

MODELLING USER TASKS WITH THE UNIFIED MODELLING LANGUAGE

Panos Markopoulos

*IPO, Center for User-System Interaction, Eindhoven University of Technology
Den Dolech 2, Eindhoven 5600 MB, The Netherlands
E-mail: PMarkopoulos@tue.nl*

Abstract: This paper discusses the extension of the Unified Modelling Language (UML) for modelling tasks and user groups. The term *task* is used widely, to include collaborative and multi-user tasks and without reference to some specific theory of user task knowledge or behaviour. The paper presents the proposed extension to the UML, focusing on the concepts involved and the abstract syntax for the notation introduced, comparing it to related formalisms and other extensions to the UML. *Copyright 2001 IFAC.*

Keywords: Interdisciplinary Design, Information Technology, Object Modelling Techniques.

1. INTRODUCTION

This paper discusses the extension of the Unified Modelling Language (UML), see OMG (1999), to support interaction design. There are several models that can help interaction design. These models appear in the literature in different variations, semantic and syntactic. This paper focuses on modelling tasks, groups of users and their roles. The term *task* is used here in its broad sense, including collaborative and multi-user tasks and it is not tied to some specific theory of task-related knowledge or behaviour. The intended users/audience for this extension to the UML are interactive software developers and interaction designers. In small projects, it is conceivable that the same individuals assume these two distinct roles. However, an important consideration for this extension of the UML is that often interaction designers do not have software expertise themselves, but do, however, need to work closely with software developers and need to communicate their designs effectively to them.

Task analysis and user profiling are widely recommended design activities, see, for example, some of the most widely read 'interaction design method' textbooks such as Mayhew (1999) and Constantine et al. (1999). Evident in these textbooks alone, but also in research-workshop reports, c.f. Bomsdorf et al. (1998) and Johnson (1996), is one of the major problems for practitioners, which is the lack of a common set of abstractions and of common

representation techniques by which interaction requirements and designs can be communicated. On the other hand, research in the fields of human computer interaction and cognitive engineering has provided a wide gamut of task analysis techniques and various ways of modeling user populations, organisations and business processes. Considering this diversity of approaches and of their theoretical underpinnings, a remarkable consensus emerges as to the basic concepts that need to be modelled, and as to the need for a common representation scheme, cf. Bomsdorf et al. (1998). However, notations to fulfill this need have not yet been established. Radically different notations disguise the similarity of the concepts represented - see for example the discussion of Johnson (1996). Furthermore, several terms are overloaded, e.g., the terms scenario, task, etc., causing considerable confusion.

The UML is a powerful notation for expressing structural and behavioural characteristics of software. Interaction design and task modelling lie outside its scope of application, but the UML provides mechanisms that allow it to be extended to such domains. The apparent popularity and wide acceptance of the UML within the software development world, provide an opportunity to represent interaction designs in a way that is understandable to a good proportion of software developers. Anticipated improvements in communication between designers and software

engineers can, potentially, lead to more usable systems and cost-savings for development teams.

The extension of the UML and its adoption as a scheme for specifying models used in interaction design is a topic of intense research and industrial interest. There has been a series of workshops on this topic. Some have been associated with major conferences in human-computer interaction, e.g., Artim et al. (1998). Other workshops have been linked to software engineering conferences, e.g., Nunes et al. (1999). Most recently, the TUPIS 2000 workshop (Web: math.uma.pt/tupis00), was linked to the UML 2000 conference, with the explicit aim to work towards an extension of the UML for interactive systems design. The TUPIS workshop was structured along two discussion topics: modelling tasks and modelling interaction spaces, respectively focusing on the user and the system side of the interaction. The discussions of the 'interaction spaces' group have been summarised in Anderson (2000). The discussion on task modelling produced some agreement as to the common abstractions needed for modeling tasks and user populations. This paper builds on the results of that discussion proposing an extension to the UML for task modelling.

Section 2 discusses related work in modelling tasks and user populations. Section 3 motivates the use of task models in design. Section 4 discusses the main concepts involved in modelling tasks: the results of the TUPIS workshop are summarised and a more concise model is proposed. Section 5, presents how these concepts are rendered in the UML, and an example is given in section 6. Section 7 describes the conclusions of this paper.

