Task-based Development Methodology for Collaborative Environments

Maik Wurdel*[1], Daniel Sinnig**, Peter Forbrig*

* University of Rostock, Department of Computer Science, Rostock, Germany
{maik.wurdel, peter.forbrig}@uni-rostock.de

**Concordia University, Faculty of Engineering and Computer Science, Montreal, Canada
d_sinnig@encs.concordia.ca

Abstract. The paper presents a task-based development methodology for collaborative applications. According to our methodology a collaborative task model may be used during analysis, requirements and design. In order to ensure that analysis information is correctly translated into subsequent development phases a refinement relation is proposed supporting the incremental development of task specifications. The development methodology is exemplified by a case study in which interactive support for a conference session is developed.

Keywords: Collaborative Task Models, Development Methodology, Refinement, Tool Support

1 Introduction & Background Information

In modern software engineering, the development lifecycle is divided into a series of iterations. With each iteration a set of disciplines and associated activities are performed while the resulting artifacts are incrementally perfected and refined. The development of cooperative applications is no exception to this rule. Analysis level models are further refined into requirements- and/or design level models, finally resulting in a complete specification of the envisioned collaborative application.

In this paper we define a development methodology for collaborative systems covering the phases from analysis to design. Such an integrated development methodology will serve as a blueprint for practitioners to derive an iterative development process according to which collaborative task models are stepwise refined. For this purpose we analyze the various roles that collaborative task models may play in software development. Moreover, we define a refinement relation for collaborative task models. The practical applicability of our development

[1] Supported by a grant of the German National Research Foundation (DFG), Graduate School 1424, Multimodal Smart Appliance Ensembles for Mobile Applications (MuSAMA)

methodology is demonstrated by a case study in which we develop interactive support for a conference session.

Within the domain of human-computer interaction collaborative task models are widely used for the specification of collaborative (multi-user) interactive systems. Among the most popular ones is Collaborative ConcurTaskTrees (CCTT) [1]. In CCTT modeling starts with the creation of a task model for each involved role in the cooperation. Additionally, a so called "coordinator model" is developed to specify the temporal dependencies of tasks involved in the cooperation. CCTT is suitable for situations where only one actor is fulfilling one role simultaneously. Often, however, this is a too rigid constraint. In order to overcome this shortcoming, we have developed the collaborative task modeling language (CTML) [2]. It is based on the idea that the behavior of an actor can be approximated through her role. CTML incorporates concepts for the specification of interrelation between different actors based on roles, where the behavior of a role is defined by collaborative task expressions. Collaborations of actors are specified by means of an OCL-like notation used to specify preconditions based on the state of the tasks of the involved actors.

The remainder of the paper is structured as follows: in Section 2 we review key principles of CTML, which will serve as foundation for the presented approach. Additionally a refinement relation, based on meta-operators for CTML specifications is proposed. Section 3, the core part of this paper, presents a methodology for the incremental and iterative development of CTML models which is guided by a refinement relation for CTML specifications. In Section 4 we exemplify our methodology by elaborating a small case study. Finally we conclude and give an outlook to future research avenues.

2 The Collaborative Task Modeling Language

Similar to [1], CTML is based on a role-based approach for modeling cooperative task models. Formally, a CTML model is a tuple consisting of a *set of actors*, a *set of roles* and a *set of collaborative task expressions* (one for each role) where each actor belongs to one or more role(s). Each collaborative task expression has the form of a task tree, where nodes are either tasks or temporal operators. Each task is attributed with an *effect* and a *precondition*. An *effect* denotes a state change of the system or environment as a result of task execution. A *precondition* adds an additional execution constraint to a task. In particular a task may be performed only if its precondition is satisfied. Conditions can be either defined over the system state or the state of other tasks (a task life cycle is defined in terms of a state chart [2]), which potentially may be part of another task definition. Both, preconditions and effects are needed to model collaboration and synchronization across collaborative task expressions. The development and simulation of CTML specifications is supported by the tool CTML Editor and Simulator, first introduced in [2].

