Integrated Representations for Task Modeling

Martijn van Welie, Gerrit van der Veer, Adriaan Koster

Division of Mathematics and Computer Science, FEW
Vrije UniversityAmsterdam
1081 HV Amsterdam
{martijn,gerrit}@cs.vu.nl
http://www.cs.vu.nl/~martijn/

ABSTRACT
During a task analysis much data is generated. Interpretation of the data and the development of knowledge are crucial to a successful task analysis. Task models are used to document and communicate the knowledge gained during the task analysis process. In order to describe all relevant aspects of the context of use, several representations are needed. This paper examines the requirements for such representations and proposes a set of complementary representations.

Keywords
Task Modeling, Task Analysis, Representations, Diagrams, UML, Contextual Design

INTRODUCTION
Knowledge elicitation and ethnographic workplace studies usually generate a lot of data and this data needs to be captured and well understood. If this data is not captured, the knowledge cannot be communicated to other members of the design team and consequently gets lost. Task modeling is the activity of transforming raw task and user related data or envisioning ideas into structured pieces of task knowledge. This knowledge is usually documented in a specification that uses several different representations. Each representation is intended to emphasize a certain aspect of this knowledge. Considering the complexity of the task world and the various possible views, it is clear that several different representations are needed. Ideally, the analysts have a collection of representations at hand that covers all aspects and views. At the same time, it is preferable that such a collection is kept small so that the designer does not drown in a plethora of overlapping representations. Representations must be useful and usable for designers. This paper discusses requirements for task modeling representations and defines a collection of representations that together cover most aspects of the knowledge that need to be documented, while minimizing the overlap.

CHOOSING REPRESENTATIONS
Task modeling for large cooperative systems can not be adequately handled by the classic task modeling techniques. Representations such as task trees are not well suited for describing the complex dynamics typically found in complex environments such as groupware systems. Additional representations are needed that can deal with aspects such as communication, coordination, social structures and work flow. However, many representations already exist for task modeling as well as other related modeling activities. Not all of them are useful in practice and the question is what makes a representation useful and usable. One aspect of a representation is that it should be *effective*. In (Macinlay 1986) Macinlay defines the effectiveness of a visual language as "*whether a language exploits the capabilities of the output medium and the human visual system*". This notion can be expanded to include *purpose* and *audience* i.e. what is the representation intended for and who is going to use it, because "*visualizations are not useful without insight about their use, about their significance and limitations*" (Petre, Blackwell, and Green 1997). Developing usable diagram techniques is difficult and requires insight in all of these aspects. In fact, one could say that usability is just as important for graphical representations as it is for user interfaces, both depending strongly on the context of use. Most of the research in this area is the field of *information visualization* (Card, Macinlay, and Shneiderman 1999) or *visual design* (Tufte 1990,Tufte 1983).

If we wish to choose representations, we must first distinguish several purposes for which they can be used and by whom (Britton and Jones 1999). Within task analysis the purposes of representations typically include:

1. To document and to communicate knowledge between designers.
2. To analyze work and to find bottlenecks and opportunities.
3. To structure thinking for individual designers.
4. To discuss aspects of the task world within the design team.
5. To propose changes or additions within the design team.
6. To compare alternatives in the design team or with a client.

Additionally we need some aspects that help discussing and comparing representations. For this discussion we will take the position that a representation essentially is *a mapping of concepts and relationships (and possibly attributes) to the visual domain*. Some aspects may concern the concepts and relationships while others concern the appropriateness of the mapping in relation to the

purpose and audience. The following aspects are important to consider for representations:

Intended Purpose. For what purpose is the representation intended? Certain representations work well for communicating with clients while others only help structure a single designer's thought. Similarly, certain representations focus on time while others focus on structure. Effectiveness is reduced when the representation does not support the purpose.
Coverage. What concepts and relationships are involved? What information is shown and what is not? Is the information suitable for task analysis purposes? The information covered by a representation determines the view it supports.
Complexity. What is the complexity of the representations in terms of the number of concepts and relationships that are shown? If the complexity is high the understandability is usually low, which is probably not desirable.
Understandability. How well can the representation be understood? Understandability concerns how successful the concepts and relationships have been mapped to a graphical representation. Representations should be easy to understand for the intended audience/users. If not, they will not be used. Stakeholders may come from different disciplines which makes a common understanding more difficult to reach. Other aspects such as visibility also play a large role i.e. how easy parts of the representations can be distinguished or what kind of first impression a representation gives.
Intended Audience. Who is going to use the representation? Certain representations are more familiar to designers from different disciplines than others. Also clients and other stakeholders may be familiar with certain representations. For example, UML is familiar to most Software Engineers while unfamiliar to ethnographers.

