Euterpe - Tool support for analyzing cooperative environments

Martijn van Welie, Gerrit C. van der Veer, Anton Eliëns

Department of Computer Science, Vrije Universiteit
de Boelelaan 1081a, 1081HV Amsterdam, Holland
{martijn, gerrit, eliens}@cs.vu.nl
http://www.cs.vu.nl/~martijn/

ABSTRACT

This paper describes a tool - EUTERPE - that offers support for analyzing cooperative environments. The support is based on a formal analysis of the task model and can be done both on a logical and a visual level. An analysis of a cooperative environment requires a model that can formally describe the task world and that allows meaningful analysis to be done. Euterpe uses a logical model that is based on Groupware Task Analysis that describes the task world including cooperative aspects. By modeling the task world in logic and deriving graphical representations from it, several ways of analysis become possible.

Keywords

Tools, Cooperation, Task Analysis

INTRODUCTION

Task analysis is useful way for getting better insight into cooperative environments. However, it is often also a very unstructured and time-consuming activity. Many methods exist, but thoughts on task models and what they describe exactly have not been stabilized yet. Furthermore, task analysis methods usually only deal with task *modeling* and not really with task *analysis*. After the task world has been modeled it is up to the analysts to interpret the task model and find out where causes of problems can be found or where there is room for optimization of the work. These may be one of the reasons that cause task analysis to be both ineffective and inefficient.

A task model that can describe the task world including cooperative aspects and that allows some form of analysis could improve the task analysis process and outcome. Preferably the analysis of the task model should be done (semi-) automatically, thereby reducing the required effort of the analysts. In (d'Ausburg et al., 1998) a formal approach based on model checking techniques is described for analyzing user interfaces. A similar approach can also be applied to analyzing task models. However performing a formal analysis of a task model requires a formal representation of the task model that is suitable for doing an analysis, especially for analyzing cooperation. The task model therefore needs to be based on a task analysis theory that recognizes the cooperative aspects of the task world.

Although a formal analysis can be the basis for analysis it is not on the level analysts prefer to work. Hence representation tools can effectively hide the formalism and provide means to assist in analyzing the environment that is being studied. In addition, a tool can also provide more structured ways of doing task analysis. The next sections describe such a tool - EUTERPE - based on Groupware Task Analysis that supports formal analysis both on a logical and a visual level. Both the used models and the analysis primitives will be described in the next sections.

ANALYSIS TOOLS

In the area of task analysis or requirement engineering (Loucopoulos et al., 1994) there are many techniques and to some extent tools that can be used. Especially in the first phase of collecting information and representing structures not many tools turn out to be useful or available for actual use. An interesting tool however is for instance U-TEL(Chung-Man et al., 1998) which is a tool that assists in elicitation of user task models from domain experts by natural language processing and wizards. It is part of the model-based user interface development environment MOBI-D(Puerta, 1997). However, the task models that can be specified are rather simple and do not allow roles and responsibilities to be specified nor are any analyzing primitives provided. In fact they are not intended for describing cooperative aspects of the task world. Some commercial software tools such as WinCREW and the Observer (Noldus, 1991) also exists. WinCREW is a tool for analyzing the behavior of small tank crews and the Observer for human behaviour analysis based on video analysis. Although they were intended for analyzing cooperative environments they use mostly statistical methods. However for analyzing a cooperative environment a formalism is needed that explicitly recognizes and describes the cooperative aspects of the task world. Such a formalism would consequently allow more deeper analysis than just statistical evaluation.

GROUPWARE TASK ANALYSIS

In the past task analysis has focussed mainly on analyzing a single user and his/her tasks. Groupware Task Analysis (Veer et al. 1996) expands task analysis by looking at a task world from the perspectives of work, agents, and situation. It regards the task world as an organization where many people do tasks, work together and interact with both people and objects. From the perspective of **work** GTA looks at tasks, goals, actions and procedures within an organization. On the other hand from the perspective of **agents** GTA looks at the persons and machines that perform the work, what their role is in the organization and how responsibilities are allocated. When looking from the perspective of **situation**, the static and dynamic aspects of the task environment are studied. This includes the objects that are present and events that happen internally or externally.

In (Veer et al., 1996b) we have presented a field study where GTA was used to analyze a cooperative environment, in this case the social security administration. These perspectives have been elaborated formally into an ontology (van Welie et al., 1998) for describing a task world. The ontology describes how GTA looks at the task world independently of graphical representations, by describing the concepts and the relationships between them.

