ProtoTask, New Task Model Simulator

Thomas Lachaume, Patrick Girard, Laurent Guittet, Allan Fousse

▶ **To cite this version:**

Thomas Lachaume, Patrick Girard, Laurent Guittet, Allan Fousse. ProtoTask, New Task Model Simulator. 4th International Conference on Human-Centered Software Engineering (HCSE), Oct 2012, Toulouse, France. pp.323-330, 10.1007/978-3-642-34347-6_24 . hal-01556834

HAL Id: hal-01556834

https://inria.hal.science/hal-01556834v1

Submitted on 5 Jul 2017

HAL is a multi-disciplinary open access archive for the deposit and dissemination of scientific research documents, whether they are published or not. The documents may come from teaching and research institutions in France or abroad, or from public or private research centers.

L'archive ouverte pluridisciplinaire **HAL**, est destinée au dépôt et à la diffusion de documents scientifiques de niveau recherche, publiés ou non, émanant des établissements d'enseignement et de recherche français ou étrangers, des laboratoires publics ou privés.

Distributed under a Creative Commons Attribution 4.0 International License

ProtoTask, new Task Model Simulator

Lachaume Thomas, Girard Patrick, Guittet Laurent and Fousse Allan

LIAS / ISAE-ENSMA, University of Poitiers
86961 Futuroscope Chasseneuil cedex, France
{Thomas.Lachaume , Patrick.Girard, Laurent.Guittet, Allan.Fousse}@ensma.fr

Abstract. One major objective of task modeling is to improve communication between design stakeholders. Current task model simulators, which require their users to understand task model notations, and provide for inappropriate information, are not really suitable for this topic. We designed ProtoTask, which allows the user to experiment task models by the way of building scenarios, without understanding task model notations. This tool presents new mechanisms that aim at facilitating the understanding of task models for all users.

Keywords: Task Model, Simulation, Human-Computer Interaction, User Experience

1 Introduction

The interest for task modeling techniques and tools is increasing in the Human Computer Interaction (HCI) community, thanks to their solid theoretical foundations, which allow, for example, the building of automatic transformation approaches. Some task model characteristics allow the models to be executed through tools named simulators. These tools simulate the activity described by the task models according to their semantics, and provide scenarios of execution.

A new area for these tools consists in exploring how they can help in facilitating exchanges between users and designers that aim at expressing the user needs. Nevertheless, simulators usually share the same design, and require task-modeling knowledge to be used and understood by final users [1]; indeed, in these tools, one has to look at the task tree to understand the simulation. This tree has edges with semantic notation. We present here a new tool, ProtoTask, which has been designed specifically to address this new area. It does not need to look at and understand the task model notation, and focuses on information and tasks that specifically address task model validation by end-users.

After a short presentation of the task-modeling field, we present an analysis of existing simulation tools, before detailing and contrasting the ProtoTask approach, which is illustrated on an example.

2 Task models and Simulation

Task models are based on the goal/sub-goal decomposition of Norman's theory [2]. Mostly, with task tree, they provide hierarchical representations for the modeling of activity. As any trees, task models have leaves and nodes. Leaves are called elementary tasks, which represent the concrete tasks that can be made during the activity. On the opposite, nodes stand for compound tasks, which are decomposed in two or more subtasks, and are not really included in the activity. They are structuring tasks, which represent the goal/sub-goal decomposition. Depending on the different models, tasks get some attributes such as goal, priority, type, and frequency.

Even if tasks have static attributes for a better understanding, activity is dynamic and includes a lot of indeterminate situations. This dynamics is expressed by scheduling and optional operators, and preconditions[3]. Scheduling operators depend on the model. CTT[4], and HAMSTERS[5] use LOTOS operators. K-MAD[6] and AMBOSS[7] use different operators, which are equivalent to a subset of the CTT's operators. Operators can be expressed between tasks at a same level of decomposition (CTTE) or linked to task decomposition, and common to all sub-tasks (AMBOSS, HAMSTERS, K-MAD, VTMB[8]). Expressiveness of task models can be enhanced by expressions, which manipulate objects.

One of the aims of task models is to improve communication between design stakeholders, end users, HCI experts, and software developers[9]. As for every notation, this point requires all participants to figure out the dynamic behavior of the model, which is the most important topic. In order to help in this task, simulation tools have been designed. They allow concrete examples of scenarios to be run on the model, giving a good understanding of the behavior of the model. Simulators are close to debuggers; they offer a step-by-step execution, and allow users to confirm, deny or simply discuss about the scenario. They are usually part of task design environments, such as CTTE, K-MADe, AMBOSS, HAMSTERS, or VTMB.