MULTIPLE VIEWPOINT

Since multiple representations are needed, it first needs to be clear which viewpoints need to be covered. In our task analysis method GTA (van der Veer, Lenting, and Bergevoet 1996), several views are defined. Constructing a set of representations means selecting a set that covers most aspects of the three views in GTA. When the GTA viewpoints are translated we can distinguish several requirements for the whole set:

- Should describe the *structure* of the work including the tasks, goals and roles.
- Should describe the *dynamics* of the work in time.
- Should describe the use of *objects, tools* and other *artifacts*.
- Should describe the *physical environment* of the work.

This does not mean however, that for each aspect a *single* representation is needed or sufficient. Moreover, representations are preferably easy to understand by engineers, ergonomists, psychologists, requiring a trade-off between representational power and comprehensiveness. These representations are targeted for the analysts and designers and not for communicating with the client which requires other representations such as scenarios and use cases. In (Killich, Luczak, Schlick, Weissenbach, Wiedenmaier, and Ziegler 1999) another set of requirements for task models is given. On the basis of their requirements, they conclude that UML (Rumbaugh, Jacobson, and Booch 1997) contains the most complete set of representations that meet the requirements. However, they do not provide a complete set that meets all requirements. Another well known set of representations is given by means of Contextual Design's work models (Beyer and Holtzblatt 1998).

Representing Work Structure

Work structure is concerned with the structure of the work and how it is allocated to people. Work structure is usually represented using task trees that show a hierarchical decomposition of the work. Often some timing information is added using constructors such as SEQ, LOOP, PAR and OR. The constructors cannot always be used especially when the task sequence uses a combination of sequential and optional tasks (van Welie, van der Veer, and Eliëns 1998). Details of the task can effectively be described using templates. Details include the state changes, frequency and duration, triggering and start/stop conditions. Tasks also need to be explicitly related to the goals and the roles that perform the tasks. When the task structure is viewed in relation to roles the UML Collaboration diagram is useful, because task trees cannot cope well with different roles and each role consequently gets its own tree.

Representing Work Dynamics

Especially when multiple roles are involved in a certain task, timing and changes in control are essential. A task tree cannot represent this well. Workflow or activity models are needed to capture this aspect. Work dynamics involve the sequence in which tasks are performed in relation to the roles that perform them. Additionally, parallel and optional tasks should be modeled, especially when the sequence depends on decisions. Cooperation and communication can partially be described using activity models. Usually scenarios (or use cases) are described in such workflow diagrams. A scenario is triggered by some event and starts with some goal being activated. The scenario usually ends when the goal is achieved but other goals may have been activated in the course of tasks and may not be reached yet. In case studies such as (van Loo, van der Veer, and van Welie 1999) it turned out that this event driven dynamic aspect of cooperative work can be very important.

Representing Tools and Artifacts

The work environment itself usually contains many objects (a hundred or more is not unusual) some of

which are used directly in tasks and other that may be "just lying around". The objects can be tools that people use either in software or in hardware but other objects may be directly manipulated in tasks. For some of these objects it may be relevant to describe them in detail. Details may include their structure, their type, and object specific attributes.

Representing the Work Environment

Often it is important to know what the physical layout of the work environment is. Pictures, drawings, maps and video clips can capture parts of this information. Usually maps or drawings are annotated with comments relating to their impact on the work such as reachability of objects. Most objects that appear in such representations are also represented when modeling *Tools* and *Artifacts*. Additionally, the social structure or the culture of the work environment needs to be represented. People rarely perform their work in solitude and social relationships influence work. Typically, roles influence other roles with a certain strength and there are certainly attitudes to be found between them.