EUTERPE

Our task analysis tool - EUTERPE -, named after one of the 9 muses from Greek mythology presiding over the arts and sciences, is a graphical tool that can be used to enter task analysis data and to analyze them. It uses the ontology as the basis for generating representations. The ontology is operationalized using DLP (Eliëns, 1992) an object oriented variant of Prolog. The ontology is specified in terms of concepts such as task, object, role, agent, event and relationships between them, represented internally using a logical programming language. The logical representation is on an abstract level and it does not imply any graphical representation. However, it is rich enough to accommodate the extraction of the information necessary to generate commonly used representations such as tree structures, process flow graphs or templates. The task world ontology is a model that describes a way of looking at a task world. We look at the task world in terms of a number of concepts that are related to each other. The following section will briefly explain the concepts and their relationships. A more detailed description can be found in (van Welie et al., 1998).

Concepts and Attributes

This section will define the concepts and the next section will define their relationships in detail.

Object. An object refers to a physical or non-physical entity. A non-physical entity could be anything ranging from messages, passwords or addresses to gestures and stories. Objects have attributes consisting of attribute-name and value pairs. What can be done with an object is specified by actions, for instance *move*, *change*, *turn off* etc. Furthermore, objects may be in a type hierarchy and can also be contained in other objects.

Agent. An agent is an entity that is considered active. Usually agents are humans but groups of humans or software components may also be considered agents. Agents are not specific individuals (like Chris) but always indicate classes of individuals with certain characteristics.

Role. A role is a meaningful collection of tasks performed by one or more agents. A meaningful role is *responsible* for the tasks that it encompasses and roles can be hierarchically composed.

Task. A task is an activity performed by agents to reach a certain goal. A task typically changes something in the task world and requires some period of time to complete. Complex tasks can be decomposed into smaller subtasks. Tasks are executed in a certain order and the completion of one task can *trigger* the execution of one or more other tasks. A task could also be started because of an event that has occurred in the task world.

Important for the task concept is the distinction between unit tasks and basic tasks, where (ideally) a unit task should only be executed by performing one or more basic tasks. The relationship between the unit task and basic task is interesting because it can indicate the problems that an agent may have in reaching his goals.

A unit task is defined by Card, Moran and Newell as the simplest task that a user really wants to perform. A basic task is a task for which a system provides a single function. Usually basic tasks are further decomposed into user actions and system operations. A user action is an action done by the human users that is only meaningful in the context of its basic task (e.g. a key press). A system operation is an action done by a system; it is not a typical task because it, as such, serves no goal for the user.

Event. An event is a change in the state of the task world at a point in time. The change may reflect changes of attribute values of internal concepts such as Object, Task, Agent or Role or could reflect changes of external concepts such as the weather or electricity supply. Events influence the task execution sequence by *triggering* tasks. This model does not specify how the event is created or by whom.

Relationships

The concepts defined in the previous section are related in specific ways. In this section we sketch the relationships. For each relationship the first-order predicate definition is given and explained. Figure 1 shows all the concepts and relationships together in a diagram.

Uses. The uses(Task,Object,Action) relationship specifies which object is used in executing the task and how it is used. The Action specifies what is being done with the object. It typically changes the state of the object.

Triggers. The triggers(Task/Event, triggeredTask, triggerType) relationship is the basis for specifying task flow. It specifies that a task is triggered (started) by an event or a task and the type of the trigger.

Several triggertypes are possible such as OR, AND, NEXT to express choice, parallelism or sequences of tasks.

Plays. Every agent should play one or more roles. The plays(Agent, Role, Appointment) relationship also indicates how this role was obtained. Currently, the Appointment parameter can be ASSIGNED, DELEGATED, MANDATED or SOCIAL.

Performed_by. The relationship performed_by(Task, Agent/Role) specifies that a task is performed by an agent. This does not mean that agent is also the one who is responsible for the task because this depends on his role and the way it was obtained. When it is not relevant to specify the agent that performs the task, a *role* can also be specified as the performing entity.

Subtask. The subtask(Task, SubTask) relationship describes the task decomposition.

Subrole. The subrole(Role, SubRole) relationship brings roles into a hierarchical structure. The subrole relationship states that a role includes other roles including the responsibility for the task that encompass the role. When a role has subroles the task responsibilities are added up for the role.

