Towards an Holistic Understanding of Tasks, Objects and Location in Collaborative Environments

Maik Wurdel

University of Rostock, Department of Computer Science
Albert-Einstein-Str. 21, 18059 Rostock, Germany
maik.wurdel@uni-rostock.de

Abstract. In this paper a task modeling approach is presented which tackles the integration of different kinds of models by a generic framework. The application of the framework is shown for collaborative environments, a certain sub set of ubiquitous computing environments. In these environments tasks have a close bond to the location of the executing actor as well the state of the involved objects. Therefore a location specification and a domain model are used to constrain the task execution. The language is supported by a tool, the CTML editor and simulator, which covers all steps of development from creation, editing, testing and verification. Such a model is particularly from interest for the intention recognition module of our experimental infrastructure of a collaborative environment.

Keywords: Collaborative Task Modeling, Domain Modeling, Collaborative Environments, Location Modeling.

1 Introduction

Task modeling has been accepted as mature research field within the domain of HCI. Especially in early stages of development task models specify the potential interaction to accomplish a certain goal with the software system. It can act as bridge between stakeholder and software engineer to manifest the envisioned interaction of user and software system. For desktop and mobile applications common notations such as [1, 2, 3] are well suited. However when using task models to specify the potential task performance in ubiquitous computing environments additional factors come into play. In this paper the question is tackled whether extended task models can be used to specify tasks in collaborative environments (CE), a particular sub set of ubiquitous environments.

A collaborative environment is a physical place equipped with computing devices where actors interact among each other to accomplish a certain goal. Stationary und personal devices assist the actors. Modeling tasks for such an environment is evidently much more complex as classical task modeling. A suitable modeling language has to incorporate means for diverse dependencies. We found the following dependencies particular for CE:

M. Kurosu (Ed.): Human Centered Design, HCII 2009, LNCS 5619, pp. 357–366, 2009.

1. **Cooperation.** Tasks need to synchronize across different actors. Common task model notations offer the opportunity to use temporal operators in order to specify the potential execution order but only some offer temporal constraints for multi-user task specifications.
2. **Objects.** To perform a task tools or artifacts may be needed. This has to be reflected by the modeling language to create an adequate model of the environment. Objects may need a certain state to be in and may also be manipulated (e.g. a presentation can only be given if a projector is available).
3. **Location.** In CEs the place of an actor highly influences the executability of tasks. Some task might only be enabled at a special place in the environment whereas others might be executable at a set of location (e.g. giving a talk at a conference, listening to a presentation).

Such a modeling language is particular relevant for intention recognition in collaborative systems which, based on the delivered model and sensor data provided by the environment, derive the intention of the actors. In this paper we illustrate the rationale of the Collaborative Task Modeling Language (CTML) and focus on a new component which integrates task modeling and location modeling for CEs.

The remainder of the paper is structured as follows: In Section 2 we discuss related work from different domains and highlight assets and drawbacks of the presented approaches. The following section introduces the general modeling approach of CTML. Subsequently location modeling in the context of CEs is discussed and the selected modeling technique is shown. Section 5 gives an overview of our tool environment to create, share and animate CTML specifications. Finally we conclude and give an outlook for future research avenues.

2 Background Information

In this section the reader is reminded of core concepts involved in classical task modeling and location modeling. We examine existing approaches in collaborative task model modeling and review commonly used formalisms and relevant related work.

Task analysis and task modeling is a well-established research field in HCI. Various notations for task modeling exist. Among the most popular ones are GOMS [4], HTA [5], CTT [2], and WTM [6]. Even though all notations differ in terms of presentation, level of formality and expressiveness, they assume the following common tenet: tasks are performed to achieve a certain goal. Moreover, complex tasks are decomposed into more basic tasks until an atomic level has been reached. Within the domain of HCI, CTT is the most popular notation, as it contains the richest set of temporal operators and it is supported by a tool, CTTE [7], which facilitates the creation, visualization and sharing of task models. A comprehensive overview on CTT can be found in [2].

