Support for Task Modeling – A "Constructive" Exploration

Anke Dittmar[1], Peter Forbrig[1], Simone Heftberger[2], Chris Stary[2]

[1]University of Rostock, Department of Computer Science
A.-Einstein-Str.21, D-18051 Rostock
[ad/pforbrig]@informatik.uni-rostock.de

[2]University of Linz, Department of Business Information Systems – Communications Engineering, Freistädterstr. 315, A-4040 Linz
[Simone.Heftberger /Christian.Stary]@jku.at

Abstract. Although model-based approaches focusing on task modeling for user-interface design are well accepted among researchers, they are rarely used by industrial developers. Besides a lack of theoretical frameworks for task modeling insufficient tool support might be the reason for the low diffusion of this approach to interactive software- development processes. Thus, we explored the leading-edge tools TaOSpec, ProcessLens, and CTTE with respect to the formal representation of work tasks, and the creation of task scenarios. The results reveal that current model-based design approaches should be more conceivable by their users with respect to work tasks and their organization. This objective can be met by embedding scenario-based design elements into current tools, thus, increasing integrative tool and organizational development support.

1. Introduction

With the emergence of interactive software systems and their widespread use over the last decades, the needs of potential users have increasingly become crucial to design. Design techniques, such as model-based approaches (cf. [1], [2]) encourage designers to embed user tasks into design representations to achieve accurate interactive functionality of software systems. Such representation might be based on common representation schemes, such as XIML [3]. Other approaches, such as participatory design (cf. [4]) and scenario-based design (cf. [5]) emphasize the active participation of users during the design process to achieve user-centered systems.

Although traditional model-based design techniques do not require user participation, they reflect user perspectives on work tasks and work processes. The designers create different models and their relations to describe tasks, task domains, user characteristics etc. A variety of representations has to be used in the course of design, in order to involve all stakeholders, to discuss their interests and to capture contextual knowledge [6].

Considering sustainable diffusion of model-based approaches to industrial software design, the latest developments (cf. [7]) do not indicate significant progress, although several task-based tools has been tested successfully from the functional perspective (cf. [8]). The user side, in particular designers and involved users accomplishing work tasks, has not been investigated thoroughly. Given the fact that scenarios of use can be created interactively from formal task specifications, scenario-based design elements might help to make task models more conceivable by users and trigger organizational developments (cf. [9]).

Consequently, we review major task-based design tools with respect to their accurate representation and capability to help users understand task-specific support capabilities based on design specifications and/or on their execution. Tools as a kind of representation of modeling concepts are intended to support task modeling activities of designers. Our review should also help developers to get more insight in applying certain representation schemes and some underlying ideas. They might recognize gaps between what they want to express and what they can describe applying a certain modeling approach. In this way, they experience a similar situation as users given a certain work task, namely when they are co-constructing an interactive application with designers.

For the sake of a structured review we first introduce a use case in Section 2. We use that case to demonstrate the capabilities of the considered task-modeling tools. For its description we use generally accepted constructs within the task modeling community. In Section 3 we briefly introduce the considered tools including their conceptual background. For each of the 3 state-of-the-art tools (TaOSpec, Process-Lens, CTTE) we demonstrate how the example introduced in Section 2 can be described formally and processed. The generated task scenarios and their interactivity are discussed in Section 4. Sophisticated model-based approaches enable to create interactive task scenarios as hands-on experience for users, and thus, trigger reflective organizational developments. Given the inputs from Section 3 and 4 we finally provide a comparative analysis of the considered approaches with respect to their capabilities in Section 5. Although differences between existing task-modeling approaches can already be identified at the conceptual level (see Section 2) besides the tool level, those differences might be required for dedicated design purposes, such as to provide hands-on experience of envisioned task scenarios, and the scope of applying representations, such as to specify workplace improvements. In Section 6 we conclude the paper stressing those benefits and proposing further constructive explorations.

2. A Sample Interactive Task

Our running example is taken from [10] (Figure 1) and specified in TaskMODL, the Task MODeling Language. In order to accomplish the task *Read email* a user has to perform the sub-tasks *Get new email* and *Manage email* in arbitrary order. Emails are managed by executing the sub-task *Manage message* iteratively. Each cycle requires a message to be read (sub-task *Read message*) and being transferred (sub-task *Transfer message*).

The concepts *Mailbox* and *Message* represent the task domain. They are specified in RML, a domain modeling language based on language and set theory. It is assumed that a Mailbox contains messages. *In* and *Out* are specific instances of *Mailbox*. Elements of the task domain are used as resources for tasks. In the graphical notation they appear within the bottom part of task nodes or at the edges between nodes (cf. [10]).

Although this example addresses a simple work task, it captures all relevant constructs and items for the purpose of studying task modeling and the propagation of those constructs to scenarios of use. We consider task modeling essential along hierarchical and sequential structures. In addition, task modeling has to capture objects of the task domain, and relations between tasks and objects (cf. [11]).

The following scenario-like description of the graphically displayed situation helps to reveal the usefulness of some other concepts for contextual task modeling.

Patty Smith works as an assistant in the small company ExampleOrg. She is responsible for receiving all inquiries and for presorting them before they are transferred to Mr. Young, the manager of ExampleOrg. In case of email messages she skims through the sender, the topic and, if necessary, she also skims through the content of a message in order to decide if it needs to be handled at all.

All members of the staff have internal mailboxes called 'Urgent' and 'Normal'. They are used to send them inquiries which need to be treated urgently or in a regular time. Everybody knows that all urgent mail messages have to be answered and that the mailbox 'Urgent' has to be empty before normal inquiries should be handled.