2. RELATED WORK

For reasons of brevity this section does not attempt a comprehensive survey of research into modelling users and their tasks. It is instructive however to review two influential approaches that seem to have achieved a large popularity: Contextual Design, (Beyer et al. 1998) and Usage Centered Design (Constantine et al. 1998). Both approaches provide a range of methodological and notational constructs to support the design of interactive systems.

Notations used for Contextual Design help represent context, whether organisational, cultural or physical. These notations are very apt for transgressing from the specific observations collected during analysis to a synthesis of a model that generalises across observation sessions and different users. It is debatable whether all the models deployed within this method would be more useful for their purpose (in supporting analysis) if they were formalised in a standard notation such as the UML. However, in

their current form, there is a considerable distance that needs to be bridged, before they can serve to guide the design and specification of the software.

Compared to Contextual Design, Usage Centred provides a more structured approach to modelling user roles and tasks. Tasks are modelled by essential use-cases which are akin to UML's use cases. A significant difference is that essential use cases describe, in an abstract form, the *goal directed interaction* of a single user with the system. Other representations utilised by Usage Centred Design are navigation and content models.

The research reported hereby aims to develop a minimal extension to the UML, that supports interaction design through representations of various models that are useful in design, but in a manner that is neutral to any particular design methodology. With respect to task models, their content and their use are quite in line with the Usage Centred Design method (Constantine et al. 1999), which is effectively an example of a task based design approach (cf. Wilson et al. 1996). Concerning the representation of tasks, rather than the tabular format of essential use cases, a diagrammatic notation is sought, that will illustrate diagrammatically hierarchical task decomposition.

An influential tree-based representation of tasks is the ConcurTaskTrees (CTT) notation by Paternó (2000). Compared to essential use cases, CTT provides a powerful set of operators that help express diagrammatically complex temporal and logical relations between tasks. CTT provides increased expressive power to the modeller, but these operators also can make the notation less accessible to non-computer scientists. In practice, it seems that analysts at the initial requirements capture phases of a project make limited use of the CTT operators, focusing on the hierarchical task structure. At more advanced phases of the design process, software developers rely on the richer operators of CTT to achieve higher precision.

Related to the research reported hereby is the Wisdom method by Nunes et al. (2000). The Wisdom method is associated with a UML profile: a subset of the UML and a set of extensions that are specifically designed to support interaction design using this method. In Wisdom the notion of a task is rather narrow: Task models are not proposed as a technique for representing user requirements, but rather as abstractions of software control entities. Wisdom models tasks using class diagrams, and stereotyped associations patterned upon the CTT notation. This is a very powerful approach that is accompanied by means to specify interactive objects and the navigation between different interaction contexts.

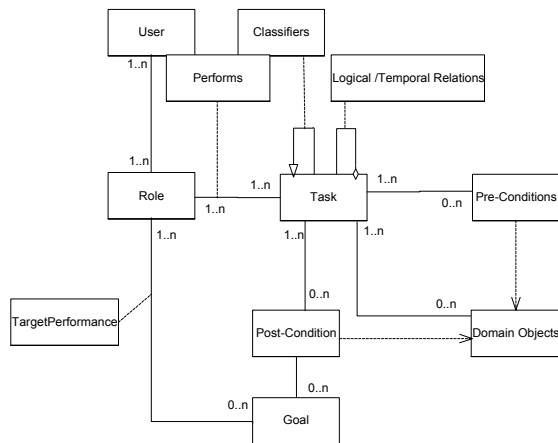


Fig. 1. Abstract syntax for representing tasks

3. TASK MODELS IN DESIGN

As has been mentioned already, the proliferation of models and notations, for overlapping purposes, and with overloaded terminology leads to confusion about differences and similarities of the various models. This issue was discussed in Markopoulos et al. (2000) and in Markopoulos (2000). That discussion concluded that the most common interaction representational needs for interaction design are served sufficiently by 6 models plus the binding representation of scenarios (where scenarios are understood here as narrative, concrete-rich descriptions of interaction episodes in the form advocated by Carrol (1997)). The 6 models are: Task models, Domain Models, User Population (Role) models, Abstract interface models describing content and layout, Navigation models and Detailed (Concrete) Interaction Models, describing the reactive behaviour of actual interface components, especially their look and feel. The following section discusses an extension to the UML notation for specifying tasks, user populations in terms of roles and their relationships and the relation of tasks to domain models. Domain models per se are served sufficiently by standard UML.