In order to support the specification of collaborative (multi-user) interactive systems, CTT has been extended to CCTT (Collaborative ConcurTaskTrees) [7]. Similar to the corporative task modeling language presented in this paper, CCTT uses a role-based approach. A CCTT specification consists of multiple task trees. One task tree for each involved user role and one task tree that acts as a “coordinator” and

specifies the collaboration and global interaction between involved user roles. The main shortcoming of CCTT is that the language does not provide means to model several actors simultaneously fulfilling the same role as well as that an actor is assumed to fulfill only one role within a CCTT specification (strict one to one mapping of actors and roles). We try to overcome this limitation by the task modeling language presented here.

Tasks are always performed within a certain context or environment and hence their interplay with the environment should also be taken into account. This issue was first tackled by Dittmar and Forbrig [8] who proposed modeling the execution environment in accordance with the task specification. The environment captures the domain entities which are manipulated, created or needed for the performance of a certain task. Unfortunately the approach by Dittmar and Forbrig is not very well integrated with standard software engineering models. We try to overcome this shortcoming by proposing UML class diagrams [9] as a notation to model the environment (domain) model while instances of the model are represented using UML object diagrams [9]. However, the environment of task execution is not limited to domain objects but several other facts can be taken into account. Present devices with their various capabilities and constraints as well as the position of the task performing actor and involved objects need to be taken into account when modeling tasks in CEs.

In [10], Molina et al. propose a generic methodology for the development of groupware user interfaces. The approach defines several interrelated models capturing roles, tasks, interactions, domain objects and presentation elements. Even though, the models cover all important aspects involved in groupware user interfaces, they are only used at the analysis stage. Subsequent development phases (e.g. requirements or design) are not covered. The methodology is not assisted by a tool which would facilitate the creation and simulation of the various models. In particular, the latter is an important shortcoming since the animation of models is an important technique to obtain stakeholder's feedback. The same disadvantages apply to the work of [11] where a modeling approach for collaborative systems is presented. The approach only covers the analysis phase. An execution semantics and tool support for the various models is not offered.

As already pointed out, the environment of task execution needs to be taken into consideration while modeling tasks. In CEs, where tasks are carried out in a physical environment, the location of the performing actors as well as the involved objects have an impact on the execution of tasks. On one hand an actor may need physical access to tools or artifacts to perform a certain task, on the other hand tasks might only be executable at a spatial position of the environment.

Different attempts have been made to model the spatial relation of objects and actors in physical environments. There are geometric models which define the spatial relation by coordinates of the objects. Applications can easily derive containment relation of objects and location. A disadvantage of those models is that properties like *is connected to* are not easy to derive.

Graph-based location models explicitly model this relation. A node specifies a location and edges represent connections between locations. Weights can be added to model distances between locations. Another type of models uses sets to specify locations and their decomposition into sub-locations. An atomic location is specified

by a shape and position. Composed locations are defined by a set of existing locations. The containment relation of locations can be easily expressed using sub-set relations. Hierarchical models are also based on a set of locations which are ordered according to the containment relation. The most used types of model combine several modeling approaches to suit the special needs of the application [12].

3 The Modeling Approach: CTML Specifications

In this section we present the collaborative task modeling language (CTML). First we point out the basic structure of CTML specifications and give reason why this structure suits best. Based upon that, we go into details about the semantics of CTML specifications. As a running example a conference session will be used.

A more comprehensive description of the syntax and semantics of CTML specification can be found in [13].

The design of CTML is based on three fundamental assumptions:

1. In limited and well-defined domains the behavior of an actor can be approximated through her/his role.
2. The behavior of each role can be adequately expressed by an associated collaborative task expression.
3. The execution of tasks may depend on the current state of the environment and in turn may lead to a state modification. The state is defined by the accumulation of states given by various models (such as the domain model).