2.1 Refinement of CTML Specifications

Refinement is a formal process which transforms one specification into another such that required properties of the original specification are preserved [3]. In support of an iterative development methodology we propose, in this section, a refinement relation for CTML models. In [4] we presented a formal approach to define and check refinement between (non-collaborative) task model specifications. In what follows, we extend the approach to CTML specifications in a straightforward manner. Refining collaborative task models can be achieved using two different instruments: Structural and behavioral refinement.

Structural Refinement: The refined CTML model may contain more detailed information than its base model. This is achieved by further refining the atomic units (i.e. the leaf tasks) of the superordinate model. It is, however, important to retain type consistency. Refined tasks need to revise their task type if necessary according to the added subordinate tasks. An exception to this rule are tasks that have been marked with the *deep binding* meta-operator (will be explained in the context of behavioral refinement). These tasks cannot change their task type and the respective subtasks need to be chosen such that type consistency is ensured.

Behavioral Refinement: Whether a behavioral refinement is valid or not depends on the usage of meta-operators in the respective CTML models. Unlike temporal operators, meta-operators do not determine the execution order of tasks, but define which tasks must be retained or may be omitted in the refining CTML model. We distinguish between three different meta-operators: *shallow binding* (⊙), *deep binding* (⊗), and *exempted binding* (⊖). All three operators denote tasks which need to be preserved in all subsequent refining CTML models. While in the case of *shallow binding* subtasks may be omitted during refinement, in the case of *deep binding* all subtasks need to be preserved. Tasks attributed with the *exempt binding* operator have been newly introduced during design and should be preserved in all subsequent refinements.

Details of the algorithm implemented to check refinement can be found in [4].

3 Development Methodology

Current software engineering processes advocate iterative development lifecycles during which artifacts are incrementally perfected and refined [5]. The development of collaborative task models is no exception to this rule. We believe a CTML model is best developed in five steps:

1. Definition of roles and corresponding collaborative task expressions
2. Animation and validation of these sub-specifications
3. Specification of the environment including actors, associated roles and devices
4. Annotation of tasks with precondition and effects
5. Animation and validation of the entire specification

Instead of creating the entire model at once, which can be quite overwhelming, we suggest to first define (1) and test (2) the involved roles and their individual

collaborative task expressions. Both steps can be performed iteratively. In case of an unsatisfying animation the developer typically adapts the underlying specification and restarts the simulation. Next (3) the designer defines the environment and involved actors. Additionally (4) task specifications are completed by adding preconditions and effects based on the analysis of the dependencies between actors and roles. Finally (5) the entire specification consisting of several "concurrently" executing task expressions can be tested and animated. This sequence is to be repeated until the simulation exhibits the expected behavior. Please note that in each stage it is possible to return to any previous step to revise made design decisions, based on evaluation results. Each of the above steps is fully supported by our tool CTML Editor and Simulator.

Fig. 1 indicates that throughout the development lifecycle of a collaborative application different "versions" of a CTML model are used. As will be detailed next, the usage and role of the CTML model vary, depending on the development stage within which it is utilized.

Analysis: The purpose of analysis is to understand the user's behaviors, their collaborations and interactions. Consequently, the analysis CTML model captures the current work situation and highlights elementary domain processes as well as exposes bottlenecks and weaknesses of the problem domain. As portrayed in Fig. 1, the focus is on the actual users while the envisioned interactive system is not yet taken into account.

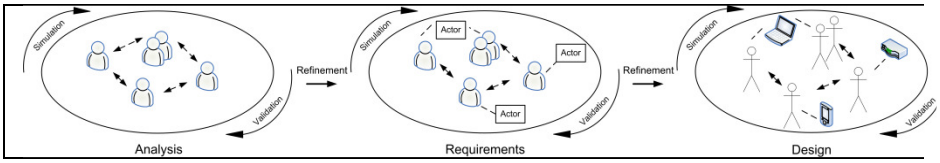

Fig. 1. CTML in the Development Lifecycle

Requirements: When moving to the requirement stage the analysis information is further refined by taking into account the support of the envisioned interactive application. Correspondingly requirements level CTML models specify the envisioned way tasks are performed using the system under development. That is, tasks that were formerly performed by the user may now be taken over by the envisioned interactive system. Generally, the artifacts gathered during requirements specification are part of the contract between stakeholders about the future application.