Responsible. The responsible(Role, Task) relationship specifies a task for which the role is responsible.

Used_by. The used_by(Object, Agent/Role, Right) relationship indicates who used which object and what the agent or role can do with it. The agents' rights regarding objects can be of existential nature (CREATE and DESTROY), indicate ownership (OWNER), or indicate daily handling of objects (USE, CHANGE).

The relationships of this model form a minimal set of relationships that exist. However, when using this model there are also other relationships that can be of interest. Consider for instance a relationship *involved_role* that indicates which roles are involved in a task. Such a relationship could be defined as the roles of the agents involved in the task and all the involved roles of the subtasks. The *involved_role* relationship is not part of the ontology because it can be defined using only the relationships of the ontology.

Deriving Graphical Representations

EUTERPE offers several representations that are all generated from the same data which guaranties consistency among the different representations. Representations include task trees, object hierarchies, templates with detailed information about entities such as tasks and objects, entity list and process graphs. Kaindl (1993) discussed the importance of availability of task analysis documentation to clients or project members. He suggests the use of hypertext documentation. EUTERPE can therefore generate HTML documentation that shows all the data including links between entities. In order to incorporate ethnographic data it is also possible to attach video fragments, images or sounds to any of the entities.

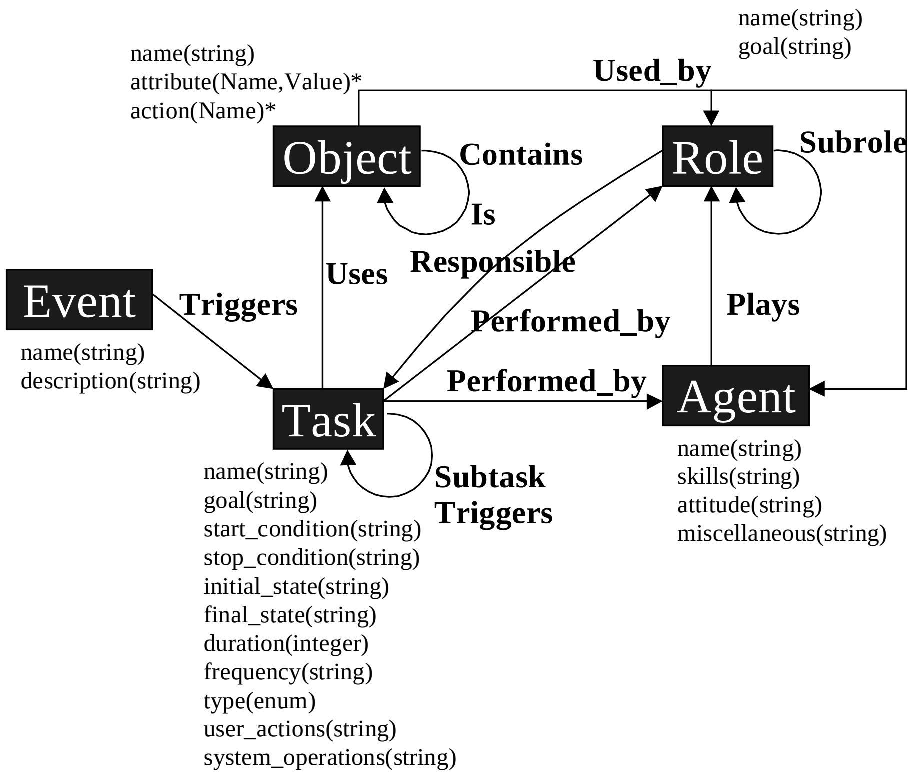

Figure 1. The concepts and relations

A NEED FOR FLEXIBLE EPRESENTATIONS

When designers using EUTERPE needed to explain the problems they found in their analysis, they often used to color certain nodes in task trees to indicate problem areas. Reasons for coloring were often closely related to attributes of entities such as tasks and objects. Common remarks were that certain tasks take too much time or happen too often. Other remarks were that certain people do things that they officially are not allowed to do or that they do thing with objects for which they officially did not have the appropriate rights. We found out that especially exceptions to the "official way of working" gave interesting information about the task world. Most remarks were things that could be "detected" by logic expression on our used model. This led to the addition of analysis primitives to EUTERPE.

