The Usability of Task Modeling Tools

Markel Vigo
School of Computer Science
University of Manchester
Manchester, United Kingdom
markel.vigo@manchester.ac.uk

Carmen Santoro
HIIS Laboratory
CNR-ISTI
Pisa, Italy
carmen.santoro@isti.cnr.it

Fabio Paternò
HIIS Laboratory
CNR-ISTI
Pisa, Italy
fabio.paterno@isti.cnr.it

***Abstract*—User interface modeling tools can play a useful role for engineering interactive systems. However, little is known about how these tools are used, the strategies and workarounds employed, and whether usability problems are encountered. To start answering this question, we run a user study with software engineers carrying out task modeling activities. The analysis of the data generated by an instrumented CTTE and an eye-tracker uncovers five workflows that illustrate how the task model hierarchy is populated and edited, operators are added to tasks, and when model consistency is checked. These findings inform design implications for task modeling tools and formulate hypotheses for future work.**

***Keywords*—Log mining, eye tracking, usability, CTTE.**

I. Introduction

Task analysis enables to conceive and understand what users want to achieve through the use of software artifacts. It is consequently a fundamental activity to design user interfaces. Formalizing the outcomes of tasks analysis contribute to discussions amongst the stakeholders who are involved in the design of such artifacts because it documents the outcomes of the analysis in a clearer manner than just informally reporting them. This formalization is often implemented through task models, which provide designers with means of representing and manipulating a formal abstraction of the activities that should be performed. This is particularly useful e.g. for designing interactive safety-critical applications where the design needs to be carefully specified in order to prevent user generated errors that can have catastrophic effects.

The notations for representing task models describe the possible activities in a hierarchical manner so that more general tasks are decomposed into more specific ones that provide more precise indications about how to accomplish the tasks. Task modeling notations such as CTT [1], GTA [2] and MAD [3] explicitly indicate the temporal relationships amongst the tasks as a way to describe users' behavior. These notations establish relationships such as whether tasks have to be performed sequentially, concurrently or whether a given task can interrupt another one. There is also the possibility to add conditions that should happen before and after task completion. Existing tools such as CTTE [4], CogTool [5] and HAMSTERS [6] alleviate the complexity of task modeling notations and support designers in building sophisticated task models.

While task modeling does not have a widespread uptake, it has a user base of user interface specialists developing complex interactive systems. For example, the modeling environment studied in this paper, the ConcurTaskTrees Environment (CTTE), has been downloaded +26K times and has a user base of +10K registered users as of April 2017. It is both used in academia, government and industry, especially by companies with an interest in ERP and safety-critical systems (e.g. air traffic control systems). Unfortunately, we know very little about how effective task modeling tools are in supporting designers to build models and we know even less about their usability. In one of the few studies where task modeling environments were evaluated [7], the findings were within the scope of the expressivity of the notation, its syntactic validity and verification of resulting models. While this is useful to advance on task modeling notations, the lack of empirical studies on the use of task modeling tools prevents the community from moving forward. This is critical as novel applications of modeling tools are emerging in fields such as the Internet of Things and ubiquitous computing [8].

The lack of previous work and underlying theory led us to run a data-driven exploratory study with no hypotheses where findings would emerge from a variety of sources including user interface events, eye-tracking and self-reported data. Consequently, we contribute to theory and formulate hypotheses through:

- The identification of the activities designers carry out when building task models.
- Design recommendations for task modeling tools.

II. The Study

The study was run on a Windows7 PC with the Tobii X2-60 eye-tracker installed on a 21.5" monitor. A post-study questionnaire collected demographic information, the perceived complexity of the tasks and the participants' familiarity with the modeling notation and the environment.

A. Apparatus

1) Instrumented CTTE: We modified CTTE in order to log user interface events whereby each logged line contains a timestamp, the name of the area where the event was triggered, the event and the object of the event. The log excerpt below illustrates a real sequence from this study where (1) a task of the category 'Interaction' is added to the model, (2) this task is selected, (3) the 'Task properties' modal dialog is opened and (4) the default task name given by CTTE (i.e. 'Task_10') is replaced by 'Open messages'.