Mr. Young opens his mailboxes every morning and late in the afternoon to react to inquiries Patty has sent him. He forwards those messages he does not answer by himself to Paul or Peter . . .

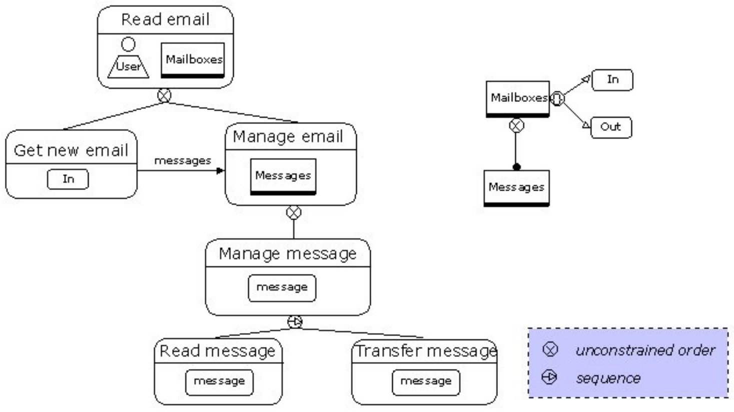


Fig. 1. Task model Read email (left side) and domain model (right side), from [10].

The scenario description enriches the graphically displayed content, since it considers human actors, their relations to tasks and their collaboration more deliberately. Hence, we will check whether and how the considered tools are able to capture those constructs applied in the example. In particular, we will analyze the use of

- tasks,

- task domains,
- actors,
- relations between tasks and actors, and
- relations between tasks and domain objects

and corresponding design support.

3. Tool Support for Creating Task Models

3.1 TaOSpec – An Executable Specification Language for Tasks and Objects

TaOSpec is a specification language that has been developed for higher-order task modelling (cf. [12]). In contrast to other task-modeling approaches in TaOSpec (sub-) models of goals, tasks, actions and (task) objects (created, deleted, used, or manipulated by actions) are structured along identical modeling principles. Models are described through cognitive elements (objects) and their mutual relations. A dedicated relation is the instance-pattern relation between objects explaining abstractions.

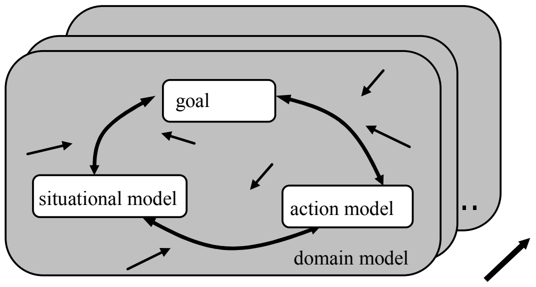


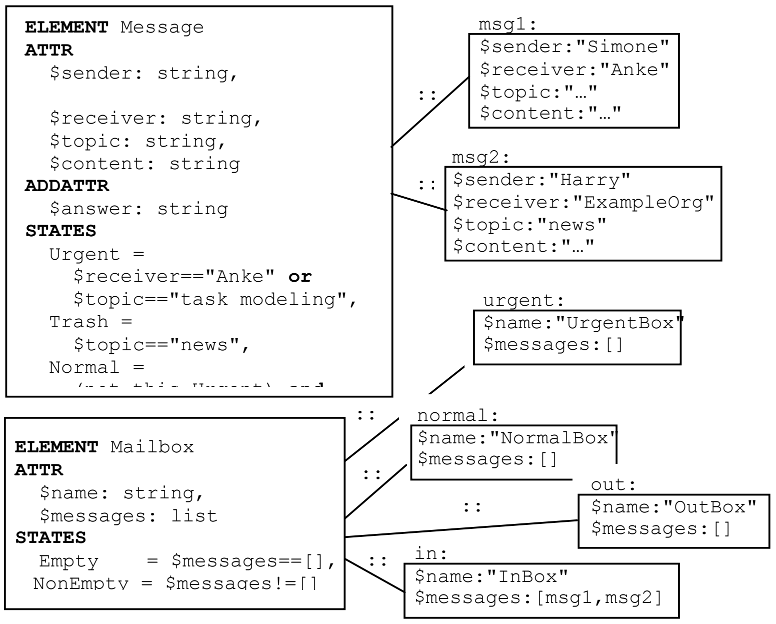
Fig. 2. Tasks as meta-actions modifying sub-models about situations, goals, and actions.

Action, *goal*, and *situational models* are considered as sub-structures within the universal model (*domain model*) human beings possess about their environment (observed world). Such sub-structures can be organized more efficiently with respect to their purpose. For example, an action is assumed to have a simple hierarchical and sequential character as already shown in the example of Figure 1, whereas goals are organized as networks, since there might be contradictions between sub-goals. Tasks are considered as meta-actions (that is to say processes) including the manipulation of sub-models capturing actual situations, goals, and actions to achieve these goals (see Figure 2).

Higher-order task models give a more comprehensive understanding of what tasks represent for humans. Since in TaOSpec there is no strict borderline between procedural and state descriptions, it is possible, for instance, to treat actions as objects of other action environments, to manipulate them with respect to certain goals, and to incorporate them as parts of other actions.

Basically, objects of the domain model are the result of basic operations on sets and sequences of symbols. However, TaOSpec supports a more elaborated structure of objects which is more convenient in order to describe pattern objects. Objects are characterized by a (finite) set of attributes (name-value pairs). We distinguish between basic and additional attributes. An object O_I is considered an instance of object O_P (called pattern object), if, at least, all names of the basic attributes of O_P also occur as attribute names in O_I, and their corresponding values are instances of the attribute values in O_P.

