Software Development and Open User Communities

P. Forbrig and A. Dittmar

University of Rostock, Department of Computer Science,
Albert-Einstein-Str. 21, D-18051 Rostock, Germany

Abstract. The request for software systems which can be used by different groups of people under different conditions asks for support by new modularization principles on the modeling level.
This paper follows the idea of combining several submodels, as e.g. task and object models, to specify different views on software systems and their situations of use as known from model-based approaches. But further, a separate specification of the general aspects of a model and the constraints imposed by concrete situations is suggested.
At the example of the adapted action model this specification strategy is deeper explained and it is shown how it can contribute to a more flexible modeling for open user communities.

1. Introduction

Human beings apply interactive software systems to fulfil tasks. In this sense, software is nothing else but a tool for us. We use it like a hammer or a mechanical machine to achieve the goals set by the appropriate tasks. Thanks to the development of new interactive techniques, the way a user can perform his task has become less restrictive. Thus, window systems and direct manipulation allow him to execute several subtasks concurrently.
“There is no point in building a system that is functionally correct or efficient if it doesn't support user's tasks or if users cannot employ the interface to understand how the system will achieve task objects” (Duke & Harrison, 1995). Model-based approaches like (Paterno 2000), (Puerta, Cheng, Ou & Min, 1999) or (Wilson, Johnson, Kelly, Cunningham & Markopoulos, 1993) follow this idea. Task-based and object-oriented techniques are used to derive, for example, user interfaces the user can cope with.
In this paper, it is assumed that a software system has to support tasks in a certain area of application which can be specified by a task model. An example of a task model describing the preparation a meal is given in Fig.1. It demonstrates that task modeling is not restricted to any kind of tasks or tools.
We claim, however, that the existing models are to rigid to specify all requirements of interactive systems for open user communities. Besides different user characteristics different constraints of the user’s actual environment have to be considered. Has the user to perform a task alone or does he work in a group? Is he disabled in a certain way? What is his standard of knowledge? Does the user have a large screen or does he use a mobile phone to work with an INTERNET connection?... The term *specific situation of use* refers to such questions.
Let us have a look at the simple example in Fig.1. Basically, the model allows us to prepare potatoes, vegetable and (most important) meat concurrently. There are only general constraints coming from the field of application like washing and tenderizing the meat before roasting it. But, the model neglects individual preferences or abilities as well as specific constraints in the actual environment. Imagine, for example, that a person can handle at most two pots or pans at

the same time. He could prepare the potatoes in one pot, steam the vegetable in a pan and, then, roast the meat 'immediately afterwards' in the same pan to get a hot meal, nevertheless.