978-1-5386-0443-4/17/$31.00 © 2017 IEEE

```
1: 71345, Task palette, Add_task, Interaction
2: 74982, Central area, Select_task, Task_10
3: 75261, Central area, Open_task_properties, Task_10
4: 87735, Task properties, Set_identifier, Open messages
```

User interactions can be described in the logs by using different levels of abstraction, which can range from input device events (e.g. key pressed) to goal related (e.g. 'submit a form'). Following the framework by Hilbert and Redmiles [9] we logged 40 different types of events that belong to the categories of user interface events and abstract interaction level – the latter are semantically more meaningful than the former:

- Add tasks to a model. Tasks can be of the following categories according to CTT: abstraction, application, interaction and user.
- Add operators to tasks. There are 12 operators with their own semantics including choice, option and iteration.
- Set the identifier and category to tasks.
- Set the insertion mode for the next task to be added, which can be set as a parent task, as a child task or as a (left/right) sibling task of the currently selected task.
- Edit operations with tasks including copy subtree, cut subtree, and cut, copy, paste, drag and delete a task.
- Environment commands such as open a file, save, undo, redo, maximize the window, etc.

2) Eye-tracker: Eye tracking data has long been used to identify usability problems. Specifically, the number of fixations is an indicator of search problems and the duration of fixations signals cognitive load [10]. We, therefore, collected these values along with their timestamp and area of interest where fixations occurred. The areas of interest (henceforth AOI) are used to compute aggregates of fixation data in particular areas. In the case of CTTE, these areas are clearly demarcated as shown by Figure 1.

- The 'Drawing area' is the container of the task hierarchy, their relationships and their operators.
- The 'Left menu' contains the task palette to establish the category of task, and the operator palette to set the operator between the current task and its rightmost sibling.
- The 'Top menu' contains the main menu bar, a horizontal palette with edition commands and a button to set the insertion mode of the next task to be added (as a parent, sibling or child of the currently selected task).
- The 'Overview' shows the structure of the hierarchy in the drawing area.

Figure 1 also shows the dialogs that were defined as AOIs: 'Dialog - 1' is the drop-down menu that unfolds when clicking on the main menu bar, 'Dialog - 2' opens up on mouse right click to perform editing operations (cut, paste, delete) and 'Dialog - 3' is the 'Task properties' dialog, where the identifier and the category of the current task can be set and modified. When the simulator is run, a new window is opened whereby the designer can run the simulator stepwise and see the output of the simulation at the same time. The syntactic problems of the model can also be viewed after the model checker is run.

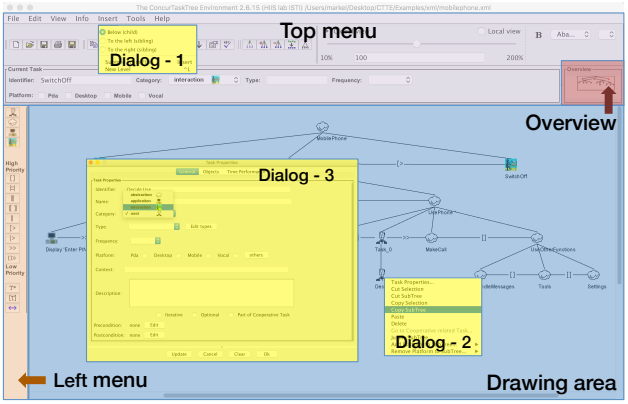

Fig. 1. The areas of interest defined in CTTE

B. Participants, Procedure and Tasks

Thirteen individuals (five female), whose average age was 30 (SD = 6.78), took part in the study. Regarding their expertise with CTTE four participants had built >8 models before, four 4–7, four 1–3 and one of them had never built one even if he was familiar with the notation and the environment. Participants were given the opportunity to view a short video and a manual containing a crash course on the functionalities of the environment, which were free to consult at any time.

Three tasks with an increased level of complexity – both cognitive and in terms of the amount of work – were given to participants. In order not to expose them to foreign domains, the modeling tasks were about common smartphone functionalities. Participants were told to check the consistency of the models at the end of every task although they were free to run the checker at any time.

Task 1. An existing task model had to be refined and extended.
Task 2. A new and optional task had to be inserted between existing tasks.
Task 3. An existing task had to be extended by adding new tasks. The task had to be moved so that it became a child of another existing task.