The main motivation for designing a notation for task modelling is to support task based design. Wilson and Johnson (1996) discuss extensively the use of task models in design, focusing on the activities designers must engage in. These activities are summarised as they determine roughly the intended usage of task modelling notations.

- Obtain an understanding of user tasks as they are currently performed (*extant* task model).
- Re-design of the user activity should be done at the task level, resulting in the *envisioned* task model.

- Design the user interface by applying the following heuristics:

- The hierarchical decomposition of tasks to subtasks can be mapped to grouping and containment relationships between interactive objects.

- The temporal sequencing of tasks should be preserved: The designed system should allow users to perform tasks in the order that they currently do, although new tasks and new task sequences should be allowed; see Markopoulos et al. (1997) for a formal expression of task conformance.

- Actions and objects pertaining to the task model should be mapped to corresponding collection of actions and objects on the user interface. This concerns both the actual semantics, their representation on the screen but also, significantly the level of abstraction. (see Markopoulos 1998).

A notation suitable for modelling tasks in support of task based design should make salient precisely those elements of tasks identified above. These requirements shape the extension summarised in the following section.

4. AN ABSTRACT SYNTAX FOR MODELLING TASKS AND MORE

This section describes an abstract syntax for specifying task models. As far as possible it is consistent with the abstract syntax agreed at the TUPIS workshop. That discussion has not been published to date, so for completeness, figure 1 shows the resulting abstract syntax for modelling tasks. A simplification of this syntax is adopted hereby, to minimise the necessary 'vocabulary' for the designer, and the required extensions to the UML. Figure 2 illustrates this simplification. The differences of the two models are highlighted in the paragraphs below.

Tasks are associated with roles, and are defined simply as 'things people do'. Avoiding terms such as 'knowledge', 'behaviour' or 'activities', which are some of the concepts usually employed in more theoretically oriented works, see for example (Johnson 1992), helps keep the notation neutral to theories about tasks or user activity and to techniques for task analysis. No distinction is made concerning top-level tasks or bottom level subtasks, sometimes called task actions, e.g., by Johnson (1992).

Subtasks of the same task can be associated with logical and temporal relations, which are defined as associations between tasks. Task ordering can be represented with these relations, but can also be

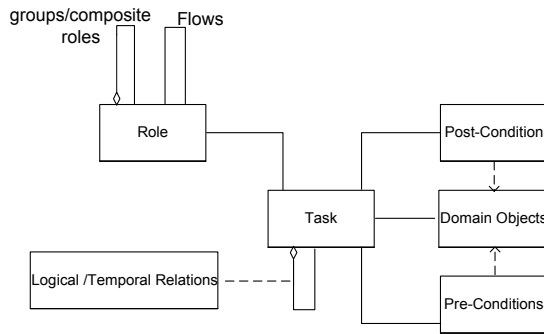


Fig. 2. A simplified abstract syntax for task modelling.

specified implicitly through pre-conditions and post-conditions. A further alternative is to associate tasks with activity diagrams, as was discussed in Markopoulos (2000). This can be particularly interesting when collaborative tasks are described, in which case a workflow is specified. Designers using this model can choose the most appropriate representation according to the problem at hand.

Pre- and post-conditions are described in terms of domain objects. In the model of figure 1, tasks are optionally associated with goals which are defined in terms of pre-conditions. Goals can be related to the roles associated with the task and they are described as post-conditions upon objects of the task domain. The association of a role to a goal can be characterised by a target-performance level. The simplified model of figure 2, is much more economical in modelling goals and task performance: In this model, tasks are assumed to describe goal related activity of the user, so goals do not need to be described separately. This may appear to be a narrower definition of a task, compared to the one of the TUPIS workshop, it seems more economical and consistent with task based design approaches.