Table 1. Entities of CTML specifications

Entity	Description
Role	A role specifies a stereotypical user of the environment in the current domain (e.g. Chairman and Presenter in a conference session).
Collaborative task expression	A task expression provides a behavioral description of the assigned role. It may contain task dependencies to other task expression.
External models	External models are used to specify dependencies to relevant entities whose state can be used to define task execution constraints (such as the domain or location model).
Configuration	A configuration specifies a runtime configuration of the CTML specification. It defines the set of actors including the assigned roles as well as the domain model instance (object model). A runtime configuration is the handle to animate the CTML specification.

Based on these assumptions we define a collaborative task model as a tuple consisting of a *set of roles*, a *set of collaborative task expressions* (one for each role), a *set of external models* (such as the *domain model* or *location model*) and a *set of configuration*s (see Table 1.).

Different types of relation between these entities exist: *depends, uses*. A role may *depend* on tasks of another role which means that the task performance of the target role needs the execution of certain tasks from the source role. The *uses* relation specifies that certain modeled objects are needed to accomplish the task set of the target role. Additionally the referenced objects can be manipulated via task execution. A high level view on a CTML specification is depicted in Figure 1.

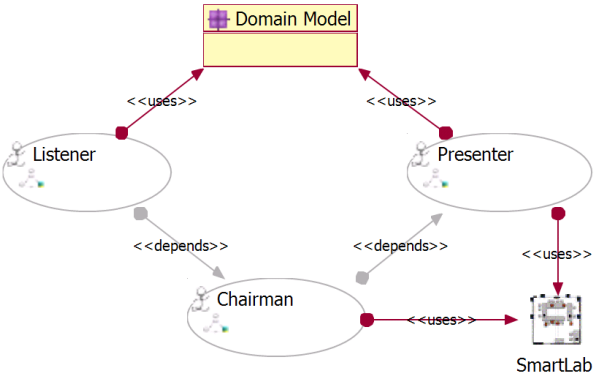

Fig. 1. CTML – Cooperation Model for "Conference Session Scenario"

As already briefly mentioned a configuration defines the runtime configuration of the model. A set of configuration may exist in a CTML specification which makes the model more flexible. A configuration contains the set of actor as well as their assignment to roles. Each actor belongs to one or more role(s). Additionally the instances of models, if necessary, are specified (the initial state of the object model corresponding to the domain model). Figure 2 shows such a configuration. The actors within the model are *Peter* and *Maik* who act as *Chairman* and/or *Presenter*.

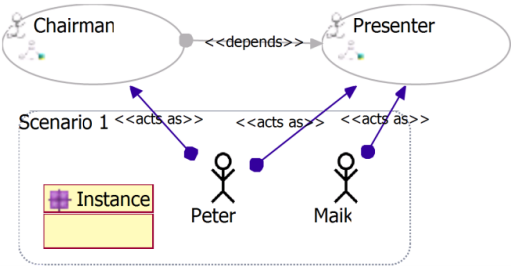

Fig. 2. Configuration *Scenario 1* of CTML Specification for "Conference Session Scenario"

The behavioral description of a role is defined by means of collaborative task expression which has the form a CTT-like task tree [2]. Besides the hierarchical structure and temporal operators a collaborative task expression consists of a set of task, each defined by an identifier, a task type (similar to CTT), a set of preconditions and effects.

Preconditions add additional execution constraints to a task as a task may only be performed if its preconditions are satisfied. Conditions can either be defined over the state of other tasks (enabled, disabled, running, suspended etc.), which potentially may be part of another task expression, or the state of external models integrated into CTML (e.g. domain and location model). An effect denotes a state change of the system or environment as a result of a task execution. Both, preconditions and effects

are needed to model collaboration and synchronization across collaborative task expressions. They also denote the binding to the external models.

3.1 Domain Modeling

An instance of a model integrated in CTML is the domain model. The domain model reflects the fundamentals of the domain: the involved entity and their structure as well as their interrelation [14]. In our case we consider the involved objects needed to execute a task within the CE as domain objects. To model these objects we employ UML class diagrams [9]. Instances, representing a state of all objects, are represented by object diagrams. The initial object state is modeled and imported into a configuration (see Figure 2.).