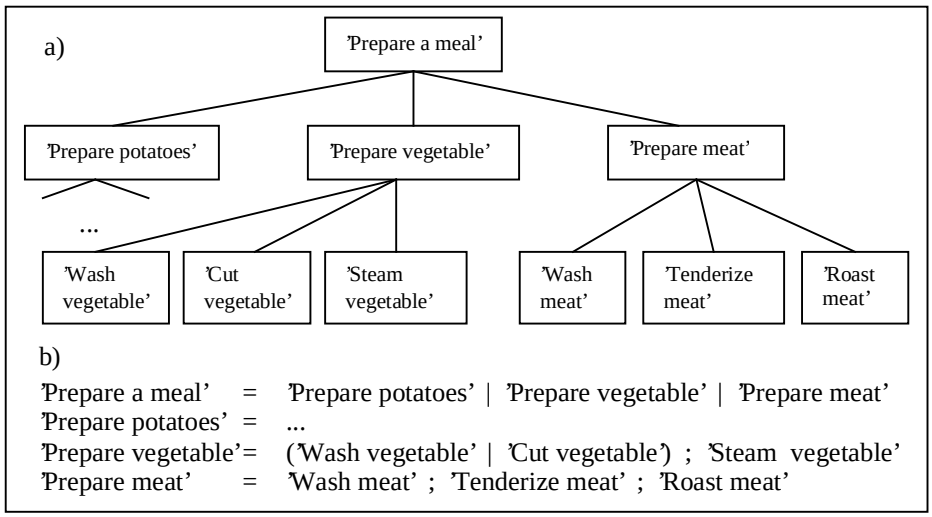

Figure 1. Preparation of a meal

Software design for open user communities asks for support by new modularization principles on the model level. We argue that a separation between kernel models which describe the general requirements imposed by the area of interest and additional models is a step in the requested direction. An additional model can be seen as an extension of the appropriate kernel model adapting it to a specific situation of use.
Before in Sect.3 this more flexible specification process is described Sect.2 gives a proposal to modify a 'classical' task model allowing model adaptations. The paper is closed by a summary.

2. From 'Classical' Task Models to Model Adaptations

A 'classical' or simple task model consists of two parts as to be seen in the abstract example of Fig.2. The *hierarchical description* H decomposes the task T into subtasks T_i until the level of basic tasks. To each basic task T_i an operation op_i is assigned which has to be executed in order to fulfill T_i. The task tree is the most stable part of the task model. It gives an impression what has to be done whereas the *sequential description* S specifies the order in which the operations can be performed to fulfil the whole task. Therefore, in S the task T is considered as a process of a process algebra in the usual way of task-based approaches (e.g. (Paterno, 2000)). S is built up by two types of equations. An equation of the first type describes temporal constraints between those subtasks which have a common parent node in the task tree (equ_T, equ_{T1}, and equ_{T2} in Fig.2). Equations of the second type define the above mentioned mapping between basic tasks and operations of the set OP (equ_{T11}, equ_{T12}, equ_{T21}, and equ_{T22} in the example). For reasons of brevity equations of the last type as well as operations are omitted later on. Temporal dependencies between subtasks can be formulated by well-known operators as used e.g. in (Johnson, Wilson & Markopoulos, 1991). In Fig.2 the temporal operators for parallel (|) and sequential execution (;) of subtasks were applied. Thus, T can be fulfilled by executing one of the following 6 sequences of basic tasks: $\langle T_{11},T_{12},T_{21},T_{22}\rangle$, $\langle T_{11},T_{21},T_{12},T_{22}\rangle$, $\langle T_{11},T_{21},T_{22},T_{12}\rangle$, $\langle T_{21},T_{11},T_{12},T_{22}\rangle$, $\langle T_{21},T_{11},T_{22},T_{12}\rangle$, $\langle T_{21},T_{22},T_{11},T_{12}\rangle$.

In this paper a simple task model is taken to describe the *kernel task model*. Its task tree determines which subtasks can be performed in general. The sequential description contains only those temporal relations which come from the area of application. In Fig.2, T_1 and T_2 could be interpreted as tasks decomposed into further subtasks concerning the input of some data to the