Furthermore, TaOSpec facilitates the description of subsets of instances of a pattern object by partial equations. On the left hand side of such an equation, the designer specifies the identifier of the subset of interest. The right hand side consists of an expression whose operands can be identifiers of other defined subsets, restrictions of attribute values and introduced additional attributes. TaOSpec offers a set of predefined state and temporal operators on these operands (like the operator or in Figure 3). For a more detailed description of TaOSpec see, e.g. [13], [14].



```
ELEMENT Message
ATTR
  $sender: string,

  $receiver: string,
  $topic: string,
  $content: string
ADDATTR
  $answer: string
STATES
  Urgent =
    $receiver=="Anke" or
    $topic=="task modeling",
  Trash =
    $topic=="news",
  Normal =
```

```
ELEMENT Mailbox
ATTR
  $name: string,
  $messages: list
STATES
  Empty    = $messages==[],
 NonEmpty = $messages!=[]
```

```
msg1:
$sender:"Simone"
$receiver:"Anke"
$topic:"…"
$content:"…"

msg2:
$sender:"Harry"
$receiver:"ExampleOrg"
$topic:"news"
$content:"…"

urgent:
$name:"UrgentBox"
$messages:[]

normal:
$name:"NormalBox"
$messages:[]

out:
$name:"OutBox"
$messages:[]

in:
$name:"InBox"
$messages:[msg1,msg2]
```

Fig. 3. Pattern objects *Mailbox* and *Message* specified in TaOSpec and some instances.

Figure 3 depicts the way of specifying objects. *Mailbox* and *Message* serve as pattern objects describing concrete task situations, as illustrated by some instances. Mailbox *in* contains 2 messages, all other mailboxes are empty. Message *msg1* is a member of subset *Urgent* according to its attribute *$receiver:"Anke"*. Another, more appropriate interpretation in this context, is that *msg1* is in state *Urgent*.

For describing the hierarchical and sequential character of actions pattern objects with partial equations containing temporal operators are used. Figure 4 shows the skeleton of the action structure for the running example. (For convenience, we chose CTTE-notation for temporal operators.)

```
EQU

HandleEMail(…)=                          ⊗
     GetEMail(…) >> PreSortEMail(…) >>
ManageMail(…),

PreSortEMail(…)=
   SelectInMail(…) >> SkimThroughMail(…)
   >> TransferOrDeleteMail(…),

ManageMail(…)=
   ManageUrgentMail(…) >> ManageNormalMail(…),

ManageUrgentMail(…) =
   SelectUrgentMail(…) >> ReadUrgentMail(…)
   >> (  AnswerUrgentMail(…)
     [] TransferUrgentMail(…) ),

ManageNormalMail(…) =
   SelectNormalMail(…)
   >> ReadNormalMail(…)
   >> (  AnswerNormalMail(…)
      [] DeleteNormalMail(…)
      [] TransferNormalMail(…) )
```

temporal operators:

>>	sequential op.
|||	concurrent op.
[]	alternative op.
*	iteration

Fig. 4. Action skeleton of *HandleEMail* in TaOSpec (for explanation of mark ⊗ see Section 4)

Actions and objects of a task domain are related by assigning pre- and post conditions to actions. Such conditions are specified by sets of objects in certain states and denoted in square brackets. Figure 5 shows part of the declaration of action HandleEMail.

```
OPERATION HandleEMail
USES Mailbox,Message
DECL
HandleEMail ($in:Mailbox,$urgent:Mailbox,$normal:Mailbox,
               $out:Mailbox) [POST $in.Empty],
GetEMail($in:Mailbox),
*PreSortEMail($in:Mailbox,$urgent:Mailbox,$normal:Mailbox),
ManageMail($urgent:Mailbox,$normal:Mailbox,$out:Mailbox)
           [PRE $urgent.NonEmpty or $normal.NonEmpty],   ...
EQU ...
TransferOrDeleteMail($m,$urgent,$normal)[$m.Urgent] = ...,
TransferOrDeleteMail($m,$urgent,$normal)[$m.Trash] = ...,
TransferOrDeleteMail($m,$urgent,$normal)[$m.Normal] = ...,    ...
```

Fig. 5. Some pre- and post conditions assigned to sub-actions of *HandleEMail*.

For example, sub-action *ManageMail* can only be performed, if at least one of the mailboxes *urgent* and *normal* is not empty. It is also possible to assign different preconditions to one sub-action as shown in the specification of the sub-action *TransferOrDeleteMail*. It depends on the actual state of the mail message referred to by *$m ($m.Urgent* denotes the request that *$m* has to be in state *Urgent* etc.) which one of the three equations is selected.

TaOSpec is an executable specification language. It allows a user to animate "concrete" actions and to observe the modifications of "concrete" domain objects caused by the actions (see Section 4). For that reason a set of basic operations is implemented which corresponds to the general structure of objects in TaOSpec. There are operations to create/remove objects, to introduce/delete additional attributes to objects, and to set/get attribute values. TaOSpec supports the use of strings, integer and real numbers, Boolean values, and lists together with some basic functionality, such as string concatenation (‘&’), arithmetic operations (‘+’, …), and the insertion/deletion of elements to/from a list (‘:’, ‘delete’) .

Figure 6 describes the effect of answering a standard message by using basic operations. After object *$m* "is changed to an answer message" it is "sent" to mailbox *$out*.

```
EQU ...
SkimThroughMail($m) = done(),

AnswerNormalMail($m,$out) =
   addAttr($m,"answer","hallo "&$m.$sender)
   >> setAttr($m,"receiver",$m.$sender)
   >> setAttr($m,"sender","ExampleOrg")
   >> setAttr($out,"messages",:($m,$out.$messages))
...
```

Fig. 6. Implementations of sub-actions using predefined operations.