Figure 1 illustrates some other concepts omitted in the simpler model. For example, classifiers were included to capture recurring patterns of task decomposition and of logical and temporal relations between tasks, in the fashion shown by Paternó (1999). This information captures similarity between tasks for the purposes of encouraging re-use and consistency between designs. An association class *Performs* models the relation between roles and tasks. This was intended to represent whether a task is performed cooperatively, interactively with the system, or by a single user. Performance of a task is associated with roles, rather than specific users.

In the simplified model adopted hereby, roles are considered as abstractions of responsibilities people assume in relation to some task and of interests they wish to ensure when they assume the role. Responsibilities and interests may be specified as attributes of a role, although in the discussion below

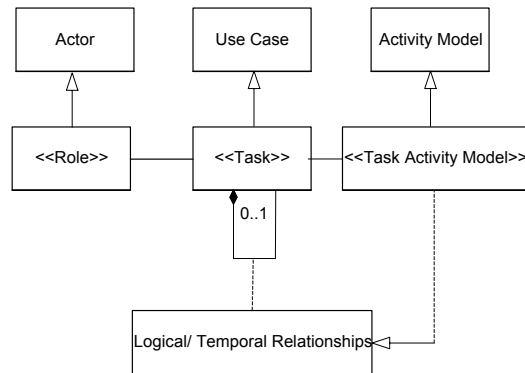


Fig. 3. Relation of tasks and task activity models to standard UML elements.

they are simply shown as notes (i.e. they are not part of the UML extension). Specific individuals are not modelled in figure 2. Roles may be associated with flows, indicating communications between different roles. Such communications may transfer information or material artefacts. This type of relationship helps represents models of work in the style of Contextual Design (Beyer et al. 1998), without detailing the temporal ordering of the flows as in activity diagrams of the UML. (An activity diagram modelling work-flow can then be seen to realise the flow associations discussed here adding temporal sequencing). Roles are related to each other via generalisation relationships and via aggregation relationships to represent groups, e.g., a purchase department, or composite roles, e.g., software engineers also acting as project managers.

5. REPRESENTATION IN THE UML

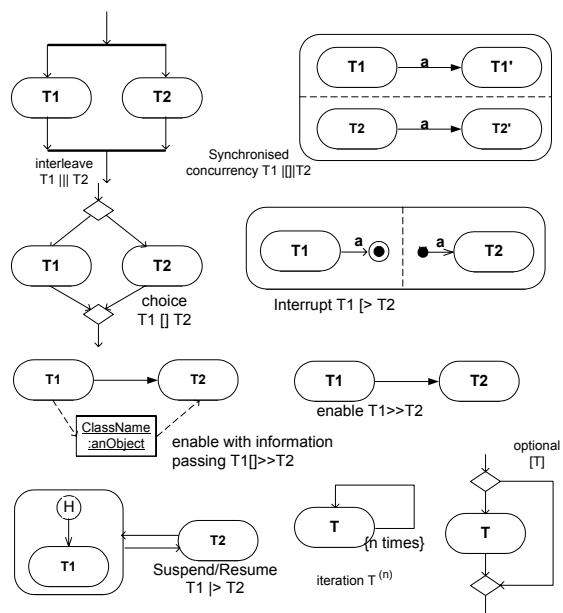


Fig. 4. Mapping of CTT operators (Paternó 1999) to activity diagrams.

This section describes how the abstract syntax of figure 2 is rendered in the UML. The relation to UML meta-model elements is shown in figure 3.

Tasks are modelled as stereotyped use cases, which are constrained to describe the goal-oriented activity of humans, abstracted through their roles. Tasks may be distinguished as those performed unobserved by the system, e.g., decisions by the user or tasks simply falling outside the scope of the system, but which are relevant to the human user – cf. Markopoulos (1998). Other authors, e.g., Paternò (1998) and Nunes (2000) distinguish tasks of the human, interactive tasks, and system tasks for the purposes of task allocation. Both approaches have their merits and can be supported by introducing relevant tagged-values for the specific class of tasks.

Roles are modelled as stereotyped actors or as aggregations of other roles, describing groups. Responsibilities and interests are written here simply as UML notes, in order to minimise the notational elements introduced. (See figure 6). A stereotyped association between roles can represent ‘flows’.