III. Results

Median completion time for Task 1 was 6:37 (SD = 8:33), 10:42 (SD = 3:25) for Task 2, and 15:37 (SD = 6:55) for Task 3, where a higher standard deviation indicates a higher variability on Task 1. The perceived median difficulty was 2, 3 and 4 for Tasks 1–3 respectively. Completion times are correlated with perceived complexity only for Task 3, where the Spearman's test yields a moderate and significant correlation, $\rho = 0.55$ and $p < 0.05$. The number of times the manual was consulted was strongly correlated with completion times $\rho = 0.70$, $p < 0.01$. That is, it took longer to complete the tasks to those who checked the manual more often. Previous experience is negatively correlated with manual checking times $\rho = -0.62$, $p < 0.05$ – suggesting that the manual was consulted by those who were less experienced. Experience and overall completion times are also negatively correlated $\rho = -0.64$, $p < 0.05$ indicating that the less experienced participants were slower in completing their tasks.

A. UI Events and Workflow Analysis

In order to find regularities in the 4 737 user interface events that were logged, we mined the logs using N-gram analysis [11]. The workflows were isolated after (1) selecting only those N-grams that occurred more often than the number of participants in the study (i.e. thirteen), (2) removing the permutations and (3) merging same subsequent events: for instance, a Select_task event after another Select_task event was transformed into one Multiple_task_selection event – this merging rule was applied to all sequences of same events regardless of their length. As a result of this analysis, we identify five main workflows. Figure 2 shows three of the workflows depicted as a finite state machine where user interface events are conveyed by states and arrows between states indicate a transition between events. We can define the sequence of events as a Markov chain by computing the probabilities between states/events to convey the likelihood of a given transition between two events [12].

1) Set task insertion mode: Because CTT is a hierarchical task model notation, before adding a new task, one has to specify whether the next task is going to be a parent, a sibling or a child of the currently selected task. The button to set the insertion mode is implemented as a looping carousel following this sequence: 'insert sibling to the left', 'insert sibling to the right' and 'insert child'. This means the mode must be set by clicking the button as many times as it is necessary so that the current insertion mode is the one desired by the designer. If the current insertion state is set to 'insert child' and the user wants to insert a sibling to the right they have to click two times. This was captured by the workflow and its specific implementations:

Set_insert_mode_w1: Set_insert_sibling→Set_insert_sibling→Set_insert_child
Set_insert_mode_w2: Set_insert_sibling→Set_insert_child
Set_insert_mode_w21: *Set_insert_mode_w2*→Add_task(s)

2) Populate and edit: Participants created tasks to right after name them or set their category. There are several options to accomplish this: through the 'Dialog - 3' modal dialog, which triggers the Open_task_properties event or through the 'Current task panel' on the 'Top menu', where the name can be set without having to open a dialog. Surprisingly, 74% of the times participants decided to open the modal dialog, while 11% of the times the 'Current task' panel was used when naming the tasks. Opening a dialog requires an extra step, so this may be due to individual preferences or because users ignore there is a more efficient way of giving a name to a task. Note the high probabilities after setting the identifier and category (0.86 and 0.59 respectively) leading to task selection. There are four implementations of this workflow:

Edit_identifier_w1: Select_task(s)→Open_task_properties→Set_identifier
Edit_identifier_w2: Select_task(s)→Set_identifier
Edit_category_w: Select_task(s)→Open_task_properties→Set_category
Add_task_edit_w: Add_task(s)→*Edit_identifier_w1*

3) Add operators to tasks: This workflow describes the activities by which relationships between tasks are established. After selecting a task there is a probability of 0.22 to add an operator to the task and a probability of 0.73 to select another task right after. Note that this transition is not reflected in Figure 2 to prevent clutter:

Add_operator_w: Select_task(s)→Add_operator

4) Check model consistency: This workflow enables users to check whether the model conforms to the grammar and rules defined by the CTT language. The model is checked after setting the category of a task or after adding operators to tasks. Often, those who did not get any consistency problem save the current environment after checking the model:

Check_model_w1: Add_operator→Check_model
Check_model_w2: Set_category→Check_model
Check_model_w11: *Add_operator_w*→Check_model
Check_model_w21: *Edit_category_w*→Check_model

