Collaborative Task Modeling: A First Prototype Integrated in HAMSTERS

Marius Koller[1(✉)], Cristian Bogdan[2], and Gerrit Meixner[1]

[1] UniTyLab, Heilbronn University, Heilbronn, Germany
{marius.koller,gerrit.meixner}@hs-heilbronn.de
[2] CSC, MID, KTH Royal Institute of Technology, Stockholm, Sweden
cristi@kth.se

Abstract. Task models are introduced in several use-cases in academia. They are usually created in collaboration between different people and disciplines. There exist many notations and associated graphical editors to create the models. However, these editors do not have integrated functions to support collaborative work. In this work, we propose the integration of collaborative functions in the HAMSTERS task modeling tool.

Keywords: Task modeling · Collaborative support · HAMSTERS

1 Introduction

The creation of task models is a complex undertaking that is usually performed by people from different professions and backgrounds. It is unlikely that just one person is working on task models in a given project. Therefore, task models will often be the result of some kind of collaboration. To support task authors and their daily work, digital support for the collaborative creation of task models is necessary. The needed digital support depends on the nature of task model creation process in each group.

In this demo we propose an integration of collaborative features in the HAMSTERS task modeling tool. We focus on the communication between the collaborators as well as on awareness between them. The proposed and prototyped functions are being implemented and will be evaluated in the future.

2 Task Models

Task Models are a well-known concept in research and academia. Task modeling has its foundations in the Hierarchical Task Analysis (HTA) that was defined in 1967 by Annett and Duncan [1]. HTA has the goal to analyze the user's tasks and structure it hierarchically. Many different task modeling notations were defined and are used in different, mostly academic, use cases, for different purposes. For example, the Useware Markup Language (useML) [7] aims to generate user interfaces that are based on the created task models. For this purpose, the in useML defined Usemodel is transformed

DOI: 10.1007/978-3-319-44902-9_24

to other models and at the end it is possible to generate the final user interface. The Groupware Task Analysis (GTA) aims to support the task-based development of collaborative applications [14] (we should however distinguish collaboration during task modeling from collaboration of end users modeled by collaboration-aware task models such as GTA). The Méthode Analytique de Description (MAD) is a semi-formal method to record user tasks based on interviews [11] that can be used during the requirements analysis phase. The ConcurTaskTrees (CTT) are and well-known approach [9] which has dominated research approaches lately. CTT aims to analyze and record the user's task and generate models from them. The Human-centered Assessment and Modeling to Support Task Engineering for Resilient Systems (HAMSTERS) [6] uses some concepts from CTT and aims to identify possible human errors and to prevent them.

Most of these notations have graphical representations and editors to create graphical task models. For instance, HAMSTERS (see Fig. 1) and CTT's editors CTTE [10] and Responsive CTT [2] use small icons for the representation of tasks. For the different type of tasks there are specific icons defined and used in the models. UseML's editor Udit [8] represents the items with boxes and differs them using a color-code. CTTE and Udit are stand-alone software whereas Responsive CTT is web-based.

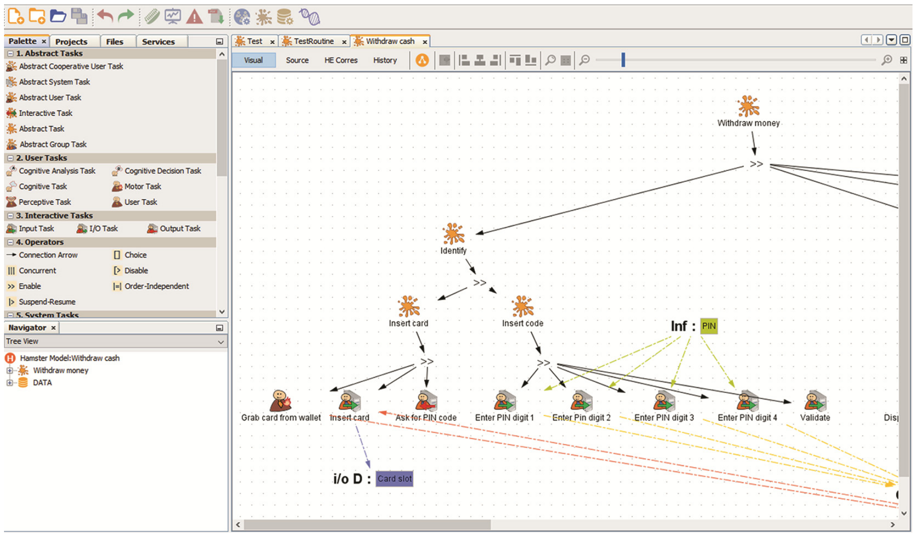


Fig. 1. The graphical editor HAMSTERS.

