an indicator about his or her task. Sensing about tools is therefore probably more useful than sensing about temperature, for instance.
- Use gaze-tracking to see where the user is looking. This is a very effective way to infer about attention [16], but its use is limited to very controlled use contexts.

A good array of sensors can produce synchronous data patterns that can be identified and analyzed separately, since they are probably related to the same task. This can be a useful pre-processing strategy in domains where gathering data through many channels is possible, the tasks are initiated and terminated sharply, and the sensor data is free enough from noise and other distractions.

As a summary, telling unrelated actions apart is a hard problem and is a general problem, not only specific to HTA-based interpretation approaches. Well-working solutions to the problem do not exist, but if multi-tasking is known to be a prevalent characteristic in the context, a general strategy to follow could be to design the tools for different tasks to provide

real-time data via different channels, and then use algorithms to separate those channels according to their patterns.

5.3 Mathematical and Practical Challenges

HTA has been generally shown to be a well-working technique for describing activity, but deploying it in probabilistic (or stochastic) modeling can be problematic because of its rigidity of representation.

The earlier work of Eric Horvitz *et al*. in Lumière Project [7] (of which eventual outcome was the MS Office Assistant) used dynamic Bayesian networks for temporal reasoning of user's goals. The prototype was used to provide assistance for MS Excel spreadsheet operations. The domain was more knowledge work oriented than the mobile elevator maintenance work presented in this paper, and therefore the relations between actions and corresponding goals were difficult to model. Because the activity could not be captured into procedures and hierarchies, also the relations between goals and the appropriate help needed were hard to assess. The solution by Horvitz *et al*. was to videotape real use situations and use experts to infer what could be the goal in each situation, having seen only the interaction in the videotape. Both the users' and the experts' reasoning was captured through talk-aloud thinking. With this information, the relevant variables and conditional probabilities in the Bayesian network were assigned. This was difficult and required a lot of effort. The parts of the network shown by Horvitz *et al*. resemble HTA but are more complex.

In domains where HTA or other similar task representation techniques can be applied, assigning probabilities may possibly be simplified. The steps in the process are (1) using HTA as a backbone and extending it with other necessary influencing factors, (2) thinking what context and activity variables will be available in mobile situations, and (3) bootstrapping the model's probabilities from real observations. In Lumière project, the influencing factors that extended the activity-related nodes in network included user profiles and task difficulty measures. These factors are valid also in mobile design domains.

In mobile situations, technology often determines what contextual variables are available for context-aware reasoning. Nowadays the typical sources of information are GSM-based location coordinates and a list of wireless services in the surroundings, and in what state each of these services is. These are the evidence *E* that will be used for teaching the model and reasoning about the activity. Also latent variables can be added to the model to describe factors that are not directly perceivable. Decay properties for variables are also needed [7]. That is, if something has happened a minute ago, it may be less significant than if it had happened a second ago. It may also happen that the variable's significance does not have any decay. For instance, the mode variable in Figure 4 persists throughout a single site visit. Mode variables are necessary also because basic Bayesian networks cannot model repetitions and loops.

The third step listed above is the calculation of the values in conditional probability tables, to name each variable's overall significance in reasoning. A possible way for doing this is to first videotape activity and then manually code it into a temporal data series that tells the tasks and sensor data available at each point of time. The probability tables can be calculated from this information automatically.

Coding videotape is however very time-consuming. In a study about the work in a cell biology laboratory, Consolvo, Arnstein and Franza used a technique called lag sequential analysis

(LSA) for analyzing the work quantitatively [3]. They watched for 23 different variables in the videotape, including personnel movements, tool use and body positions. Having 18 hours of video, it took 40,5 hours for training people to ensure inter-person comparability, and 85,5 hours for actual coding. They used the coded data to evaluate their ubicomp prototype in the laboratory, and not for creating activity models, however. In our case coding will probably take more time, since Consolvo *et al*. used one-minute time slices, which is too coarse for the purposes in this paper.

Although coding is tedious, it may pay back later. Having enough data available, it can be used both for teaching the model (as was discussed above), and also for evaluating the result. In this case part of the data is excluded from teaching phase and is spared for simulating real use situations in the new model, allowing for investigating model's tracking capabilities. This can be helpful in many ways. It allows for experimentation with the network structure, decay functions and the necessary input channels that are needed for satisfactory interpretation, and their influence on computational complexity. Experimentation can be carried out computationally without a need for building prototypes, which saves time and money.

As a summary, creating a probabilistic model based on the information from HTA is theoretically possible, but cannot be automatized. Effort is needed in extending the hierarchy with extra nodes and constructing data in order to automatically assign the probabilities in each network node.

6 Related Research

Activity interpretation and goal recognition in HCI have been approached in many ways over time. This section provides a look on related hierarchical approaches.

Kuenzer *et al*. [9] have compared six different hidden Markov model topologies (special cases of dynamic Bayesian networks), most of them hierarchical, to interpret a GUI interaction task. The task had a hierarchical structure, but although a HTA was conducted to describe the task, it was not used for teaching the model. Instead, the models had 34 hidden states (input variables) on each level, representing the possible user interactions. That is, there were 34 different commands constantly available to the user in the user interface. The results showed no significant differences between the models in predicting users' actions. On average, predictions were correct in 46% of the cases, while pure guessing would have resulted in 1/34 ≈ 2,9% accuracy. Kuenzer *et al*. did not report whether the models would have been light enough for real-time use. With the arrangement of 34 hidden states, the sizes of the transition probability tables varied between 103 and 105 depending on the model, which means that the models could have been too heavy for real-time use.