The major drawback of simulation tools is that they are designed as tools for task specialists. All the information provided to users is related to the task model. It is necessary for simulator users to lean on the task model itself, and it is not possible to understand the context of the simulated tasks without looking at the task model. Moreover, richer the model, more complex is the layout of the simulator. This is good for designers, who master the notation, and are able to deal with all the shown information. This is not good for end-users who need to learn the task tree notation to understand the simulation and to deal with inappropriate information. A previous study shown that this prerequisite is an obstacle to using task model simulators as a medium of communication between end-users and HCI specialists[1].

Our approach consists in exploring the idea that end users do not need to really understand the task model notation to validate the task analysis. The goal of such a validation can be summarized by two points: (1) the user must be able to run the scenario he/she has in mind, and (2) for each step of the scenario, all different options are legal. For that purpose, we created a tool, ProtoTask, specially designed to promote exchanges between users and designers, which allows building scenarios without showing the task tree. ProtoTask is built on top of, but not linked to, the K-MAD notation.

3 Simulation tools analysis

In this section, we describe the usage of simulators, by giving a concrete example. Even if the K-MADe simulator is closer to our solution, we chose to mainly illustrate the simulators by CTTE, the environment for the CTT notation, for two reasons: (1) the K-MADe simulator is the most complex simulator that can be found in task modeling tools, and (2) CTTE is the most widely used tool. Nevertheless, we examine different tools, and try to generalize our study to all approaches.

3.1 A short description of the CTTE simulator

The main goal of simulators is to allow the execution of the model, i.e. running one task after another at the right time, according to the model temporal semantics. The most important visible area within the CTTE simulator is the task tree (cf. zone 1 Figure 1). Once the simulation is launched, the user can see, surrounded by green squares, the *enabled task sets*[3], i.e. the tasks that can be done at present time according to the model semantics. Enabled tasks can only be leaves, i.e. only the most accurate description of the model, and the user must look at the tree to understand the context (e.g. the node task and the operator) and how it is a part of the decomposition. The user can use the enabled tasks (zone 2 Figure 1) panel to search for the task to be run (a click on the enabled task centers the tree on the task in the display).

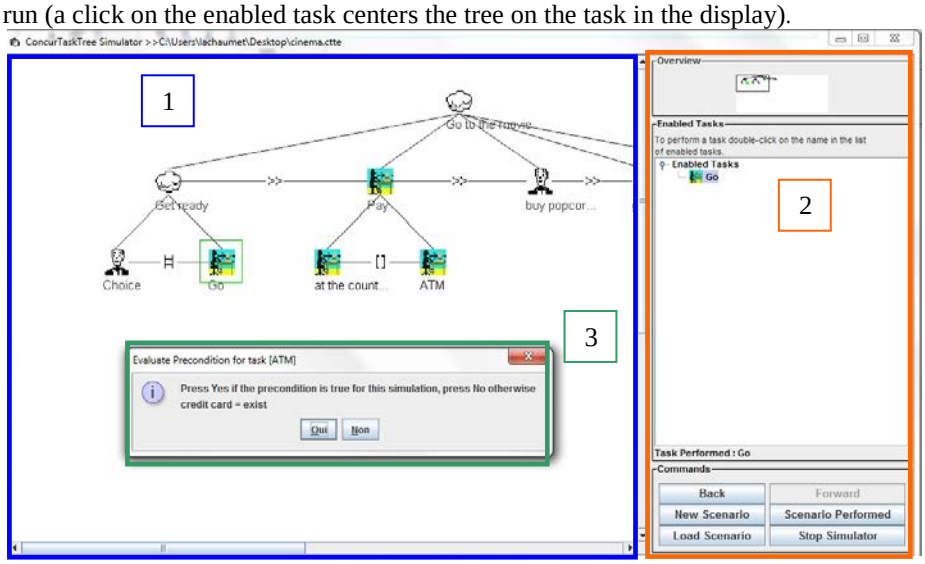

Fig. 1. CTTE simulation tool

In this simulator, the user can easily access the type and the identifier of the task displayed in the tree, but other attributes like description or frequency are not accessible. The operators between tasks are visible too; nevertheless, remembering and understanding these operators needs some apprenticeship time. The precondition system is based on task objects. Conditions are comparisons (equals, differs, contains, ends, starts) between objects or between objects and values are fixed before the

simulation. During the simulation, preconditions are displayed in pop-up windows (zone 3 figure 1), which ask the user for the precondition value. The user can answer *yes* or *no*; if the condition is true, the task is added to the enabled tasks sets, if not the task cannot be chosen. The user cannot change his/her choice without starting a new simulation or loading a previous scenario. If the parent task is repetitive for example the question is asked again.