ANALYZING COOPERATION

The basis of EUTERPE is that the theory of GTA is formalized into the ontology and that the ontology is represented as a first order predicate logic. The representation of an ontology in logic allows us to analyze the task world in all its facets, the people with their work and the organization they are part of. One criticisms on task analysis has always been the fact that it remained unclear what exactly to do with the data, "we have the data now what?" What should be next is an analysis of the data, finding problem areas and designing a "New World" that relieves these problems. The analysis that is usually done has an informal character and is usually based on insight on the data. However, we found that some problem areas have a more general nature which are domain independent.

- **Problems in individual task structures.** The task structure is sub-optimal because too many subtask needs to be done or certain tasks are too time-consuming or have a high frequency.
- **Differences between the formal and actual task performance.** In cooperative environments, usually regulations and work practices exist which are documented, for instance as part of ISO9000 compliance. In reality tasks are mostly not performed exactly as is described on paper and that "one way" of how the tasks are done does not exist. When persons in a cooperative environment think differently about what needs to be done, problems arise.
- **Inefficient interaction in the organization.** Complex tasks usually have many people involved who need to communicate and interact for various reasons, such as knowledge about tasks or responsibility for tasks. This can be the cause for time-consuming tasks but also for irritation between interacting people.
- **Inconsistencies in tasks.** Tasks are defined but not performed by anyone or tasks are executed in contradictory order.
- **People are doing things they are not allowed to do**. In complex environments often people have to do tasks for which they did not get a official permission or they are using/changing objects they are not allowed to change.

Of course not all of these problems can be automatically detected. However using our model for describing task world models many characteristics can be detected semi-automatically by providing the analyst with a set of analysis primitives. Analyzing a cooperative environment can be done when the data present in the model is transformed into qualitative information about the task world. EUTERPE basically has two primitives of qualitative analysis. First of all visually in graphical presentations. When the data has certain features, these can lead to modifications of the graphical representations. The second primitive is to analyze the data on a logic level by putting some constraints on the model. Constraints that cannot hold may show interesting features of the task world. These two primitives allow several ways of analyzing a task model. We distinguish four ways: *inspection*, *analysis*, *verification* and *validation*. In the next sections these ways of analysis will be elaborated and clarified with examples.

Inspection

Inspection means browsing through your data. A task model based on the ontology is a complex model. In projects done by designers the task models typically consists of about 100 tasks, 20 object, 15 roles, 10 events and 10 agents. This a lot of information that needs to be understood. Graphical representations in general show specific aspects of the data, for instance a tree shows the hierarchical structure of tasks. Other useful representations include flow graphs, interaction diagrams, templates and hyperlinked structures. Euterpe offers several of these representations and provides a coherent and consistent view on the data.

Additionally a coloring mechanism can be using to tune the graphical representations e.g., the coloring of nodes in a tree can be used to analyze task/agent allocation. The user can specify a condition for coloring of a node, for instance "all tasks performed by Chris". The user can choose from a range of predefined conditions of specify the condition in logic. Conditions can be arbitrarily complicated and range from showing task/agent allocation to showing instances of delegation of task responsibility. Figure 2 shows an example of a task tree with colored nodes. Another possibility is browsing through the concepts and seeing their details and relationships for instance by following links in the HTML representation or viewing templates.

Analysis

Whereas inspection is merely "looking at" analysis is "finding out what is going on". Here the goal is to gain understanding of the task world and to find the nature and causes of problems. This can be achieved by using several different representations like those used in inspection and by using certain derived characteristics. For instance, coloring all tasks in which a certain role is involved may help to gain insight in the involvement of a role in the task structures. Euterpe has built-in characteristics that can be checked on request but for the advanced analyst it is also allowed to specify additional characteristics. Some examples of predefined characteristics are:

- `agent X:`
  `all tasks performed by agent X`
- `cooperative task:`
  `all tasks where more than 3 agents are involved`
- `boring_task:`
  `all tasks that are performed more than 20 times per hour`
- `comlex_task:`
  `all tasks that have more than 3 levels of sub-tasks`

In DLP syntax a boring task could be defined:

```
   boring_tasks(T) :-
      gta_task <- is_instance(T),
      T <- frequency(Freq, Unit),
      Freq > 20,
      Unit = hour.
```

Cooperation can be seen as a dependency or interaction between certain tasks performed by different agents. Using that definition it is possible to define an expression that shows the frequency of interaction or how tight the cooperation is, for instance by counting the number of agents involved in a number of tasks. Because GTA looks at an organization of people and tasks instead of looking at one person, tasks are explicitly related to agents, objects and roles. All these relationships are established when the data is entered in EUTERPE.