INTEGRATING REPRESENTATIONS

The aspects of the different viewpoints and representations need to be semantically integrated. In any set of representations, it holds that each representation is a view of the same task world model where the same elements may appear in different representations. Our task world ontology (van Welie, van der Veer, and Eliëns 1998), see Figure 1, captures most of the fundamental concepts and relationship that form the basis of all the representations. However, some representations such as videos are not very clearly structured and hence, are not covered by the ontology. For other representations such as the physical layout, this is theoretically possible but in practice this is not useful to do. In the next sections we will use the concepts and relationships to define what is shown in the representations.

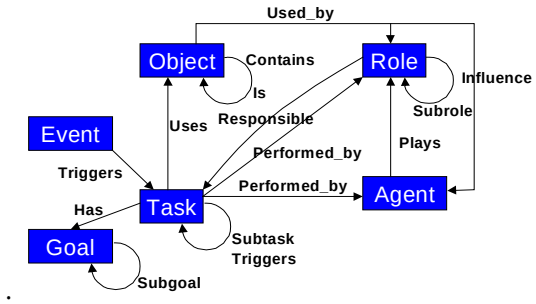

Figure 1 The Task World Ontology

A COLLECTION OF ONTOLOGY BASED REPRESENTATIONS

It is clear that some are more useful/usable than others and that improvements can be made. In this section, we will define a collection of representations that cover the views as defined in the previous sections. This collection of coherent representations is an attempt to provide a more useful collection of representations for practitioners. For each of the views we will define one or more representations that form a useful "package" for that view. Together, the representations can form a practical tool set for the designer. The representations are based on existing representations but include some additions or modifications to make them more usable and useful for task modeling.

Constructing a set of Representations

The collection of representations that is discussed in the next sections combines several existing representations. Additionally, some modifications have been made. Compared to Contextual Design's (Beyer and Holtzblatt 1998) work models (CWM), the main differences are:

- The CWM sequence model is replaced by a work flow model similar to the UML Activity diagram.
- The CWM sequence and CWM flow model are combined into one representation.
- Decomposition trees are added.
- The CWM cultural model has been redesigned.
- The number of concepts is larger than in CWM.

Compared to UML (Rumbaugh, Jacobson, and Booch 1997), we use a modification of the Activity Model. We have added and event and goals lane as well as changed representations for parallelism and choice.

Modeling the Work Structure

The purpose of the work structure model is to represent how people divide their work into smaller meaningful pieces in order to achieve certain goals. Knowing the structure of work allows the designers to understand how people think about their work, to see where problems arise and how tasks are related to the user's goals. The relation between tasks and goals helps the designers to choose which tasks need to be supported by the system and why i.e. which user goals are independent of the technology used.

For modeling work structure the task decomposition tree has proven to be useful and usable in practice. The tree is essentially based on the *subtask* relationship between tasks. Besides tasks, goals can also be incorporated. At the highest level a tree can start with a goal and subgoals and then proceed with tasks and subtasks. In that case the *subgoal* and *has* relationship are also used. A task decomposition is modeled from the viewpoint of one role or goal. If complex systems are modeled, several task trees are needed to describe the work for all the roles. It then becomes difficult to see how work is interleaved. Trees normally contain a time ordering using constructors from top to bottom or left to right, depending on the way the tree is drawn. The inclusion of time information can be insightful but it is often also problematic. ConcurTaskTrees (Paterno, Mancini, and Meniconi 1997) use LOTOS

operators, which are probably the best-defined time operators. On the other hand, it is not always necessary to be very precise in everything that is modeled. Designers will typically model that certain tasks occur *sometimes* or *almost never*. In our opinion, including some time information is useful but this kind of information is better represented in a work flow model if precision is required.

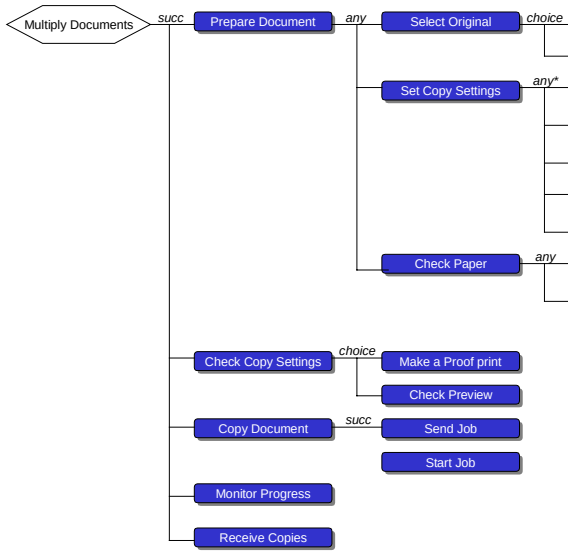

Figure 2 **Representing work structure**

If time is included, then a number of time operators are plausible. In our experience, it is useful to have a set of standard operators while also allowing designers to create their own operators when needed. For the average usage, the following time relationships have proven sufficient:

- **Concurrent**. The tasks occur concurrently.
- **Choice**. One out of a set of tasks is done.
- **AnyOrder**. All tasks of a set of tasks are done in no fixed order.
- **Successive**. One task is followed by another.
- **Any**. Zero or more tasks of a set of tasks are done in no fixed order.
- ***** combined with other constructor. Used to express iteration.

In the work structure model, the root of the tree is a goal with possibly some subgoals. Connected to goals are tasks which are represented as rounded rectangles. The tree is drawn from left to right instead of top-to-bottom for more economical use of space, especially when trees become large, see Figure 2. Other aspects of work structure include role structures and the relationships with tasks. For role structures trees can also be used. When used to show goal or role hierarchies the time constructors are not used.

Modeling the Work Dynamics

The purpose of the work dynamics model is to show work in relation to time and roles. The model gives the designer insight in the order in which tasks are performed and how different people are involved in them. Additionally, it can show how people work together and communicate by exchanging objects or messages. Typically, such a flow model describes a small scenario involving one or more roles. This way, it shows how work is interleaved.

The flow model specified here is a variation on the UML Activity graph. We included events and goals to make it more suitable for task analysis. Additionally, the representations of the time operators have been modified to be more appealing. This way the collaboration diagram (or Contextual Design's Flow Model) is not needed anymore since the information has been combined in one representation. Each flow model describes a scenario that is triggered by an event. Work usually does not start by itself but instead is often highly event driven (van der Veer, van Welie, and Thorborg 1999). The event is represented by an oval which is connected to the first task. The sequence of tasks is given using a **Concurrent** operator or a **Choice** operator and not any of the other operators as suggested for the structure model. Tasks can optionally be arranged in *swim lanes*, one for each role. Objects can be passed between tasks that have different roles and are drawn on the border of the adjacent swim lanes. When needed, goals can also be added to this representation. With a certain task a new goal can get "activated" until it is "reached" in a later task. The goals are written in the first column with vertical lines to show how long they are activated.

The flow model does not show hierarchical relationships between tasks and a flow model can only use tasks that are hierarchically on the same level. For subtasks, a new flow model needs to be specified. The addition of the goal lane can show many useful aspects when analyzing the work flow. For example, Figure 3 shows that once the "teacher" has received the book his goal is achieved but the scenario is not finished yet.

In terms of the ontology, the flow model is based on the concepts Event, Task, Object and Role. The relationships used are *triggers*, *responsible*, and *uses*. The operators are **Concurrent**, **Choice**, and **Successive** which are parameters in the *triggers* relationship. The **AnyOrder** and **Any** constructors are not valid in this representation and the **Successive** operator is implicit in the direction of the arrows. For objects that are being passed between roles it holds that each object must be associated to both tasks with the *uses* relationship. Note that the objects that are used in one task are not shown in the representations.

Iteration is not specified in the flow diagram. If a task is done several times, an asterisk can be used to indicate that the task and it subtasks are done several times. However, usually iterations are specified in the Work structure model. Iteration is specified on