In TaOSpec, we use the keyword OPERATION in specifications instead of ACTION, since delivering executable basic operations makes an action model *operational*. Finally, there is a dedicated basic operation called *done()* which has no effect at all. It can be used to leave sub-actions “unspecified” as shown for *SkimThroughMail* in Figure 6.

3.2 ProcessLens – Framework and Tool

ProcessLens supports the task- und role-sensitive development of interactive software through providing an ontology that captures the essentials of work processes (cf. [15],[16]). It incorporates task and user models into a model-based representation scheme. The unifying specification language BILA (Business Intelligence Language) is based on UML and allows to capture model-specific elements and relationships, as well as the structural and dynamic linking of executable models.

The ProcessLens approach also contains a certain design procedure that is based on the representation scheme as shown in Figure 7. The ProcessLens models relevant for task modeling are the user, task and data model:

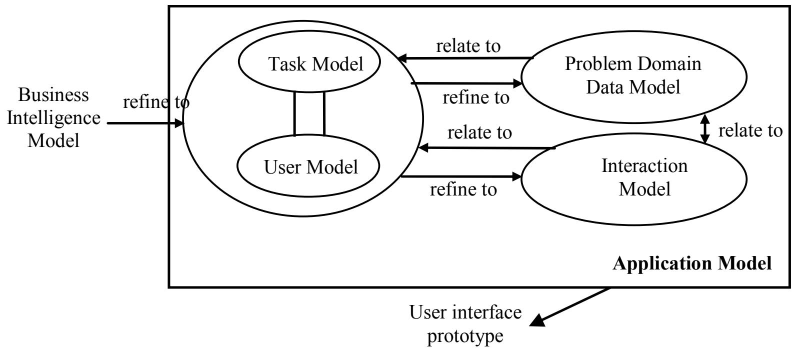


Fig. 7. The ProcessLens Model-Based Framework.

- The *user model* represents a role model by defining specific views on tasks and data (according to the functional roles of users). Typical elements of BILA used in this context are *organizational unit*, *position* and *person*.
- The *task model* comprises the decomposition of user tasks according to the economic and the social organization of work as well as the different activities that users have to perform to accomplish their tasks. Typical elements used for modeling are *task*, *activity* and *tool*.
- The (*problem domain*) *data model* describes the data required for work-task accomplishment. In contrast to traditional data modeling, in ProcessLens both aspects of data are captured: structure and behavior. A particular element of BILA is used extensively in the data model, namely *material*.

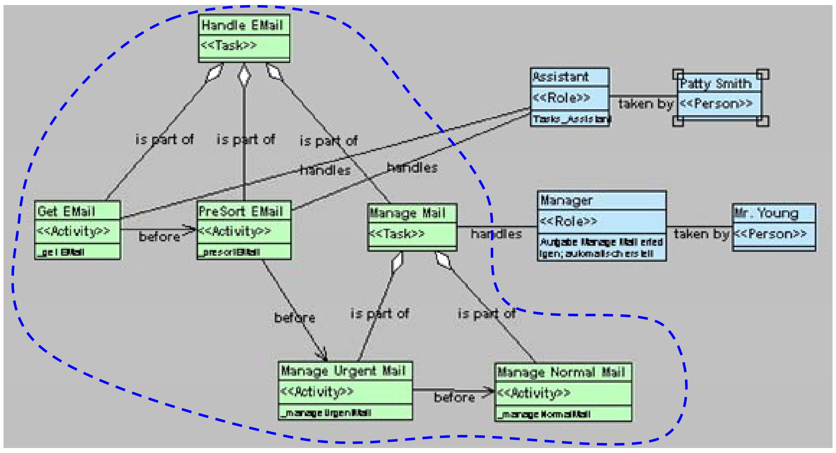


Fig. 8. An integrated structure view on tasks and users of *Handle Email*.

In ProcessLens we use UML class diagrams [17] to specify the structure of all models and their mutual relationships. A set of predefined elements and relations (some of them are mentioned above) supports the modeling activities of work processes. Figure 8 depicts the task model (encircled with a dotted line), the user model and some relations for the running example. Task domain objects (data objects) have to be added and related to the task model.

Designers or users have to specify and assign activity diagrams [18] to dedicated model elements to describe the actual accomplishment of tasks (including the manipulation of data) and role-specific behavior. If elements from different models are related (structure level) their corresponding activity diagrams have to be synchronized using a special kind of ProcessLens transition (synchronization transition at the behavior level). This dynamic linking makes the models operational and is illustrated in Figure 9, in conformance to our example.

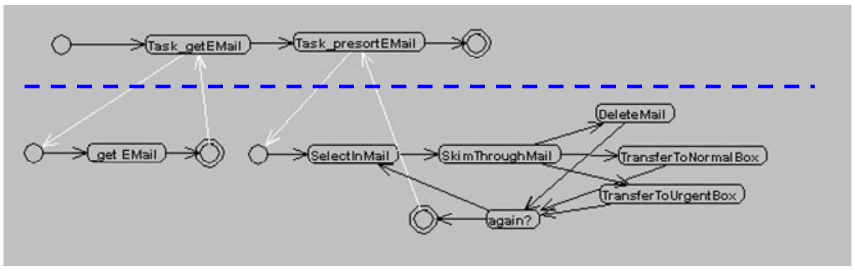


Fig. 9. Activity diagrams of the role element Assistant (above the dotted line) and of the activity elements Get EMail and PreSort EMail of the task model (left and right bottom part), including their synchronization - the white directed links denote synchronization transitions.