To constrain the task execution of an actor fulfilling a role one can specify precondition using OCL syntax (Object Constraint Language of the UML [9]). The same applies for effects but a slightly adapted syntax is used since OCL offers purely querying capabilities. For more detailed information consult [15].

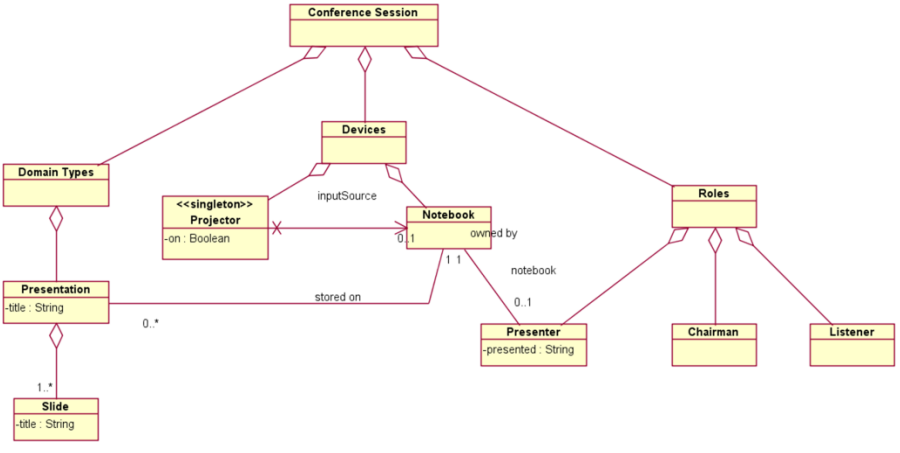

Fig. 3. Domain Model of the "Conference Session Scenario"

3.2 Execution Semantics

After defining the core concepts and involved entities we pinpoint the execution semantics of CTML specifications. A well-defined execution semantic can not only rule out ambiguities but also allow for animation. This is particular important for the creation of prototypes and derivation of lower level models used to operate the collaborative environment [16, 17].

To clarify the execution semantic the runtime model has to be investigated first. To animate a CTML specification the selected configuration has to be transformed into a CTML animation. This step is called instantiation and creates for every actor the corresponding task model animation. This animation is determined by the role assignment of the actor. For each assigned role the corresponding collaborative task expression is translated into a task animation expression. These animations are

composed via the temporal operator interleaving (also known as concurrent). In this vein it is specified that roles can be fulfilled concurrently.

Whether a task is enabled during animation depends upon three different factors: (1.) the task-subtask decomposition, (2.) the defined temporal operators, and (3.) the precondition specified for the task. (1.) and (2.) are commonly known concepts from task analysis and modeling. Even though there are notations which support preconditions (3.) their usage is most often limited to informal precondition in text form. Only in [8] the authors use automatically evaluable preconditions to constraint the task execution. Our approach goes beyond that by offering different types of precondition to integrate different models. So far, there are four types of preconditions as depicted in Table 2.

Table 2. Types of Preconditions

Type	Rationale
Simple Task Precondition	This precondition enhances temporal operator by adding additional execution constraint within the collaborative task expression of the role.
Cooperative Task Precondition	This precondition addresses tasks of other roles defined in the CTML specifications. Because there may exist several actors fulfilling the same role these statements need to be quantified using the *allInstance*, *noInstance* or *oneInstance* quantifier. Those statements address multiple tasks, namely of each actor fulfilling the role.
Domain Precondition	Precondition defined in OCL are used to restrict the task execution with regard to the domain model.
Location Precondition	Precondition addressing the location of the actor performing the task (see Section 4).

In the same vein we defined different types of effects to reflect how the performance of a task changes the system state. In Figure 3 a visualization of a task modeling animation for the conference session is given.

4 Location Modeling

So far we have introduced the Collaborative Task Modeling Language and have given the reader a hint that location modeling needs to be integrated with task modeling for the application of CEs.