3 Computer Supported Cooperative Work

The field of Computer Supported Cooperative Work (CSCW) has been aiming to support technology-mediated collaboration for several decades.

Collaborative editors are an important area of CSCW research, and it is collaborative task model editors that we aim to create a system for collaborative editing. It needs to support the following features, according to [4]:

1. **Distribution**: the clients are not co-located but distributed and connected over the internet or an internal network.
2. **Sharing**: the clients should be able to share documents but be aware with whom they are sharing it.
3. **Autonomy**: a document has an owner that has the right manage it, e.g. grant access.
4. **Reliability**: the stored data has to be robust to connection losses and be accessible when a client crashes.
5. **Reconfiguration**: the clients should easily configure the system and its behavior.
6. **Concurrency**: the system has to allow editing at one time.
7. **Consistency**: the stored data has to be consistent even during the concurrent editing.
8. **Performance**: the responses should have a reasonable delay that allows a fluent working with the system.
9. **High-Quality User Interface**: the provided user interface should support the clients during their work.

Sutcliffe defines requirements for groupware, some of them are similar to the above [13]. He adds for example the "group's collective goal awareness" – that means the group should be aware of its goal, means why they are collaborate and that may include sub-goals. Also, possible conflicts have to be managed in a transparent and fair way. Sutcliffe's findings do not change the defined requirements but support and extend them.

One challenge in CSCW is the concept of "awareness" defined as "understanding the activity of others in the context of one's own activity" [3]. One central question is "what should the user be aware of?" [12].

4 Collaborative Support in Task Modeling Environments

To understand the needs for a collaborative support we need to know: who are the collaborators? What are their professions, background? We found in a small pilot-study that many different professions engage in task model authoring. They may not have a technical background and this needs to be taken into account in the collaborative graphical editor design. We identified three groups: Usability Experts, Software Developers/ Engineers and User Interface Designers. From literature and other projects, we know that in some cases the management is involved as well. In our own studies we found that the teams may be distributed and the interaction or collaboration –which means the implemented process - differs between companies.

We looked at four task modeling tools and evaluated their possibilities for cooperation support.

At first we identified features that afford some sort of collaboration. The most basic form for developers is the sharing of file with a Version Control System (VCS) like for example Git or Subversion (SVN). Other ways of sharing files, like simple "saving in the cloud" and sharing with others – at the same or different time. Also an important aspect for collaboration is the communication. There are different communication-channels that could be used: text-based chat, call or video-telephony. For the distributed creation of task models, it is useful if users could comment on tasks, branches of a tree model or on the whole model. Users may want to know how a model evolved during the creation. For that a history feature is useful where the users can see what was changed.

As shown in Table 1, HAMSTERS is the only tool that provides some aspects of collaborative support. HAMSTERS has an integrated version control (VCS) that allows the users to share their project. The history is closely related to the VCS. The view shows the underlying XML-file and shows the differences between the selected version and the current working-copy. There is no possibility to show the differences in the graphical representation.

Table 1. Analyzed tools.

Feature	Udit	CTTE	Responsive CTTE	HAMSTERS
Sharing	✗	?	?	✓
Chat	✗	✗	✗	✗
Comment	✗	✗	✗	✗
History	✗	✗	✗	✓

Several incipient collaboration features are supported by other modeling tools as well. In CTTE it is possible to upload the file to a public or private repository at HIIS. Responsive CTTE could be able to share the models. Since it is web-based and the models are saved on servers, it could be possible to share the files. Currently this function is not implemented. Udit has no functionality for collaboration integrated.

Hili et al. [5] introduce with FlexiLab a tool that offers collaborative functions amongst others. They focus in their work on real-time communication and sharing of models. The sharing enables the users to work on different parts of a model and share it.