Design: During design, the various tasks of the requirements model are "instantiated" to a particular target device by taking into account its interaction capabilities. Typically, new design specific, tasks are also introduced. An example of such a design specific task for a conference session management system (will be introduced in Section 4) is "Register Presenter". This task was not part of the analysis or requirements model, but is needed during design such that the session management system is able to keep track of the participating presenters.

When moving from analysis to requirements to design, the collaborative task model is further refined since application and design specific information is added. With each refinement step it is important to verify that the refining model is a valid refinement of its base specification. The interpretation of what constitutes a valid

refinement depends on the artifacts involved, as well as on their purpose in the software lifecycle.

4 Case Study

In this section we showcase the application of the presented development methodology by elaborating a small case study which has as its goal the development of interactive tool support for a conference session. For this purpose let us consider the following scenario:

Before starting the session Peter, the chairman, connects his notebook to the projector installed in the conference room and switches to presentation mode. Afterwards he starts the session by introducing himself and giving a short introduction about the presentations to be given during the session. Then, Peter gives the floor to the first speakers, Daniel and Maik, who give a joined presentation. Daniel connects his notebook to the projector and starts the presentation by briefly introducing the general approach. The technical details are explained by Maik. His slides are stored on his own notebook, which has to be connected to the projector before he presents his ideas. Afterwards, Daniel resumes the talk by giving the conclusion and an outlook for future research which results in an additional reconfiguration of the notebook and the projector. After finishing the talk the chairman asks for questions from the plenum which are answered by the speakers. The subsequent talks are given in ordinary manner until Peter closes the session.

Based on our experiences such a scenario is quite common. The technical burden of state of the art computing devices leads to a tedious and error prone configuration process. But pure automation does not solve this problem. From our point of view a thorough analysis of the collaboration of the actors involved in this process is able to expose where automation is really helpful. The question to be addressed is: “What is the appropriate assistance in the current situation for the actual actor?”

Clearly the scenario shows that actors involved in a joint presentation have to synchronize and agree on who is taking the control of the presentation. Daniel and Maik must not perform the task “Present” concurrently. This is a key collaboration constraint and hence should be taken into account in any corresponding collaborative task model. In Fig. 2 the analysis level CTML model for the joint presentation is given. It is role-based and represents how involved presenters perform their joint presentation. As already hinted by the afore-mentioned scenario, a presenter has to gain control and set up the equipment before presenting his slides. After finishing her/his part the presenter surrenders the control and hence enables other actors to present their parts.

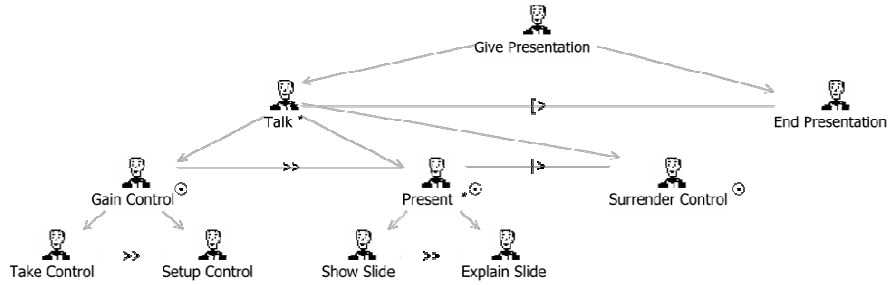

Fig. 2. Analysis Task Model for the Role "Presenter"

The interplay between gaining and surrendering control is modeled using the effects given in Table 1. The effect of an actor performing the task "Gain Control" is that for all other presenters the "Gain Control" task becomes disabled. Conversely, the execution of the task "Surrender Control" enables the "Gain Control" task to all participating presenters among which, one presenter will be able to "Gain Control" of the presentation.