By modelling tasks as use case stereotypes task attributes may be simply written as attributes of these use cases. Alternative task decompositions for the same task (OR relationship between the task structures) can be represented with the generalisation relationship. The task-subtask relationship, where the task consists in the composition of its sub-tasks, can be expressed as stereotyped `<<includes>>` dependency between use cases. The `<<extends>>` dependency is not used for task diagrams.

Temporal relations between tasks, and properties such as optionality and atomicity (the task cannot be interrupted once it begins), can be represented as constraints. For example, constraints can directly be written using the CTT operators. The UML does not restrict the constraints that should be used, but staying with a standard set such as the CTT operators should encourage consistent specification styles.

As with all use cases, tasks can be associated with corresponding activity diagrams. For clarity, these are termed task-activity diagrams here, although as a notation they are no different to any UML activity diagram. The task activity model is simply an activity model that describes the behavioural aspects of a task. Its activity states are further specified as task activity models, that correspond to the subtasks of this task. There is potential redundancy between a task use case model and the task activity model. The task activity model is an intensional specification of the logical and temporal relationships between tasks specified (extensionally) in the task diagram. The correspondence of CTT operators used in the task diagrams and the activity diagrams of the UML is illustrated in figure 4. All CTT operators can be

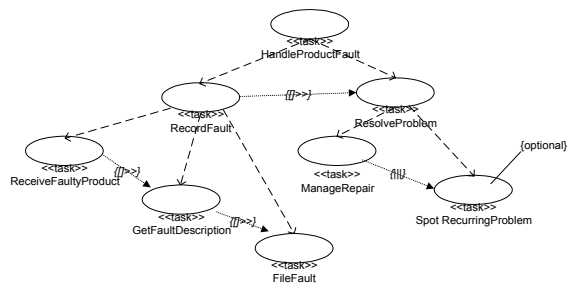


Fig 5 Specifying the task model using use cases simply written as constraints between classes. These constraints that are part of the task model, (the collection of tasks and role classes), are realised in task activity diagrams.

6. EXAMPLE

The example concerns an imaginary task of repairing faulty products, performed at a repair desk of a manufacturing company reseller. The task is a simplified and abstract version of a task model studied in the context of a case-study. The task model is shown in figure 5 using the stereotyped use cases. The related roles are shown on figure 6.

7. CONCLUSIONS

This paper has outlined an abstract syntax for modelling tasks, upon which an extension of the UML is based. The proposed extension gives rise to a series of stereotypes for standard modelling elements of the UML, but is consistent with the UML meta-model. The extension is minimal in that it consists only in three stereotypes, and avoids the introduction of icons and tagged values that are not strictly necessary. However, such elements may prove useful in practice, and may be incorporated in the notation in the future.

Describing tasks as stereotype use cases affords concise specifications of the task hierarchy, to match those of special purpose task modelling notations, such as the CTT notation (Paternò 1999). This circumvents the limitations to the readability of

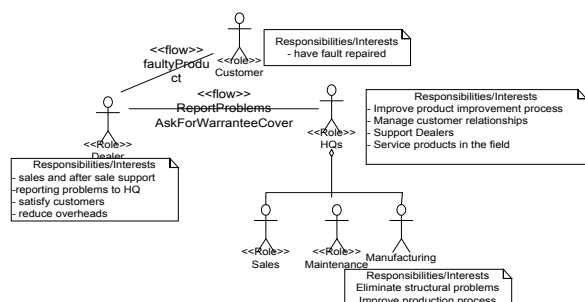


Fig. 6. Example of the role model showing responsibilities and flows between roles.

activity diagrams that were highlighted in (Markopoulos 2000). The notation introduced is consistent with widely differing approaches to task analysis, such as contextual task analysis (Beyer et al. 1998) and the Task Knowledge Structures Theory by Johnson (1992).