In order to animate an application based on its specification, several aspects need to be considered for synchronization. First, action states of activity diagrams of elements of the user model have to be synchronized with (parts of) activity diagrams assigned to elements of the task model – ProcessLens supports role-specific user-interface prototyping. Secondly, action states of activity diagrams of the task model have to be synchronized within the task model as well as with (parts of) activity diagrams of the data model. Although the way of specifying is similar, the semantics for synchronization is different: In the first case (as shown in Figure 9) the division of labor directs the synchronization, whereas in the second case the detailed design of (interactive) functionality is captured.

3.3 CTTE

CTTE is a popular task modeling tool (cf. [19],[20]). Figure 10 illustrates how we applied the tool to model the cooperation between Mr. Young and Patty Smith (see Section 2) in the roles *Manager* and *Assistant*. There are task trees for each role. Some of their nodes are mapped to leaf nodes in the cooperation tree.

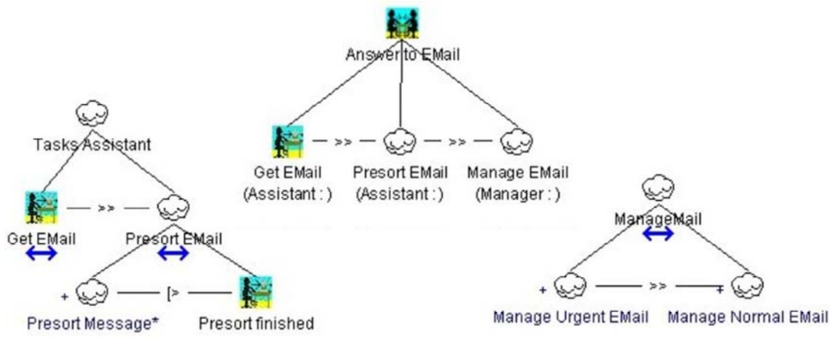
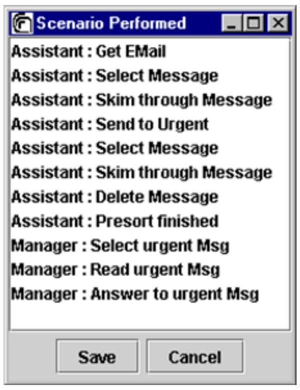


Fig. 10. Cooperation tree of *Handle Email* and parts of the task trees of roles *Assistant* and *Manager* in CTTE-notation.

4. Tool Support for Task Scenarios and Organizational Development

In the following we give two examples of applying task-based approaches in the context of scenario-based developments.



```
Assistant : Get EMail
Assistant : Select Message
Assistant : Skim through Message
Assistant : Send to Urgent
Assistant : Select Message
Assistant : Skim through Message
Assistant : Delete Message
Assistant : Presort finished
Manager : Select urgent Msg
Manager : Read urgent Msg
Manager : Answer to urgent Msg
```

Fig. 11. A task scenario of *Handle Email* in CTTE.

Tool Support 1: Improving the Description of Existing Work Situations

"I'm not sure", said Mr. Young as we animated an execution of task 'Handle EMail' (see CTTE-model in Figure 10 and a snapshot of the animation in Figure 11). "but I think there is something wrong here. Patty works on the incoming mail messages during the whole day. So, if there are some messages in my mailboxes I don't need to wait for her in order to manage the inquiries she has already transferred to me."

Task models are abstract descriptions. Corresponding tools enable users to animate (more or less) concrete task executions interactively. One run of such an animation is referred to a task scenario in the context of this work. As indicated in the introduction, these scenarios can (and should) bridge the gap between model-based and scenario-based ideas. They do not have a narrative character like the scenarios in [5] since they are created on the basis of a formal model. In this way, they are not likely to reflect implicit goals or reveal intrinsic motivation of stakeholders. However, as the above comment of Mr. Young shows task scenarios might encourage involved stakeholders to discuss alternative task scenarios and organizational issues of work when provided with a formal task model.

```
?- animation(situation1).
actual task situation:
(1) Mailbox - {messages:[{sender:"Simone", receiver:"Anke", ...},
                         {sender:"Harry", ..., topic:"news", ...}],
              name="InBox"}
(2) Mailbox - {messages:[], name:"UrgentBox"}
(3) Mailbox - {messages:[], name:"NormalBox"}
(4) Mailbox - {messages:[], name:"OutBox"}
enabled actions:
(1) GetEMail
>: 1
------------------------------
actual task situation: …
enabled actions:
(1) SelectInMail
(2) _PreSortEMail          /* finish cycle PreSortEMail */
>: 1
------------------------------

after performing steps SkimThroughMail and TransferOrDeleteMail[$m.Urgent]
…
------------------------------
actual task situation:
(1) Mailbox - {messages:[{sender:"Harry", ..., topic:"news", ...}],
              name:"InBox"}
(2) Mailbox - {messages:[(5)], name:"UrgentBox"}
(3) Mailbox - {messages:[], name:"NormalBox"}
(4) Mailbox - {messages:[], name:"OutBox"}
(5) Message - {sender="Simone", receiver="Anke", ... / answer:nil}
enabled actions:
(1) SelectInMail
(2) _PreSortEMail
(3)  SelectUrgentMail
(4) _ManageUrgentMail
>: 3
```

Fig. 12. Parts of the task scenario < GetEMail, SelectInMail, SkimThroughMail, TransferOrDeleteMail[$m.Urgent], SelectUrgentMail, …>.

Furthermore, task scenarios reveal the capabilities of the underlying modeling mechanisms. For instance, it is not possible to formalize the description of a task domain in CTTE although required for task modeling.