In the context of CTML this can be achieved by integrating location preconditions and effects into CTML specifications which was already sketched in the last section (see Table 2). In order to define proper location dependencies a conceptual model of the location needs to be introduced. On one hand the model has to be powerful enough to express complex location specifications for a CE, on the other hand specifying locations should not be a burden.

Therefore we use a local geometric model which uses a simple set of geometric figures (rectangle, ellipse, point) and their compositions. In Figure 5 a screenshot of the location editor is shown. It uses a 2-d model of our experimental infrastructure at our university as background to ease location modeling. The orange figures (located on top of the 2-d model) are simple figures (rectangle, ellipse, point) whereas blue figures (located on the right hand side) denote composed locations. Thus, complex locations can be modeled by composing simple ones. To make use of the defined

locations the model has to be imported into a cooperation model and referenced via the *uses* relation (see Figure 1). This relation expresses that all defined location can be used to constrain the task execution of the role involved in the relation. Informal this means that the behavior of the role fulfilling actor depends upon the position of the actor in the CE.

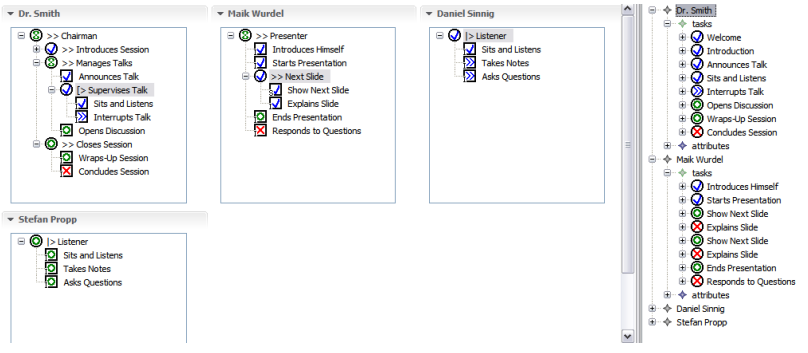

Fig. 4. Interactive Animation of the Session Scenario with four Actors

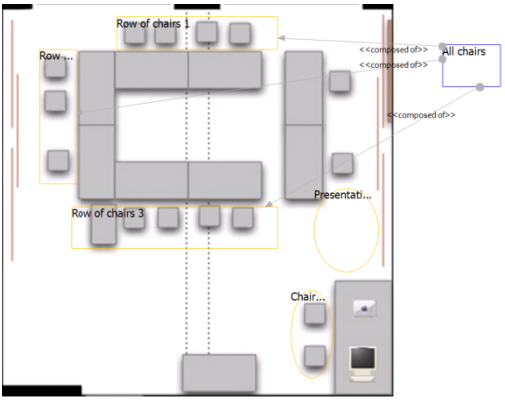

Fig. 5. Location model of Experimental CE "SmartLab" for the session scenario

Locations can either be used to define preconditions or effects. Evidently a location precondition defines that a certain task can only be performed if the actor is located within location referenced via the precondition. Performing a task can also change the position of the actor which is considered as an effect of the task execution in our model. Therefore an effect defines that an actor performing a task moves from one location to another.

5 Conclusion and Future Research

In this paper we have introduced a holistic approach to task modeling, namely the Collaborative Task Modeling Language, which offers a generic framework to

integrate external models. The instruments used for integration are preconditions and effects of tasks. The integration was already accomplished for the domain and location model. Because CTML specifications can become quite complex an editor and interpreter was implemented which allows for interactive animation of CTML specification in order to ease the design and foster the understanding (see Figure 4.).

Within our research project an experimental infrastructure, namely "SmartLab", has been set up. To provide proactive assistance the intention of the users has to be determined. The intention recognition module of the system uses as input the potential execution order of tasks (an acyclic graph) plus the position data of users over time. Because CTML specifications have a well-defined semantics an acyclic graph of the potential task execution order can be derived straightforward by the use of the interpreter. Evidently task models can be created much handier than acyclic graphs [17]. We are currently setting up the interface to the system.