Verification

This kind of analysis is on a more logical level. Verification concerns only the model as it has been specified. Only a limited degree of verification of a task model can be supported due to the inherently lack of formal foundations for task models. There is not a model to verify the task models with. However it is possible to see if the task model satisfies certain domain independent constraints. The task world ontology merely defines the concepts and relationships without any constraints. This was done deliberately to give the analyst as much freedom as possible to specify what they find during data gathering. There are however constraints that we would like to have satisfied independently of the specify domain that is being studied. For example we would like that for each task there is at least one responsible role and that each task is really being performed by an agent. These constraints can be specified as logical predicates and can be checked automatically. Examples are:

- `unauthorized task performance:`
  `a task that is performed by an agent who's role does not encompass the responsibility for the task.`
- `unperformed tasks:`
  `a task where no performing agent has been specified.`
- `unhandled event:`
  `an event where no task is being triggered.`
- `occurance of delegation:`
  `a task that is being performed by an agent other than one that is responsible for the task.`
- `impossible task sequence:`
  `a sequence A followed by B followed by C and`
  `the sequence A followed by C followed by B`

In DLP syntax delegation is expressed as:

```
   delegation(Src, Dest) :-
```

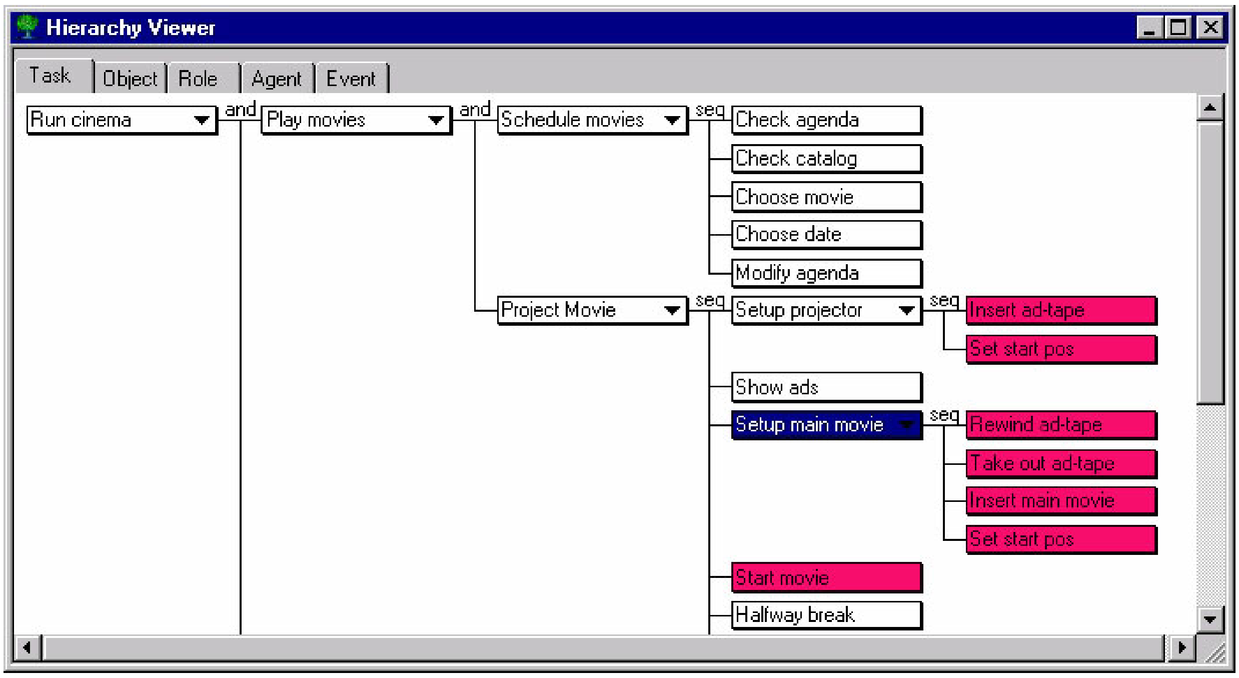

Figure 2 A task tree with colored nodes

```
gta_task <- is_instance(T),
gta_role <- is_instance(Src),
gta_role <- is_instance(Dst),
Src <- responsible(T),
not(Dest <- responsible(T)),
T <- performed_by(Dest).
```

These constraints are similar to the characteristics used in analysis. However, they have been defined irrespectively to the specific study being done. They should hold in any domain. A task model were all constraints are obeyed may be considered "better" than one that does not obey all the constraints. In other words the constraints allow us to denote classes of model which may have an order of preference.