3.2 Simulator analysis

Using simulators as a basis of discussion between end users and task analysts is hard, because of several simulator design choices that have bad consequences. This section results from preliminary studies we made to experiment the usage of simulators.

Firstly, simulators do not provide for appropriate task information during simulation. Tasks are only known by their name in the model, which stands for a short description. Some simulators hide much information (CTTE) or provide too much information (K-MADe), which makes end-users lost. We assume that some information from tasks, such as a more detailed description of the task, is very important for end-users during validation.

Secondly, scenarios are not directly visible in the simulator, such as for CTTE or VTMB. The user cannot refer to them during the simulation to be sure he/she is right. Other tools show the scenario during its building (K-MADe, AMBOSS), but it is reduced to a sequence of elementary tasks (the ones the user run). This list hides the goal/sub-goal underlying decomposition, and becomes very hard to understand without looking at the tree. Following Go and Carroll[10], we think that the context is important for scenarios. Figure 2 below shows the difference between a scenario with only elementary tasks (as in CTTE or K-MADe), and a scenario that displays the task/subtask hierarchy (ProtoTask).

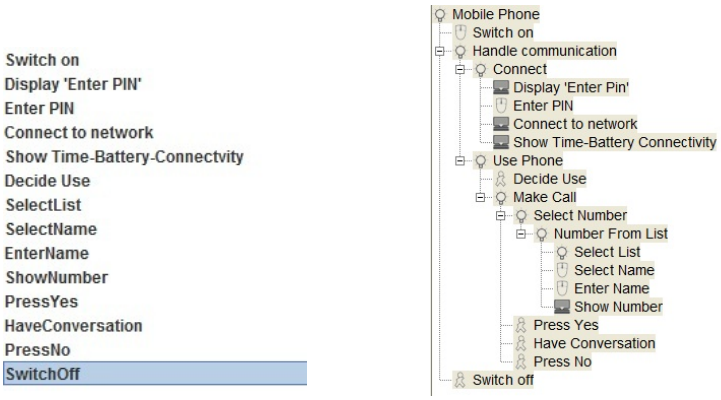

Fig. 2. A simple scenario (left), and the same structured scenario (right)

Thirdly, carrying out a simulation with all simulators requires the user to look at the tree. While it is not a problem for task modelers, other stakeholders need explanations or need learning about notations. More important, the communication is

being focused on the task model notation, instead of staying focused on the task model itself, the only thing stakeholders are really interested in.

Lastly, the support for condition execution is not appropriate to end-users. In CTTE, they can give at one time a value (*yes* or *no*) for the condition, which is used during the simulation. If the user needs to use it again (because it is part of a repetitive task, for example), he/she cannot have a remind of its previous value. In K-MADe, expressions lean completely on objects, which leads to very complex manipulations.

4 ProtoTask

In order to switch the primary focus on communication about the activity itself, we designed ProtoTask, a new simulator for task models. Its interface does not display the task tree and allows building scenarios step by step, with a top-down approach (i.e. nodes need to be started and finished). We chose to hide much information such as frequency and duration, because our first experiences show that, in most cases, they are not relevant in the early stage of the software life cycle. On the opposite, we emphasized the full description of the task, which is clearly visible, and concentrated on information that is important for end-users.