Table 1. Effects of Analysis Task Model for "Presenter"

#	Task	Effect
(1.)	*Gain Control*	*Presenter.allInstances.Gain Control.disable*
(2.)	*Surrender Control*	*Presenter.allInstances.Gain Control.enable*

Before moving to the requirements stage, we have to ensure that pivotal domain specific tasks are preserved in all subsequent refining models. This is done by the use of meta-operators which have been introduced in the previous section. In the context of this case study, the important tasks to be retained are "Gain Control", "Present" and "Surrender Control" and therefore are marked with the shallow *binding operator.*

During the requirement stage new aspects come into play. Compared to the analysis model, the envisioned work situation is enriched by taking into account the support of interactive devices. In our case the interactive support consists of a remote presenter device and a steerable projector. The former can be used to navigate through the slides but also to surrender and gain control of the presentation. The latter can soft-switch between multiple input sources and projection surfaces and hence, can relieve the presenters from manually setting up the equipment (e.g. connecting the laptop to the projector).

As depicted in Fig. 3 the requirements level task model refines the analysis model in terms of structure and behavior. The task "Gain Control" has been structurally refined into interaction and application subtasks denoting how the control of the presentation is gained using the envisioned software system. In particular the execution of the subtask "Assign Control" assigns the control of the remote presenter device and thus to its user. The "Present" task is now regarded as an interaction task since presentations given with the new system are requiring the interaction with the newly introduced remote presenter device. The execution of the "Setup Equipment" task has the effect that the input source of the steerable projector is set to the current actor's laptop. Note that for the sake of simplicity the necessary preconditions and effects are not shown.

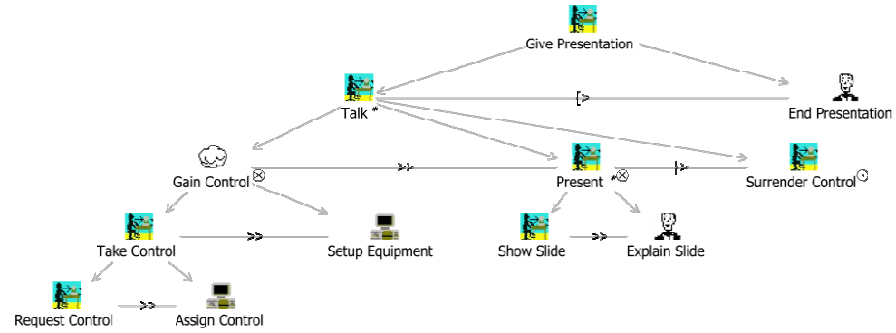

Fig. 3. Requirement Task Model for the Role "Presenter"

In order to ensure that the requirements are preserved in subsequent design models the tasks "Gain Control" and "Present" are marked with the *deep binding* meta-operator. This guarantees that each of these tasks including the subtasks is carried on to the design stage. Additionally "Surrender Control" keeps being marked with the shallow binding operator.

During design, the focus is put on tasks related to the specific interaction with the newly introduced system. Fig. 4 portrays the corresponding task model for our case study. In particular the task "Request Control" has been further refined with subtasks which take into account concrete interactions with the remote presenter (e.g. "Press Request Button"). The same applies for "Surrender Control". Additionally, technology related tasks are introduced. In the context of the case study the presenter has to register her/his remote presenter device to the system ("Register Presenter") before it can be used. The "Register Presenter" task has been attributed with the *exempted binding* operator, denoting that it should be preserved in all subsequent refinements.