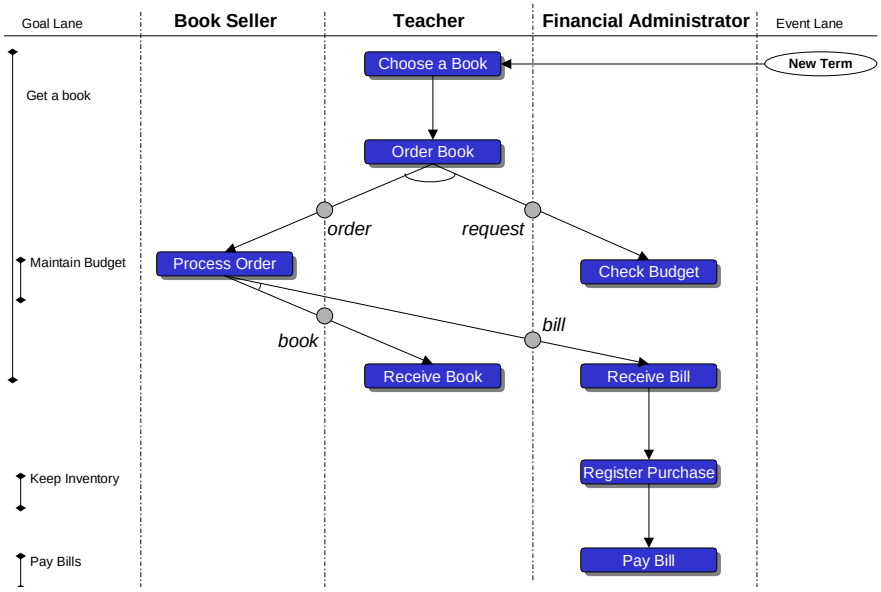

Figure 3 Representing work flow

subtasks and not tasks on the same level, which are shown in the Work Flow model.

Modeling the Work Artifacts

The artifact model shows two relationships between objects: the *containment* and *type* relationship. For both a tree diagram is used. The objects themselves can be annotated with their attributes or their visual appearance. In order to express containment and type, the UML class diagram notation can be used. However, it is important to remember that we are only modeling objects that are relevant to the user and not any irrelevant internal system objects. To some this may suggest that the task model describes an object oriented system model, which is *not* the case.

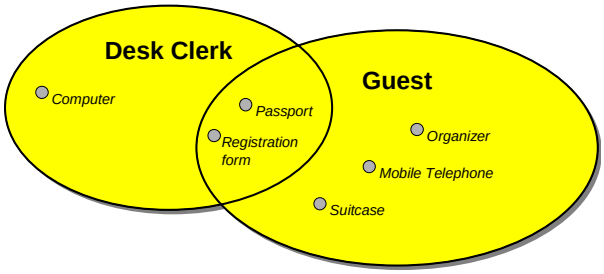

Figure 4 Artifacts and roles

The use of objects in tasks is partly covered by the Work Flow model. The Work Artifacts model the structural aspects of the objects. However, objects may also be connected to their users i.e. roles or agents, instead of to the tasks where they are used. In such a diagram, the users are represented as ovals and the objects are labeled dots within the ovals. The ovals may overlap if more than one user uses the object, see Figure 4.

Modeling the Work Environment

The environment model describes two aspects of the environment. Firstly the physical layout of the environment and secondly the culture within the environment. The **physical model** is simply described by one or more annotated "maps" of the environment. The purpose is to show where objects are located in relation to each other. The objects are those that are relevant for the work and also those that are in the same space. Such layout diagrams can easily be drawn using commercial drawing software such as Visio.

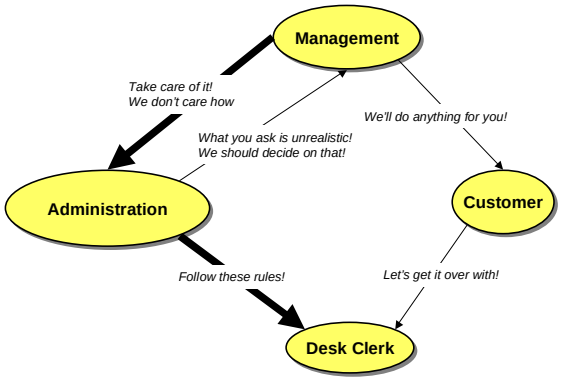

Figure 5 **Representing Culture**

The other model is the **Culture Model**. The culture model we describe here is an adaptation of the culture model from Contextual Design. In Contextual Design the roles are represented in overlapping circles. However, overlapping of circles does not have any meaning although it suggests that there is one. Hence we adapted the model, see Figure 5. We define the culture model as follows.

- Roles are represented as ovals.
- The ovals are connected by arrows if there is a *force* between roles. The relative strength of the

force is depicted in the width of the arrow and forces can be bi-directional.

- Forces are annotated with *attitudes* of the force relationship.

In some cases, a force applies to more than one role. By drawing an extra circle around roles, a force can indicate one-to-many forces which can typically be used to describe "corporate culture".