Other approaches to interpreting user's activity hierarchically include the works by Hoppe [6] and Lesh *et al*. [10]. Both are based on modeling interaction using production rules and thus they are not probabilistic models. Hoppe's approach was based on finding the best fit between an interaction sequence and hierarchical task descriptions. The domain was Unix command prediction. Hoppe does not provide results with real user data, and therefore the model's feasibility is difficult to assess. The other production rule system was by Lesh *et al*. for collaborative agent-user problem-solving, and is based on production rules called recipes. Because their model relies on continuous turntaking between the agent and the user, goal prediction accuracy is not possible to evaluate due to the human component in the model. In

addition, the approach would not fit for the purposes of this paper, since it is not designed for unobtrusive inference “over the shoulder”.

7 Conclusions

This paper started with a short introduction to probabilistic reasoning about user goals in mixed-initiative user interfaces. Then, needs for further investigation were identified regarding how the model should allow for user support in more than one way in a situation, and what supporting actions and user tasks should be tied together. As a potential solution, hierarchical task analysis was suggested, high-lighting its capabilities for describing activity in different layers of specificity. This property allows for identifying means of support that vary from very general actions to very specific ones, thus creating a natural mapping between the uncertainty of interpretation and generality of task support. This mapping was used for extending the original decision-theoretic approach.

Using HTA in extending the model implies that it is most likely suitable for domains where activity is orderly and goal-directed, and system’s proactive support does not change user’s activity to the extent that the description created using HTA would not hold anymore. The extended model seems applicable for mobile contexts where it is typical that activity may not be observed with an extensive number of sensors. This was confirmed with a conceptual evaluation against observations from elevator maintenance work. After that, questions of multi-tasking and practical issues of model construction were discussed.

8 Acknowledgments

The empirical part of this study was funded by The Finnish Work Environment Fund, Kone Corporation and Finland’s Slot Machine Association. Other participants in addition to the author were Hannu Kuoppala, Sirpa Riihiaho, Petri Mannonen. Wendy Mackay, Antti Oulasvirta, Miikka Miettinen and Mika P. Nieminen provided valuable input in different stages of developing the ideas and writing the paper.

9 References

1 Barley, S.R. Technicians in the workplace: Ethnographic evidence for bringing work into organization studies. Administrative Science Quarterly 41, 3 (1996), 404–441.

2 Beyer, H. and Holtzblatt, K. Contextual design: Defining customer-centered systems. Morgan Kaufmann, San Francisco, CA, USA, 1998.

3 Consolvo, S., Arnstein, L. and Franza, B.R. User study techniques in the design and evaluation of a ubicomp environment. Proc. Ubicomp’02, Lecture Notes in Computer Science 2498, 73–90.

4 Heritage, J. Garfinkel and ethnomethodology. Polity Press, Cambridge, UK, 1984.

5 Hollan, J., Hutchins, E. and Kirsh, D. Distributed cognition: Toward a new foundation for human–computer interaction research. ACM Transactions on Human–Computer Interaction 7, 2 (2000), 174–196.

6 Hoppe, H.U. Task-oriented parsing – A diagnostic method to be used by adaptive systems. Proc. CHI'88, ACM Press (1988), 241–247.

7 Horvitz, E., Breese, J., Heckerman, D., Hovel, D. and Rommelse, K. The Lumière project: Bayesian user modeling for inferring the goals and needs of software users. Proc. UAI'98, Morgan Kaufmann (1998), 265–265.

8 Horvitz, E. Principles of mixed-initiative user interfaces. Proc. CHI'99, ACM Press (1999), 159–166.

9 Kuenzer, A., Schlick, C., Ohmann, F., Schmidt, L. and Luczak, H. An empirical study of dynamic Bayesian networks for user modeling. Paper presented at Workshop on Machine Learning for User Modeling in UM'01. Sonthofen, Germany, July 13, 2001.

10 Lesh, N., Rich, C. and Sidner, C.L. Using plan recognition in human–computer collaboration. Proc. UM'99, Springer-Verlag (1999), 23–32.

11 Mackay, W.E. Which interaction technique works when? Floating palettes, marking menus and toolglasses support different task strategies. Proc. AVI'02, ACM Press (2002), 203–208.

12 Maes, P. Agents that reduce work and information overflow. Communications of the ACM 37, 7 (1994), 31–40 and 146.

13 Mannonen, P., Kuoppala, H. and Nieminen, M.P. Photograph-based artefact analysis. Proc. Interact'03, IOS Press (2003), 833–836.

14 Orr, J.E. Talking about Machines: An ethnography of a modern job. ILR Press, Ithaca, NY, USA, 1996.

15 Salovaara, A. and Oulasvirta, A. Six modes of proactive resource management: A user-centric typology for proactive behaviors. Proc. NordiCHI'04, ACM Press, 57–60.

16 Shell, J., Selker, T. and Vertegaal, R. Interacting with groups of computers. Communications of the ACM 46, 3 (2003), 40–46.

17 Shepherd, A. Analysis and training in information technology tasks. In Diaper, D. (ed.), Task Analysis for Human–Computer Interaction, Ellis Horwood (1989), Chichester, UK, 15–55.

18 Suchman, L.A. Plans and situated actions: The problem of human–machine communication. Cambridge University Press, Cambridge, UK, 1987.

19 Yamauchi, Y., Whalen, J. and Bobrow, D.G. Information use of service technicians in difficult cases. Proc. CHI'03, CHI Letters 5(1), 81–88.

20 York, J. and Pendharkar, P.C. Human–computer interaction issues for mobile computing in a variable work context. Intl. J. of Human–Computer Studies 60, 5-6 (2004), 771–797.