5 Mock-Ups of Collaborative Functions

Since we have some functions that are already integrated in HAMSTERS, we decided to continue working with it adding collaborative support. In order to generate ideas on how a collaborative support could look like, and in order to be able to analyze and judge these design ideas, we developed prototypes for the different collaborative functions. The ideas are being implemented as online prototypes, and prepared for demo. Since HAMSTERS is based on NetBeans™, we intend to stick on the NetBeans tab-based navigation concept.

The first function that we considered is the communication between users. We thus integrated a tab that shows a chat tool We decided to introduce a tab with the same behavior like the other tabs (e.g. tabs could be arranged according to the user's preferences). An important point to keep in mind is that the chat has to be visible for every user, including what was said before he or she joined the talk. The mockup is depicted in Fig. 2.

As a second major function we identified commenting. We believe that for the collaborative creation it is important to tell the other editors how you are thinking while designing a task model. The comments are anchored on specific parts of a task model. Therefore, we intend to introduce comments that can be attached to a specific task or a

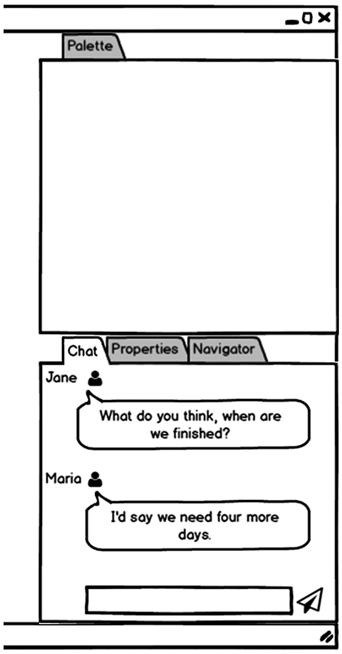

Palette
Chat
Properties
Navigator
Jane
What do you think, when are we finished?
Maria
I'd say we need four more days.

Fig. 2. The integrated chat.

branch of the model. The possibility to add a comment will be shown at the object of interest. Depending on the kind of task. e.g. Abstract Task or User Task, the comment will refer to a whole branch or only to this specific task. The first comment in Fig. 3 includes the possibility to comment on the whole branch while the second comment or conversation is regarding this particular task. It will be possible to start discussions within the comments.

The current HAMSTERS implementation of a history is readable only by experts, e.g. Software Engineers, as it is based on version control specific to software. Users from other disciplines that use the application and its models are currently not able to understand the changes without an expert explanation. To support such users, we will introduce a graphical representation of task model history. One design alternative is to display history as grayed-out objects.

The sharing of the model with real-time editing may need changes in the user account structure of HAMSTERS. For example, we need to display the currently active users and which users have access to the model (see Fig. 4). To achieve that, we have to provide a login for the users. Again, to be consistent with the existing UI design of HAMSTERS we will introduce a tab that contains the currently active users and the users that have access to the model.

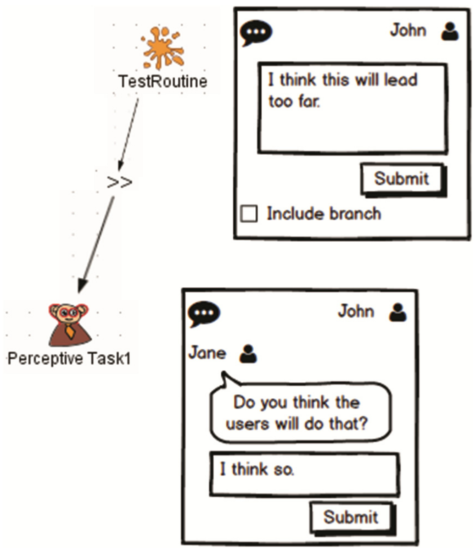
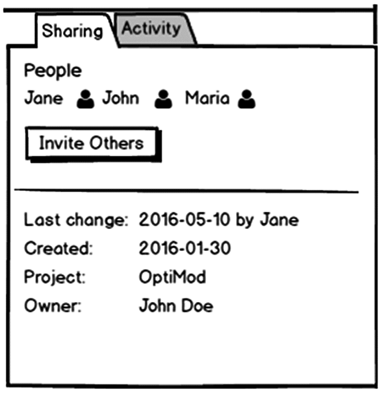


Fig. 3. Two comments on tasks.