STATIC VS. DYNAMIC REPRESENTATIONS

All of the representations discussed in the previous sections are static. However, representations can also be more dynamic. Traditionally, a representation is static i.e. it does not change after it is drawn and is designed for use on paper. However, it is often convenient to emphasize a certain aspect in the representation. When software is used to draw the representations, the representations can be changed dynamically. In (Card, Macinlay, and Shneiderman 1999) they are called *active diagrams*. For example, one could easily switch between a flow model with or without swim lanes. Alternatively, it could be possible to add some extra information by *marking* tasks as "problematic" or "uses object X", see (van Welie, van der Veer, and Eliëns 1998). Such annotations are often done by designers to explain certain aspects to others during a presentation or in documentation. In software we are already very much used to active diagrams and they occur in scrolling, zooming and syntax highlighting. This asks for a more flexible view on what constitutes a representation and when a representation can be modified. The dynamic aspects could be controlled manually by the viewer but could also be pre-specified using a function in which case we usually speak of animation. Now that it becomes increasingly easier to create dynamic representations it is important to understand when and how they could be applied usefully in design.

In task modeling, animation is a way to create more dynamic representations. Animation can be used in simulations of scenarios or task models (Biere, Bomsdorf, and Szwillus). Using simulations an analyst can step through a scenario and get a different feel for what goes on. Other purposes might be to "debug" a task model which is particularly useful for envisioned task models.

CONCLUSIONS

In this paper, the need for a set of integrated representations has been discussed. Such a set has been defined and shown with examples. Most of the representations can be integrated using a task world ontology. The task world ontology serves as the underlying model for the representations. Together, the representations cover several important viewpoints for task modeling.

REFERENCES

Beyer, H. and Holtzblatt, K. (1998), *Contextual Design*, Morgan Kaufmann Publishers,

Biere, M., Bomsdorf, B., and Szwillus, G. The Visual Task Model Builder, *Proceedings of CADUI'99*, Louvain-la-Neuve, Belgium.

Britton, C. and Jones, S. (1999) The Untrained Eye: How Language for Software Specification Support Understanding in Untrained Users, Human-Computer Interaction 14, 191-244.

Card, S., Macinlay, J. and Shneiderman, B. (1999), *Readings in Information Visualization: Using Vision to Think*, Morgan Kaufmann Publishers,

Killich, S., Luczak, H., Schlick, C., Weissenbach, M., Wiedenmaier, S. and Ziegler, J. (1999) Task Modelling for cooperative work, Behaviour & Information Technology 18, 325-338.

Macinlay, J. (1986) Automating the Design of Graphical Presentations of Relational Information, ACM Transactions on Graphics 5, 110-141.

Paterno, F. D., Mancini, C., and Meniconi, S. (1997), ConcurTaskTrees: A Diagrammatic Notation for Specifying Task Models, *Proceedings of Interact '97*, Sydney, Chapman & Hall.

Petre, M., Blackwell, A. and Green, T. (1997), *Cognitive Questions in Software Visualisation*, in: Stasko, J., Domingue, J., Price, B., and Brown, M., Software Visualization: Programming As a Multi-Media Experience , MIT Press.

Rumbaugh, J., Jacobson, I. and Booch, G. (1997), *Unified Modeling Language Reference Manual*, Addison Wesley,

Tufte, E. (1983), *The Visual Display of Quantative Information*, Graphics Press, Connecticut

Tufte, E. (1990), *Envisioning Information*, Graphics Press, Connecticut

van der Veer, G.C., Lenting, B.F. and Bergevoet, B.A.J. (1996), GTA: Groupware Task Analysis - Modeling Complexity, *Acta Psychologica*, no. 91, pp.297-322.

van der Veer, G.C., van Welie, M. and Thorborg, D. (1999) Modeling Complex Processes in GTA, Ergonomics 42, 1572-1587.

van Loo, Reinard, van der Veer, G. C., and van Welie, M. (1999), Groupware Task Analysis in Practice: a scientific approach meets security problems, *7th European Conference on Cognitive Science Approaches to Process Control* , Villeneuve d'Ascq, France .

van Welie, M., van der Veer, G. C., and Eliëns, A. (1998), Euterpe - Tool support for analyzing cooperative environments, *Ninth European Conference on Cognitive Ergonomics*, Limerick, Ireland.

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