Figure 12 contains parts of a task scenario as created by interpreting the TaOSpec-model developed in Section 3.1 and modified. TaOSpec not only presents sub-tasks (sub-actions in the context of TaOSpec) to users which are executable through step-by-step animation, but also the state of each task object of the actual task situation. In addition, users can choose the initial task situation (in this case, situation1 which is illustrated in Figure 3).

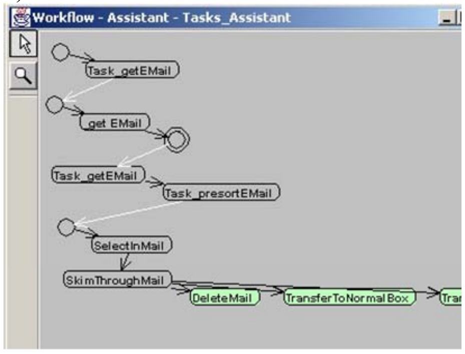


Fig. 13. A task scenario of the tasks of role *Assistant* in ProcessLens.

The integration of knowledge about tasks (actions) and domain objects in TaOSpec allows precise task descriptions. In the example, only one sequential (temporal) operator had to be changed to a concurrent one, in order to solve the problem Mr. Young had with the existing model (see in Figure 4). The precondition on ManageMail (see Figure 5) guarantees that this action is only enabled if there is a message for Mr. Young.

Tool Support 2: Development and Description of the Envisioned Organization of Work

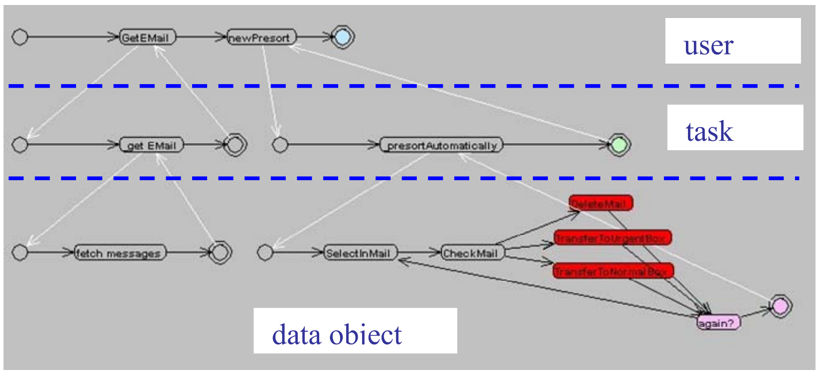


Fig. 14. Synchronization of activity diagrams belonging to the user, task, and data object level according to the envisioned task allocation.

During the creation of the task scenario shown in Figure 13 Patty Smith proposed to automate task PreSort Email (see Figure 8, Figure 9)…

Envisioning organizational developments comprises issues such as the task allocation between humans and software systems. Task modeling tools are useful for discussing such issues. In ProcessLens Patty Smith's proposal can be described and then animated by simplifying the activity diagram of the (human) activity PreSort Email (leading to a single action state _presortAutomatically) and "moving the work" to activity diagrams of the respective data objects, in this case, to the material object Mailbox In, as illustrated in Figure 14.

5. Comparative Analysis

Although each of the described tools and their conceptual foundations have been developed within the model-based tradition of design, and consequently support the representation of tasks, they focus on different aspects: While ProcessLens and CTTE mainly focus on the development of interactive systems, even allowing hands-on UI-experience in case of ProjectLens, TaOSpec targets towards explaining human activities including those in work systems. These differences are reflected through their means for describing task models and task scenarios.

Table 1. Comparative analysis

	task	**task domain**	**actor**	**task ↔ domain**	**task ↔ actor**
TaOSpec	- task = meta action - action hierarchy - explicit temporal relations between sibling actions - predefined operations assigned to basic actions	objects with attributes and state descriptions	one implicit actor	by pre- and post-conditions of actions	none
	- same description mechanism (objects with attributes and partial equations), - instance-pattern relationship				
ProcessLens	- task hierarchy - sequential temporal relations between tasks - activities with behavior	data objects comprising attributes and behavior spec.	organizat. units, roles, persons, incl. behavior spec.	- predefined static relations (e.g., creates) - synchronization of corresponding behavior	predefined static relations (e.g., handles)
	class and activity diagrams to describe structure and behavior of model elements (conform to UML)				
CTTE	- cooperation tree to control task trees - explicit temporal relations between sibling tasks	informal description	roles	none	- one task tree for each role - simple concept of coordination

	task scenario
TaOSpec	sequence of basic actions in a concrete task domain (a set of instances of pattern objects occurring in pre- and post conditions)
ProcessLens	- combination of user and task model: sequence of action states of activities dedicated to a task of an actor - combination of user, task, and data model: sequence of action states of corresponding user, task and data objects - no representation of concrete task domains
CTTE	- sequence of tasks of all roles involved - no representation of a task domain

It turns out that the tools offer different types of presentation of (sub-)models and their relations, with TaOSpec providing textual presentations of models and relationships, ProcessLens and CTTE providing diagrammatic notations for specification. In the concluding section, we propose an integration of different representations.

From the comparative data it also becomes evident that task models seem to be related to cooperation models and workflow descriptions. Some concepts like the cooperative trees in CTTE reflect this fact.

Finally, it can be observed that in none of the tools existing work descriptions are distinguished from envisioned ones (cf. Tool Support 2 in Section 4). A mechanism similar to task domain modeling might be used to capture the temporal scope of task descriptions.