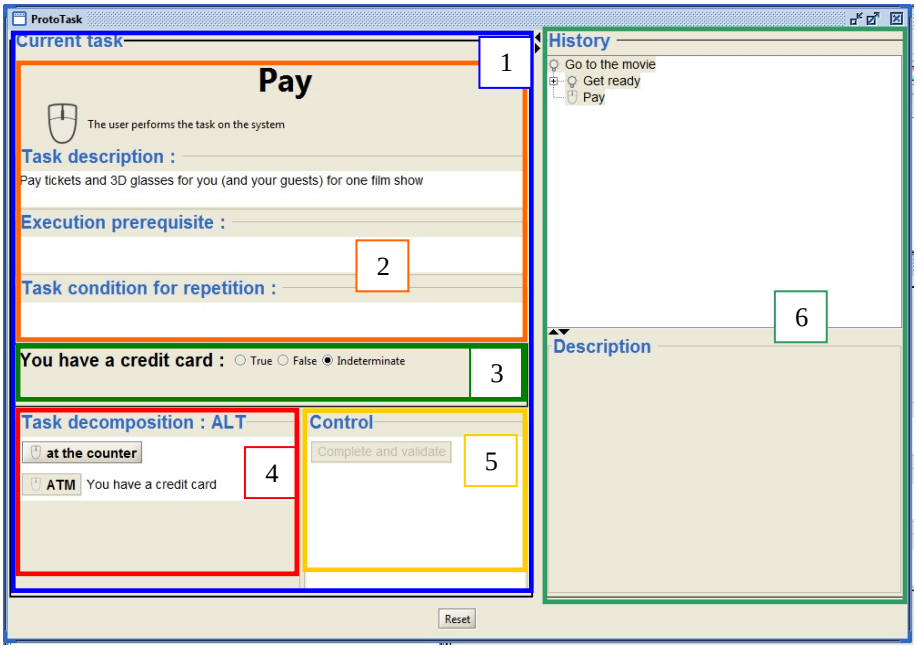

Fig. 3. ProtoTask

The main panel (zone 1 figure 3) represents the current task. The panel is divided into four areas: the task description (zone 2), the condition area (zone 3), the task decomposition area (zone 4), and the control area (zone 5). A second panel (zone 6) shows the scenario being built, under the name of “History”.

In Figure 3, the example task model is centered on “going to see a movie”. The “History” part shows a tree, with no scheduling operators. It represents the current scenario. Nevertheless, the structure of the tree enhances the parent/child relationship. It displays only started tasks. One can see the main task, “Go to the movie” (in progress), a “Get ready” task, which has been completed (the + symbol at the left indicates that this task is compound), and the current opened task, “Pay”. By providing a correct and a hierarchical sequence of tasks, this history can be used as a scenario for designers, but this history is also important for the user to remind the context. By default, only the path of the last task is expanded.

The first panel is completely devoted to the current task. The description area, below the name of the task, refers to the extended description of the task, which allows a good understanding of what the task does. This description can be changed directly in ProtoTask, and saved in the model. Below, the precondition of the current task is displayed, and then, the iteration expression if it exists.

A black border delimits the condition area, which contains both the iteration of the current task (not in the figure), and the precondition of the subtasks that can be started. Conditions are sentences or groups of sentences including OR, AND, and NOT operators. For each task, in the context of the current scenario, the user must state whether a condition is true, false, or indeterminate (default value). Depending on the state of each sentence, buttons in the task decomposition panel (4) and in the control panel (5) are enabled or not, which allows the user to understand the consequences of his/her choices for the reminder of the scenario.

Buttons from panels 4 and 5 allow the user to reach the next step of the simulation. According to his/her choice, and depending on the scheduling operator, he/she can start a subtask (panel 4), finish or repeat the current task (panel 5). When a task is elementary, or when it has no mandatory subtask, the user needs to validate (i.e. no error) and complete the task. No other simulation tool provides this function. It appears very important to confirm, in the end-user mind, the fact that the current task is really correct and completed.

Condition sentences keep their state if the task is repeated and if they are used in other tasks. Their value determines the possible options for the user. For example, if two sub-tasks, T1 and T2, can be launch according to the scheduling operator, and T1 have for precondition the sentence S, with a true result, and T2 have the opposite precondition (same sentence S, with a false result), only one sentence is displayed, and if S is true, the user can only launch T1, if S is false, he/she can only launch T2, and if S is indeterminate none of them are accessible. Figure 4 and 5 illustrate this management in the context of our example. The current task is “Pay”. Two modalities are available: paying at the counter (with cash), or paying at the automatic machine (with credit card). To make his/her scenario, the user needs to determine if he has a credit card (which allows or not to pay at the automatic machine) or not (he/she can only pay by cash at the counter). Indeterminate is the default value, and does not allow using the result of the condition; in this case, it is similar to a false value. In our example (figure 4), the user decided he/she owns a credit card. The two options (paying at the counter or by card) are always available, which is correct.

Fig. 4. The context or the "Pay" task, when the user owns a credit card

In fact, the user decided to pay at the counter, choosing the "at the counter" subtask. This is an elementary task, which must be completed and validated explicitly. The next point is illustrated by figure 5: the current task is "Pay" again (it is not explicitly completed). Because its decomposition is "ALT" (a choice, one sub-task among all) the two subtasks are disabled. The "at the counter" has been struck, so, all subtasks are now disabled. The user must complete and validate the task, or decide that the model is wrong, because something else should be done, for example.