Validation

Validation of task models means checking if the task models corresponds with the task world is describes. In the process of validation one may find that certain tasks are missing or there are more conditions that are involved in executing a task. Often one finds that there are exceptions that had not been found in earlier knowledge elicitation. Consequently validation needs to be done in cooperation with persons from the task world and is not directly automatable by any tool. However it is possible to assist in the validation process for instance by generating scenarios automatically that can be used to confront the person from the task world. Such generated scenarios are in fact simulations of pieces of the task model. Generating simulations has not been implemented in Euterpe but recent work on early task model simulations (Bomsdorf et al., 1997] has shown promising examples of early simulations based on task models.

MANAGING CONFLICTS

When doing an analysis irrespectively of the kind of analysis, problems or conflicts may arise. By conflicts we mean situations that need to be handled by the analyst. Examples of conflicts are contradictory data gathered from different persons describing the same task. In this case one of the persons may have made a mistake or forgot something. Another possibility is that different persons really do the task in a different way in which case it is very interesting to note this fact. Conflicts do not always need to be "solved" but they certainly require attention and should be seen as a hint for possible interesting aspect of the task world. The ontology we use allows inconsistencies and others causes of conflicts to be specified because we found in practice that it is often very important to be aware of these conflicts. Many analysts or designers develop models of a task world with the goal of finding *one* model that captures how the task world works. When modeling complex cooperative environments this is almost never the case and the analysts should not have this one model as the most important goal.

CONCLUSIONS

EUTERPE can be a useful tool for analyzing cooperative environments. Because it is based on the theory of Groupware Task Analysis it has a well founded theoretical background and it offers a structured approach to task analysis. Furthermore, it offers primitives to analyze the captured data in several ways. Analysis primitives are specified logically and can be modified or added. EUTERPE allows cooperation analysis and enables experimentation with analysis primitives, which is also part of our future research.

REFERENCES

d'Ausburg B. (1998) Using Model Checking for the Automatic Validation of User Interface Systems", DSV-IS98, 3-5 June, Abingdon, UK.

Bomsdorf B., Swillus G. (1996) Early Prototyping based on executable task models. In: CHI '96 Conference Companion, Short Paper, Vancouver, Canada, April

Chung-Man Tam R, Maulsby D. and Puerta A. (1998) U-TEL: A Tool for Eliciting User Task Models from Domain Experts, *Proceedings of IUI 98*, pp. 77-80, San Francisco, USA

Eliëns, A. (1992) DLP - A Language for Distributed Logic Programming, *John Wiley & Sons*, Chichester

Kaindl H. (1993) The Missing Link in Requirements Engineering, *ACM SIGSOFT Software Engineering Notes*, vol 18 no 2. April pp. 3039

Loucopoulos P., Karakostas V. (1995) System Requirements Engineering, *McGraw-Hill*, London

Noldus, L.P.J.J. (1991) The Observer: a software system for collection and analysis of observational data, *Behavior Research Methods Instruments & Computers*, 23, pp. 415-429. 1991.

Puerta A. (1997) A Model Based Interface Development Environment, *IEEE Software*, July/August 1997, pp. 40-47

Veer, G.C. van der, Lenting B.F. and Bergevoet B.A.J. (1996) GTA:Groupware Task Analysis - Modeling Complexity, *Acta Psychologica 91*, pp. 297-322

Veer, G.C. van der, Hoeve M. and Lenting B.F. (1996) Modeling complex work systems - method meets reality, In: T.R.G. Green, J.J. Canas and C.P. Warren (eds) Cognition and the worksystem, 8th European Conference on Cognitive Ergonomics (EACE) Inria, Le Chesnay cedex, pp. 115-120

Welie M. van, Veer G.C. van der, Eliëns A. (1998) An Ontology for Task World Models, *Proceedings of DSV-IS98*, 3-5 June, Abingdon, UK