6. Concluding Proposals

Ann Simpson and Simon Brown are responsible for describing the management of incoming inquiries by the staff of ExampleOrg. Usually, they apply the CTTE-tool to represent such task models. However, the tool ProcessLens was introduced in their company three months ago: "I'm happy that I can use activity diagrams to show how tasks are completed.", said Simon who has written a diploma thesis about object-oriented analysis. "I hate these temporal operators in CTTE. I always forget their semantics and precedence."

Ann knows Simon's problem (and his deep task trees with all the "artificial" nodes). "Sure, but I think it should be possible to describe richer temporal constraints between sibling tasks in ProcessLens. In that respect, I prefer CTTE."

6.1 Integrating Different Task Representations

When exploring the reasons for the low acceptance of model-based design approaches (cf. [8]) we have investigated three different task-modeling tools. Although these tools and their underlying theoretical or conceptual base assume similar (sub-) models representing tasks (actions), task domains, and users (actors), we could identify significant differences with respect to formal granularity and semantic expressiveness when describing these (sub-)models. For instance, CTTE does not allow formal specifications of task objects in contrast to TaOSpec. The temporal relationships between tasks are less formal defined in ProcessLens than in CTTE or in TaOSpec. For that reason much of the behavior description has to be moved to the level of activities (as the bottom part of a task hierarchy). Evidently, so far there exists no commonly agreed level of description, either for fine-grained specifications or abstract descriptions.

We know from our experience when teaching task modeling and applying corresponding tools in projects that we need both means to describe sequences of sub-tasks and means to describe states of objects of a task domain. We also have noticed that people accept the idea to assign temporal descriptions to each level of a task hierarchy (as realized in CTTE and TaOSpec) although this strategy restricts the expressiveness of temporal constraints [21]. However, many of them have similar problems as Simon Brown. For example, they introduce nodes into a CTTE-hierarchy which do not play the role of a conceptual sub-task, but rather do allow more sophisticated temporal descriptions.

In order to guide users to accurate modeling dedicated elements help that supports the (partially) separate consideration of hierarchi¬cal and sequential aspects of tasks. For example, a temporal equation can be assigned to each non-basic task T containing all direct sub-tasks of T (cf. [21]). Other representations are possible as well. In this case, temporal equations can be replaced by activity diagrams. Figure 15 shows an abstract example. (Note that ProcessLens does not allow the assignment of activity diagrams to tasks.) It can be shown that each temporal equation with temporal operators like >> (sequence), ||| (concurrency), [] (alternative), […] (option), or * (iteration) can be transformed into a corresponding activity diagram. It is beyond the scope of this paper to give the set of transformation and simplification rules.

TaOSpec offers a hybrid notation of temporal constraints between sub-tasks and constraints on object states (in pre- and post conditions of sub-tasks). For those developers more used to activity diagrams, object flows can support such a hybrid notation.

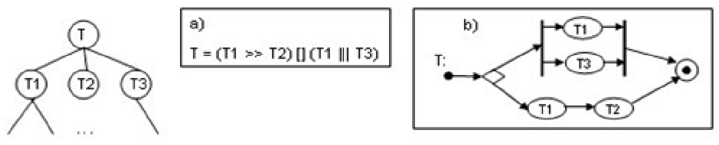
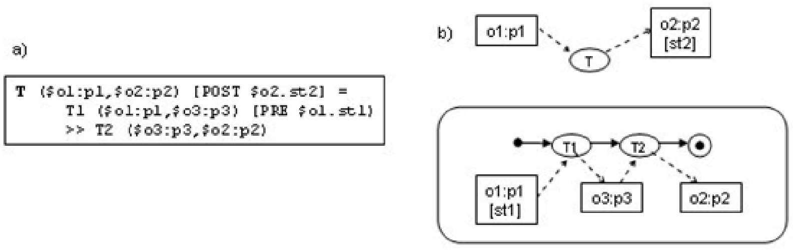


Fig. 15. Part of an abstract task hierarchy of task T (left side), a) a temporal equation assigned to T, b) a corresponding activity diagram.

In Figure 16, a mapping of an abstract fragment in TaOSpec to an activity diagram with object flows is shown. Implicit object flows in TaOSpec become explicit object flows in activity diagrams.



```
T ($o1:p1,$o2:p2) [POST $o2.st2] =
      T1 ($o1:p1,$o3:p3) [PRE $o1.st1)
      >> T2 ($o3:p3,$o2:p2)
```

Fig. 16. Constraints on temporal relations and object states of task T: a) in TaOSpec, b) in activity diagrams with object flows.

6.2 Integrating Different Design Approaches

We suggest not only striving for modeling conventions but also for modeling tools (and the underlying frameworks) that encourage an integrated use of different design approaches. As we have demonstrated, the creation of concrete task scenarios helps to connect model-based and scenario-based ideas. However, in order to achieve that goal elaborated relations between model elements are required, in particular some instance-pattern relationship (cf. Tool Support 1 in Section 4).

An animation of task scenarios at different levels of granularity could also be useful. Existing animation or prototyping techniques could be improved so that users need not to concentrate on the correct use of animation features, but rather on the improvement of the task scenarios and the organization of work.

For each of the tools some kind of self-referential application of scenario- or model-based design ideas could lead to improvements of their user interfaces. For example, more convenient interaction techniques for changing an activity node to a task node, e.g., in ProcessLens, could be achieved through interactive temporal relations.

Overall, a combination of different perspectives on design processes and the creation of different (task) representations could facilitate tool-based task modeling besides creating (task) scenarios. The latter can bridge the gap between formal models and scenarios in a narrative form. An advantage of such a linkage is that concepts like goals which are difficult to formalize can nevertheless control design activities like the development of scenarios (which, in return, influence more formal modeling activities again).