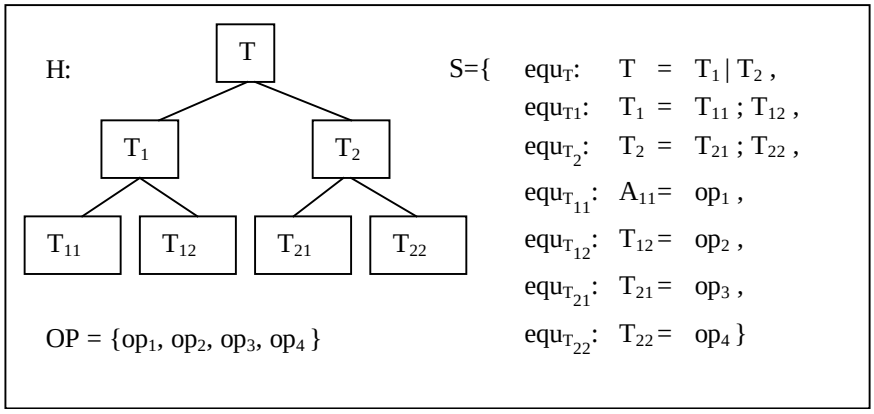

Figure 2. A simple task model for the abstract task T

software system (T_{11} and T_{21}) and the calculation of the appropriate results (T_{12} and T_{22}). Further it is known that the execution of T_{12} and T_{22} takes a long time (even with the best processor). Obviously, T_{11} has to be performed before T_{12} just as T_{21} before T_{22}. But, there are no temporal constraints between T_1 and T_2. A user interface derived from this task model could consists of two windows W_1 and W_2 in parallel, W_1 reflecting T_1, W_2 reflecting T_2. In both windows the calculation is disabled until all input data are given.
Considering open user communities there are also changing constraints arising from the specific situation of use. In our example, a user could work with a small device which can display only one of the windows W_1 and W_2 at the same time. It is blocked until a calculation is finished. The task model could be adapted to this situation by saying that T_1 has to be executed for enabling T_2 because T_1 has a higher priority, for example. (We admit this is only a scholarly example but we hope it is sufficient to explain the idea.)

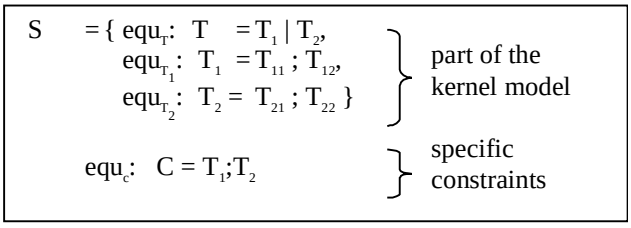

Figure 3. An adaptation of Fig.2

In our approach, a kernel task model is adapted to a specific situation of use by additional temporal constraints noted as equations similar to them in the sequential description of the kernel model. That means that the task hierarchy itself is considered as stable but the set of possible execution sequences can be restricted. The possibility of disabling some of optional subtasks is also included. Fig.3 shows an adaptation of the kernel model in Fig.2.
The above mentioned situation is reflected in the specific constraint equ_C which restricts the 6 possible execution sequences to only one: ($\langle T_{11},T_{12},T_{21},T_{22}\rangle$).

Fig.4 illustrates an adapted model for the introductory example (a person could handle 2 pots or pans at most). The temporal operator ⟨⟨ ⟩⟩ is used to specify the relation ('Roast meat' *immediately afterwards* 'Steam vegetable') because $\langle\langle T_i \rangle\rangle$ means that the execution of subtask T_i has not to be disturbed by any other subtask of the task model.

S = { Prep. meal = Prep. potatoes | Prep. vegetable | Prep. meat,
Prep. potatoes = ... ,
Prep, vegetable = (Wash vegetable | Cut vegetable) ; Steam vegetable,
Prep. meat = Wash meat ; Tenderize meat ; Roast meat }

equ_c: C = ⟨⟨Steam vegetable ; Roast meat⟩⟩

Figure 4. An adaptation of Fig.1

3. A More Flexible Specification of Software

It is widely accepted that the use of different models during the software development leads to more matured systems because they can catch different aspects of the problem space. Most model-based methods distinguish between a model of the existing task situation and the envisioned one. These models are further subdivided into submodels describing the tasks (task model), the environment (business object model) and the users (user model). They support the derivation of a design model of the interactive system. Deeper discussions of this topic can be found in (Forbrig, 1999).