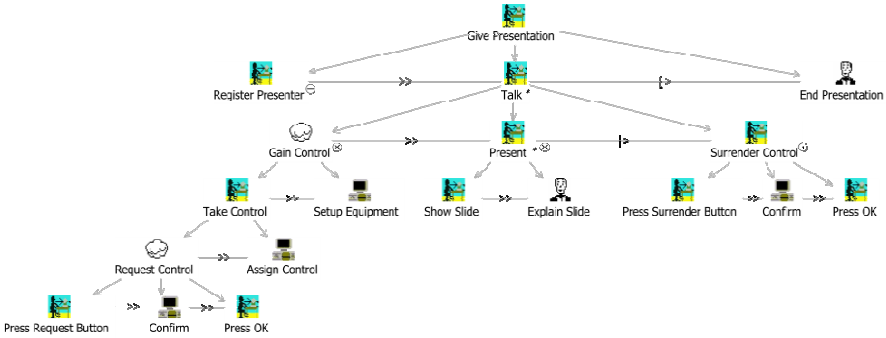

Fig. 4. Design Task Model for the Role "Presenter"

We conclude this section by noting that for each phase (i.e. analysis, requirements and design) we interactively animated the developed CTML models using the tool CTML Editor and Simulator. This was particularly helpful in gradually refining the model until the envisioned behavior was achieved. A snapshot of the interactive animation of the requirements level task model is depicted on the right hand side of Fig. 5. On the left hand side a snapshot of the tool in specification mode is given.

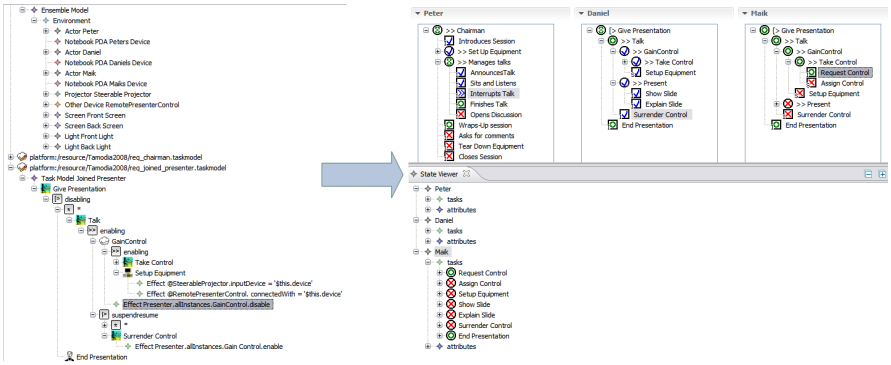

Fig. 5. CTML Editor and Simulator in Specification and Animation Mode

6 Conclusion & Future Work

In this paper we presented a development methodology for collaborative task models. In particular, we proposed a set of steps for the incremental development of CTML models. Each step is supported by our tool the CTML Editor and Simulator. We explored the different roles of a CTML model within the development lifecycle of a collaborative application. In particular we proposed a development methodology according to which an analysis level CTML model is further refined to a requirements and design level model. Finally we validated and illustrated our proposed development methodology by elaborating a small case study, which had as its goal the development of interactive support for a conference session.

As future work we are currently investigating how CTML can be integrated into state of the art model-based UI development processes for collaborative environments. Another future avenue deals with the enhancement of the CTML Editor and Simulator with model checking capabilities such that the tool will be able to prove certain properties of a CTML model (e.g. livelock and deadlock freedom) and mechanizes the verification of refinement between CTML specifications.

References

1. Mori, G., F. Paternò;, and C. Santoro, *CTTE: Support for Developing and Analyzing Task Models for Interactive System Design.* IEEE Trans. Softw. Eng., 2002. **28**(8): p. 797-813.
2. Wurdel, M., D. Sinnig, and P. Forbrig, *Towards a Formal Task-based Specification Framework for Collaborative Environments*, in *CADUI 2008*. 2008: Albacete, Spain.
3. Bowen, J. and S. Reeves. *Refinement for User Interface Designs*. in *FMIS 2007*. 2007. Lancaster, UK.
4. Wurdel, M., D. Sinnig, and P. Forbrig, *Task Model Refinement with Meta Operators*, in *DSV-IS 2007*. 2008: Kingston, Canada.
5. Larman, C., *Applying UML and Patterns: An Introduction to Object-Oriented Analysis and Design and Iterative Development (3rd Edition)*. 2004: Prentice Hall PTR.