References

1. P. Johnson, S. Wilson. A framework for task based design. Proceedings of VAMMS'93, second Czech-British Symposium, Prague. Ellis Horwood, 1993.

2. A. Puerta, E. Cheng, T. Ou, J. Min. MOBILE: User-centered interface building. Proceedings of the ACM Conf. on Human Aspects on Computing Systems CHI'99. ACM Press, pages 426-433, New York, 1999.
3. XIML: A Universal Language for User Interfaces. http://www.ximl.org.
4. E. O'Neill. User-developer cooperation in software development: building common ground and usable systems. PhD thesis. Queen Mary and Westfield College, Univ. of London, 1998.
5. M.B.Rosson, J.M.Carroll. Usability Engineering – Scenario-Based Devel¬op¬¬ment of Human-Computer Interaction. Morgan Kaufmann Publishers, 2002.
6. L. Constantine, Canonical Abstract Prototypes for Abstract Visual and Interaction Design. In [7].
7. J. Jorge, N. J. Nunes, J. F. e Cunha, editors, DSV-IS 2003 : Issues in Designing New-generation Interactive Systems Proceedings of the Tenth Workshop on the Design, Specification and Verification of Interactive Systems. Nr. LNCS volume 2844, Springer, 2003.
8. H. Trætteberg, P. Molina, N. Nunes. Making model-based UI design practical: Usable and open methods and tools. Workshop at the International Conference on Computer-Aided Design of User Interfaces, CADUI 2004, Madeira, 2004.
9. P. Forbrig, A. Dittmar. Bridging the gap between scenarios and formal models. In C. Stephanidis, Proc. of the HCI International 2003, pages 98-102, Greece, 2003.
10. H. Trætteberg. Model-based User Interface Design. PhD thesis. Norwegian University of Science and Technology - NTNU Trondheim, 2002.
11. Q. Limbourg, C. Pribeanu, J. Vanderdonckt. Towards Uniformed Task Models in a Model-Based Approach. In C. Johnson, editor, DSV-IS 2001, LNCS 2220, pages 165-182, Springer, 2001.
12. A. Dittmar, P. Forbrig. Higher-Order Task Models. In [7].
13. A. Dittmar. Ein formales Metamodell für den aufgabenbasierten Entwurf interaktiver Systeme. PhD thesis. University of Rostock, 2002.
14. M. Stoy. TaOSpec - Implementation einer aktionsorientierten Spezifikationssprache. Studienarbeit, FB Informatik, Univ. Rostock, 2003.
15. C. Stary. TADEUS: Seamless Development of Task-Based and User-Oriented Inter¬fa¬ces. IEEE Transactions on Systems, Man, and Cybernetics, Vol. 30, pp. 509-525, 2000.
16. C. Stary, S. Stoiber. Model-based Electronic Performance Support. In [7].
17. M. Fowler, S. Kendall. UML Distilled - Applying the Standard Object Modeling Language. Addison Wesley, Reading, Massachusetts, 1997.
18. G. Booch, J. Rumbaugh, I. Jacobson. The Unified Modeling Language User Guide. Addison Wesley, 1999.
19. CTTE: The ConcurTaskTree Environment. http://giove.cnuce.cnr.it/ctte.html.
20. F. Paterno, C. Mancini, S. Meniconi. ConcurTaskTrees: A notation for specifying task models. In INTERACT'97, pages 362-369, 1997.
21. A. Dittmar. More precise Descriptions of Temporal Relations within Task Models. In: Palanque, P., Paterno, F., editors, DSV-IS 2000, LNCS 1946, Springer-Verlag, pages 151-168, 2000.

Discussion

[Gerrit van der Veer] As far as I understand, it seemed that all three approaches have no concept like event or trigger. E.g. in your scenario you have an inquiry arriving, but none of these three tools can model this properly, since these all model reactive processes. In real life there should be proactive agents, showing new things arriving from the outside. This is a basic problem with all three tools--they don't model the arrival of new events.

> [Anke Dittmar, Peter Forbrig] This is not in the current analysis, but we think that all three could describe these process interruptions.

[Gerrit van der Veer] Yes, but the tools can only model where the tasks have to be waiting for something to happen.

> [Anke Dittmar, Peter Forbrig] Yes, you are right. We are not interested in modifying modelling concepts; we are just looking at what is being modelled right now. However, you could easily make this change to these tools, to allow that a message is coming from the outside and a task has to respond to it.

[Juergen Ziegler] Towards the end of your talk you showed how you can model this approach to UML activity diagrams. What are the advantages of your approach to activity diagrams or equivalent notations? In your approach you are focusing on the decompositions of tasks instead of the flow aspects. Do you have any rules, in your mapping, as to where in the decomposition you may or may not use sequential or temporal definitions?

> [Anke Dittmar, Peter Forbrig] Our specification is much richer than UML diagrams. For example, UML diagrams cannot specify interrupts. Our notation is much richer, and it can also specify new temporal relations. But it might be useful to present these ideas in UML diagrams. Also, UML (or whatever) specifications are just a means to express task modelling concepts. For example, here we use activity diagrams to represent the relation between siblings within the structure. Also, task models are very simple. For example, they only allow temporal constraints on one level of the hierarchy. So this is restricted, compared with something like Petri nets. So you cannot describe complex temporal relations with this notation.

[Michael Harrison] How do your tools help you to express non-normative behaviours, work-arounds, and errors? For example, attaching (or forgetting to attach) files within the email example.

> [Anke Dittmar, Peter Forbrig] To do this you need to modify the modeling concepts themselves, so that you can combine different task models. But that is not the topic of this talk. Perhaps we can do this in the future with something like an aspect-oriented specification.