For future research avenues we consider the integration of a device model reflecting the potential task support by devices. At present we think that a state chart based approach is most suited [9]. A device can be defined as a class within the domain model which represents the static structure of the entity whereas the behavioral description is delivered through a state chart. States and transitions can be referenced in preconditions and effects of tasks. Due to the structure of the framework other model types can be integrated as well, such as sensor data or contextual information.

Another issue which needs further investigation is the team model. Usually individual tasks are not only influenced by the environment state and task of other actors but by a higher order model. If we consider a session at a conference, there is usually an agenda which defines when a certain talk is given. Clearly such a model is a sort of task model as well. We think that CTML specification can be used as notation for such a team model. The relation of team tasks to individual tasks will be examined in future research as well.

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

References

1. van der Veer, G., Lenting, B., Bergevoet, B.: GTA: Groupware task analysis - modeling complexity. Acta Psychologica 91, 297–332 (1996)
2. Paterno, F.: Model-Based Design and Evaluation of Interactive Applications. Springer, London (1999)
3. Dittmar, A., Forbrig, P.: The Influence of Improved Task Models on Dialogues. In: CADUI, pp. 1–14 (2004)
4. John, B.E., Kieras, D.E.: The GOMS family of user interface analysis techniques: comparison and contrast. ACM Transactions on Computer-Human Interaction 3(4), 320–351 (1996)
5. Annett, J., Duncan, K.D.: Task Analysis and Training Design. Journal of Occupational Psychology 41, 211–221 (1967)

6. Bomsdorf, B.: The WebTaskModel Approach to Web Process Modelling. In: Winckler, M., Johnson, H., Palanque, P. (eds.) TAMODIA 2007. LNCS, vol. 4849, pp. 240–253. Springer, Heidelberg (2007)
7. Mori, G., Paternò, F., Santoro, C.: CTTE: Support for Developing and Analyzing Task Models for Interactive System Design. IEEE Trans. Softw. Eng. 28(8), 797–813 (2002)
8. Dittmar, A., Forbrig, P.: Higher-order Task Models. In: Jorge, J.A., Jardim Nunes, N., Falcão e Cunha, J. (eds.) DSV-IS 2003. LNCS, vol. 2844, pp. 187–202. Springer, Heidelberg (2003)
9. UML, Unified Modeling Language, `http://www.uml.org/` (accessed July 10, 2008)
10. Molina, A.I., Redondo, M.A., Ortega, M., Hoppe, U.: CIAM: A Methodology for the Development of Groupware User Interfaces. Journal of Universal Computer Science 14, 1435–1446 (2008)
11. Penichet, V.M.R., Lozano, M.D., Gallud, J.A., Tesoriero, R.: Analysis models for user interface development in collaborative systems. In: CADUI 2008, Alabcete, Spain (2008)
12. Becker, C., Dürr, F.: On location models for ubiquitous computing. Personal Ubiquitous Computing 9(1), 20–31 (2005)
13. Wurdel, M., Sinnig, D., Forbrig, P.: Towards a Formal Task-based Specification Framework for Collaborative Environments. In: CADUI 2008, Albacete, Spain (2008)
14. Sommerville, I.: Software Engineering (Update), 8th edn. International Computer Science. Addison-Wesley Longman Publishing Co., Inc., Boston (2006)
15. Wurdel, M., Sinnig, D., Forbrig, P.: CTML: Domain and Task Modeling for Collaborative Environments. JUCS, 14(human-Computer Interaction) (2008)
16. Giersich, M., Forbrig, P., Fuchs, G., Kirste, T., Reichart, D., Schumann, H.: Towards an Integrated Approach for Task Modeling and Human Behavior Recognition. In: Jacko, J.A. (ed.) HCI 2007. LNCS, vol. 4550, pp. 1109–1118. Springer, Heidelberg (2007)
17. Wurdel, M., Burghardt, C., Forbrig, P.: Supporting Ambient Environments by Extended Task Models. In: Proc. AMI 2007 Workshop on MDSE for AmI Applications (2007)