Tasks are generic constructs that can specify cooperation and workflow. They are associated with models of roles, which portray static pictures of user groups, their structure, the tasks they perform, the responsibilities they assume and the interests they protect. The notations introduced can be used to represent tasks as they are currently performed, or as they will be performed in an envisioned situation. Also, tasks can be system independent, idealised task descriptions or system specific interaction tasks. Focusing on tasks, rather than use cases, does not radically change the application scope of the UML, (work flows and user activities can be represented in standard UML) but changes its focus towards focusing on users and their tasks, as opposed to the system functionality. This has long been a tenet for the human computer interaction community, and it seems a natural adaptation of the UML to apply it in this field.

The current proposal has arisen out of an attempt to apply the UML to interaction design within the context of two case studies, one industrial and one small-scale research project. Its intended audience is multi-disciplinary teams, where interaction designers are not assumed to be software experts. Planned work is to test this notation in the field and to reshape it following the feedback of its target audience, developing the notation itself in a user centred manner.

8. ACKNOWLEDGEMENTS / DISCLAIMER

S.Kovacevic, F.Paternó, J.Cunha, N.Nunes, P. da Silva, M.v.Harmelen. and the other participants of the TUPIS 2000 workshop are acknowledged. The abstract syntax of figure 1 captures some of the ideas of the workshop discussions, but does not necessarily reflect faithfully the ideas of all participants.

9. REFERENCES

Anderson, D., (2000) TUPIS2000 - Notes on Interaction Spaces from UML 2000, Web: <http://http://www.uidesign.net/2000/conference/TUPISreport.html>.

Artim, J., et al, (1998) Incorporating Work, Process and Task Analysis Into Commercial and Industrial Object-Oriented System Development, SIGCHI Bulletin, 30(4).

Beyer, H. and Holtzblatt, K., (1998) Contextual Design: Defining Customer-Centered Systems. Morgan-Kaufmann.

Bomsdorf, B., and Szwillus, G., (1998) From Task to Dialogue: Task Based User Interface Design, SIGCHI Bulletin, 30(4).

Carroll, J. M. (1997) Scenario based design, In Helander, M.; Landauer, T. and Prabhu (Eds.) Handbook of Human Computer Interaction, Elsevier, 1997, 383-406.

Constantine, L.L. and Lockwood, L.A.D, (1999) Software for Use: A Practical Guide to the Models and Methods of Usage Centered design, Addison-Wesley.

Johnson, C., (1996) The Namur Principles: Criteria for the evaluation of user interface notations. In Bodart, F., and Vanderdonck, J., (Eds) DSV-IS '96.

Johnson, P., (1992) Human-Computer Interaction. Psychology, Task Analysis, and Software Engineering, McGraw Hill.

Markopoulos, P., Johnson, P. & Rowson J., (1997) Formal Aspects of Task Based Design., In Harrison, M.D., and Torres J.C., (Eds.), DSV-IS'97: Springer-Verlag, 209-224.

Markopoulos, P., (1998) Comparing non-deterministic models of tasks and devices. FAHCI'98, Sheffield-Hallam Univ., 5-6 September 1998.

Markopoulos, P. and Marijnissen, P., (2000) UML as a representation for Interaction Design. In Paris, C., Ozkan, N., Howard, S., and Lu., S. (Eds.) Proceedings OZCHI' 2000, 240-249.

Markopoulos, P., (2000) Supporting interaction design with the UML: Task Modelling, TUPIS 2000 workshop, York, October 2-3, 2000. Web: math.uma.pt/tupis00.

Mayhew, D.J., (1999) The Usability Engineering Lifecycle, A Handbook for User Interface Design, Morgan-Kaufmann.

Nunes, N., et al, (1999) Workshop on Interactive System Design and Object Models, in Moreira, A., (Eds.), ECOOP'99 Workshop Reader, Springer-Verlag LNCS, 1999.

Nunes, N.J., and Cunha, J.F., (2000). Wisdom: a software engineering method for small software development companies. IEEE Software, Sept./Oct. 2000, 113-119.

OMG (1999) OMG Unified Modeling Language Specification, version 1.3 (1999).

Paternó, F.,(1999) Model Based Design and Evaluation of Interactive Applications, Springer-Verlag, London.

Wilson S., and Johnson, P., (1996) Bridging the Generation Gap: From Work Tasks to User Interface Designs. In Vanderdonck, J., (Ed.) Proc. CADUI'96, Presses Universitaires de Namur, 77-94.