Fig. 5. Explicitly finishing the "Pay" task

When the user clicks on the "Complete and validate" button, the "Pay" task is completed and ProtoTask comes back to the "Go to the movie" task to pursue it with respect to the task model.

ProtoTask differs from other simulation tools into several important aspects.

- Firstly, we chose to highlight the task description. This description is informal and allows writing lot of information such as reference documents, procedures, stories, etc. It can be edited jointly with the end-user during the simulation.
- Secondly, users construct and see their scenario when they perform the simulation, in a structured (tree) way, conforming to the task/subtask decomposition of the model. This feedback helps the user to ensure he/she is right, and can help him/her when he/she is lost, after a discussion or a break for example.
- Thirdly, users do not need to look at and learn about the task tree notation. As early results have shown [11], the user can understand the context better by using a top-down approach and he/she can use the history when he/she is lost.
- Lastly, condition systems are new too, the sentence format allowing to be focused on the task and not on the algorithm; the task modeler can, for example, refer to a document where all conditions are written ("Conditions in document HC53 are verified").

5 Conclusion and future works

In this paper, we present a new simulation tool for task models, ProtoTask, which does not require understanding task model notations. ProtoTask is built on top of the K-MAD method, but can be easily adapted to other methods, such as CTT for example. We are currently comparing ProtoTask to other simulation approaches for understanding and validating task models. First result with computer designers [12] showed that for understanding a new model, ProtoTask gets best results than simulators such as CTTE or K-MADe. We need to enhance this study with end-users.

ProtoTask can be improved in several ways. At present time, all features of task models have not been included and we have to study their interest for final users. Multi-actor and parallelism need to be studied cautiously.

K-MADe uses objects for expressing conditions[13]. Instead, Prototask abstracts the conditions. We need to study how we can insert full conditions for later stage of the life cycle, and how the understanding and the validation are impacted.

6 References

1. Caffiau, S., Guittet, L., Scapin, D.L., Sanou, L.: Utiliser les outils de simulation des modèles de tâches pour la validation des besoins utilisateur : une revue des problèmes. In: ERGO'IA. pp. 257-258. ESTIA, Biarritz, France (2008).
2. Norman, D.A., Draper, S.W.: User Centered System Design. Lawrence Erlbaum Associates (1986).
3. Paternò, F.: Model-Based Design and Evaluation of Interactive Applications. Springer (2000).
4. CTTE: http://giove.cnuce.cnr.it/ctte.html.
5. Hamsters: http://www.irit.fr/recherches/ICS/softwares/hamsters/index.html.
6. K-MADe: http://lisi-forge.ensma.fr/forge/projects/kmad.
7. Giese, M., Mistrzyk, T., Pfau, A., Szwillus, G., Detten, M. von: AMBOSS: A Task Modeling Approach for Safety-Critical Systems. HCSE 2008 and TAMODIA 2008. pp. 98-109. Springer (LNCS 5247), Pisa, Italy (2008).
8. Biere, M., Bomsdorf, B., Szwillus, G.: The Visual Task Model Builder. In: Vanderdonkt, J. and Puerta, A. (eds.) Third Conference on Computer-Aided Design of User Interfaces (CADUI'99). pp. 245-256. Kluwer Academic Publishers, Louvain-la-neuve, Belgique (1999).
9. Diaper, D.: Understanding Task Analysis for Human-Computer Interaction. In: Diaper, D. and Stanton, N. (eds.) The Handbook of Task Analysis for Human-Computer Interaction. pp. 5-48. Lawrence Erlbaum Associates, Inc., Mahwah (2004).
10. Go, K., Carroll, J.M.: Scenario-Based Task Analysis. In: Diaper, D. and Stanton, N.A. (eds.) The Handbook of Task Analysis for Human-Computer Interaction. pp. 117-134. Lawrence Erlbaum Associates (2004).
11. Lachaume, T., Girard, P., Guittet, L., Fousse, A.: Prototypage basé sur les modèles de tâches : une étude pilote. IHM'2011. pp. 2-5. , Sophia-Antipolis, France (2011).
12. Lachaume, T., Caffiau, S., Girard, P., Fousse, A., Guittet, L.: Comparaison de différentes approches de simulation dans les modèles de tâches. ERGO-IHM 2012. ACM Press (in press) (2012).
13. Caffiau, S., Scapin, D.L., Girard, P., Baron, M., Jambon, F.: Increasing the expressive power of task analysis: systematic comparison and empirical assessment of tool-supported task models. Interacting with Computers. 22, pp. 569-593 (2010).