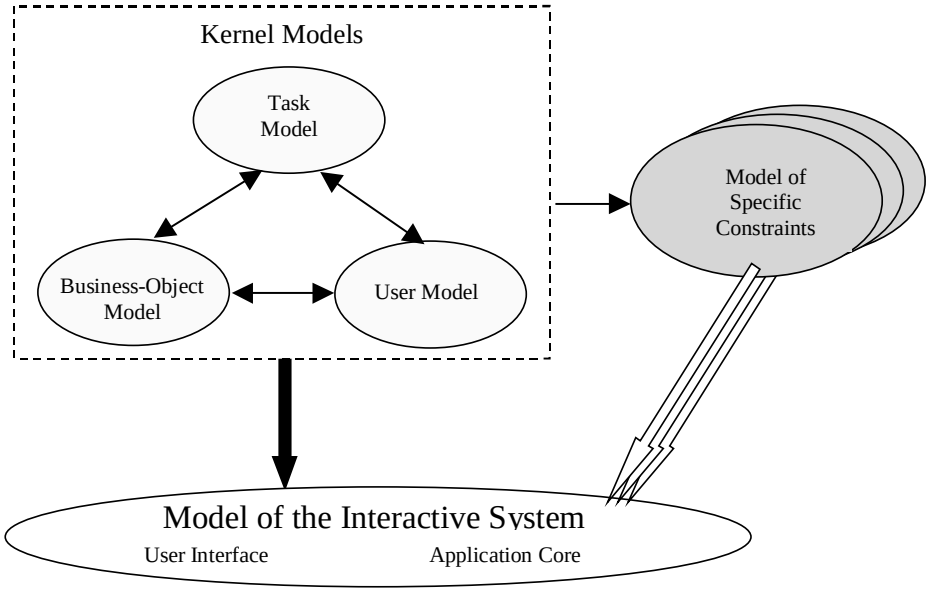

Figure 5. Ontology of the model-based approach

Although such approaches contribute to a flexible specification of software the submodels itself are often to rigid to describe different user needs. Consequently, we are often confronted with

specifications which are either to restrictive for single users or give them too much freedom within the application context.
This paper proposes to separate a submodel into a kernel part and additional ones to allow more flexible specifications. Whereas the kernel part is stable or static (as far as you can consider a model as stable) an additional part depends on the specific situation of use. The general idea of the approach is illustrated in Fig.5. As to be seen the model of the interactive system is derived from the kernel submodels as usual. But in the kernel models only the general constraints given by the field of application are specified. Models of specific constraints can adapt a kernel model to a specific situation of use and change the model of the interactive system dynamically.

4. Summary

Software systems for open user communities have to reflect the different needs of users. Hence, it is reasonable to distinguish even in the specification phase between two parts of description. Firstly, a general one which is valid in all situations of use. Secondly, a specific one which takes into consideration the characteristics of a single user or group of users.
In Sect.2 a proposal of adapting task models was made. Models of specific temporal constraints adapt a kernel model by restriction without changing the nature of it completely. The idea is based on process algebras which allow a very precise specification of temporal relations. A more detailed explanation is given in (Dittmar 2000). We believe that similar mechanisms of adaptation can also be found for the other submodels within a model-based approach.
Of course, there are many open questions. Sometimes it can be difficult to categorize a constraint as general or specific, for example. Nevertheless, we believe that the principle of adaptation can support a more flexible software development.

5. References

Duke, D. J., Harrison, M. D.: Mapping user requirements to implementations. Software Engineering Journal, Vol. 1 (1995) 13-20

Johnson, P., Wilson, S., Markopoulos P.: A framework for task based design (1991)

Paterno, F.: Model-Based Design and Evaluation of Interactive Applications. Springer Verlag, 2000

Puerta, A., Cheng, E., Ou, T., Min, J.: MOBILE: User-Centred Interface Building. In: Proceedings of the ACM Conf. on Human Aspects on Computing Systems CHI'99. ACM Press, New York (1999) 426-433

Wilson, S., Johnson, P., Kelly, C., Cunningham, J., Markopoulos, P.: Beyond hacking: A model based approach to user interface design. In: Alty, J.L., Diaper, D., Guest, S. (eds.): People and Computers VIII, Proceedings of the HCI'93 Conference (1993), 418-423

Dittmar, A: More precise descriptions of temporal relations within task models, ICSE2000, Workshop Design Specification and Verification of Interactive Systems, 2000, LNCS 1946,151-168

Forbrig, P.: Task- and object-oriented development of interactive systems - How many models are necessary?. Proc. DSVIS 99, Braga, 1999, 225-237