Last change:	2016-05-10 by Jane
Created:	2016-01-30
Project:	OptiMod
Owner:	John Doe

Fig. 4. Sharing the model.

After implementing such basic collaborative support for task editing, we will look into more advanced support, derived from field studies of task modeling that we are conducting. For example, awareness [4, 11] will be initially supported only by our history feature, which is a good beginning is history is shown permanently in a tab, and maybe recent changes are promoted in the tab title. More awareness support can be added to emphasize important task model changes made by collaborators.

6 Conclusions

We regard task modeling as complex interdisciplinary collaborative work. Different disciplines and profession combine their knowledge and expertise in these artifacts. However, in the current tools for task modeling we found only little collaborative support. Only HAMSTERS includes with SVN and Git which are well-known collaborative tools for software developers. We propose new features that introduce collaborative support in HAMSTERS. We believe that especially the possibility to comment on tasks or branches will support distributed teams that work asynchronously. The other features will ease the creation of task models in groups, too.

We are in the process of implementing the collaborative task modeling features and we aim to later evaluate these features.

References

1. Annett, J., Duncan, K.D.: Task analysis and training design. Occup. Psychol. **41**, 211–221 (1967)
2. Anzalone, D., et al.: Responsive task modelling. In: Proceedings of the 7th ACM SIGCHI Symposium on Engineering Interactive Computing Systems, pp. 126–131. ACM, New York (2015)
3. Dourish, P., Bellotti, V.: Awareness and coordination in shared workspaces. In: Proceedings of the 1992 ACM Conference on Computer-supported Cooperative Work, pp. 107–114 ACM, New York (1992)
4. Greif, I., et al.: A case study of CES: a distributed collaborative editing system implemented in Argus. IEEE Trans. Softw. Eng. **18**(9), 827–839 (1992)
5. Hili, N., et al.: Innovative key features for mastering model complexity: flexilab, a multimodel editor illustrated on task modeling. In: Proceedings of the 7th ACM SIGCHI Symposium on Engineering Interactive Computing Systems, pp. 234–237 ACM, New York (2015)
6. Martinie, C., et al.: A generic tool-supported framework for coupling task models and interactive applications. In: Proceedings of the 7th ACM SIGCHI Symposium on Engineering Interactive Computing Systems, pp. 244–253 ACM, New York (2015)
7. Meixner, G., Seissler, M., Breiner, K.: Model-driven useware engineering. In: Hussmann, H., Meixner, G., Zuehlke, D. (eds.) Model-Driven Development of Advanced User Interfaces. SCI, vol. 340, pp. 1–26. Springer, Heidelberg (2011)
8. Meixner, G., et al.: Udit–a graphical editor for task models. In: Proceedings of the 4th International Workshop on Model-Driven Development of Advanced User Interfaces (MDDAUI), Sanibel Island, USA, CEUR Workshop Proceedings. Citeseer (2009)
9. Paternò, F.: ConcurTaskTrees: an engineered notation for task models. In: The Handbook of Task Analysis for HCI, pp. 483–503 (2003)
10. Paternò, F. et al.: CTTE: an environment for analysis and development of task models of cooperative applications. In: CHI 2001 Extended Abstracts on Human Factors in Computing Systems, pp. 21–22 ACM, New York (2001)
11. Rodriguez, F.G., Scapin, D.L.: Editing MAD* task descriptions for specifying user interfaces, at both semantic and presentation levels. In: Harrison, M.D., Torres, J.C. (eds.) Design, Specification and Verification of Interactive Systems 1997, pp. 193–208. Springer, Vienna (1997)

12. Schmidt, K.: The problem with ‘awareness’: introductory remarks on ‘awareness in CSCW’. Comput. Support. Coop. Work CSCW **11**(3–4), 285–298 (2002)
13. Sutcliffe, A.: Applying small group theory to analysis and design of CSCW systems. In: Proceedings of the 2005 Workshop on Human and Social Factors of Software Engineering, pp. 1–6. ACM, New York (2005)
14. Van Der Veer, G.C., et al.: GTA: groupware task analysis—modeling complexity. Acta Psychol. (Amst.) **91**(3), 297–322 (1996)