5) Cut, paste and delete: These are typical workflows in any activity involving user interfaces. In CTTE these events happen, understandably, after selecting tasks. Interestingly, some transitions are reflexive: paste occurs after pasting 31% of the time while cutting the selected tasks occurs 15% after cutting tasks. Again, for clarity's sake we do not include these workflows in Figure 2 although, again, the implementations are simple:

Cut_task_w: Select_task(s)→Cut_selection
Paste_task_w: Select_task(s)→Paste_task
Delete_task_w: Select_task(s)→Delete_task

The above workflows indicate that adding new tasks to the model and setting their name, category and adding operators between tasks are the central activities for task modeling. These activities are often preceded with setting the insertion mode for the next task(s) and concatenated with model checking workflows. It was observed that tasks are first added in a batch (typically in a breadth-first fashion) to be later either renamed and then have the operators between tasks added (i.e. Add_task_edit_w + Add_operator_w) or have the operators between tasks added and then the tasks renamed, i.e. Add_operator_w + Edit_identifier_w1.

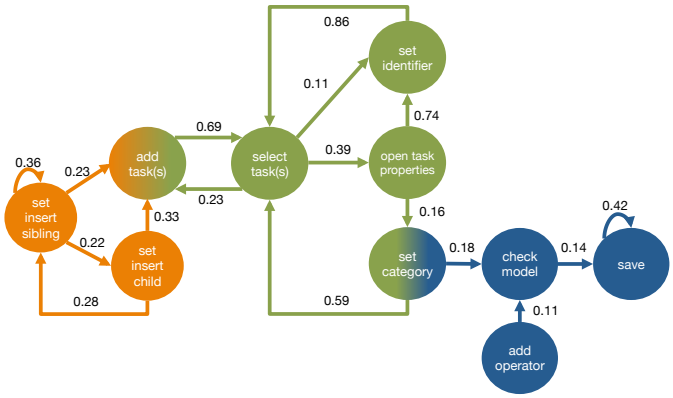

Fig. 2. Exhibited workflows: 'Set task insertion mode' (orange), 'Populate and edit' (green), 'Check the consistency of the model' (blue)

B. Eye-Tracking Data Analysis

The 'Drawing area' receives 67% of the fixations and therefore it is the area where participants' attention dwelled for a longer time (68% of the time). This was expected due to the large amount of space allocated to this area, and also because it is the area where the task model is actually rendered. The manual also receives considerable attention, 13% of fixations. Friedman tests indicate an effect of AOI on dwell time $\chi^2(2) = 68.6$, $p < 0.001$ and on the number of fixations received $\chi^2(2) = 70.81$, $p < 0.001$. Due to the layout of CTTE the area allocated to the AOIs is unbalanced: if we look at the AOIs of the main screen on Figure 1, the 'Drawing area' takes 70% of the area while the 'Left menu' takes a 3%. In order to compare the areas of the main screen, Table I normalizes dwell time and number of fixations per 1 000 pixels. This data shows some interesting insights: while the 'Drawing area' is 24x larger than the 'Left menu', it only gets 2.5x dwell time and 2x fixations when data is normalized. This suggests that, in terms of how the attention is allocated on the screen, the 'Drawing area' is overrepresented and the 'Left menu' should be allocated more space if needed.

TABLE I
DWELL TIME AND NUMBER OF FIXATIONS ON CTTE'S MAIN SCREEN

AOI	Area	%		Normalized	
		Dwell time	Fixations	Dwell time	Fixations
Drawing area	73%	93.6	92	3.07	9.91
Left menu	3%	1.5	1.9	1.23	4.91
Overview	1%	0.3	0.2	0.6	1.84
Top menu	23%	4.6	5.5	0.47	1.86

IV. IMPLICATIONS FOR DESIGN

While the identified workflows fall within the scope of CTTE, we assume that similar activities will be exhibited in other task modeling environments. Indeed, the modeling tools discussed in Section I have a common set of functionalities which include hierarchical task decomposition, the use of operators to indicate temporal relationships between tasks, the possibility of specifying various attributes to characterize the tasks, etc. Consequently, the following implications for design would apply to the common denominator set of functionalities.

A. Support an Efficient Population of the Model

Through the 'Cut, paste and delete' workflow participants copy existing subtrees in the task hierarchy, paste them elsewhere and modify their names and categories as a strategy to avoid repetition. Currently CTTE supports the edition of multiple task operators through a single action, which was used by nine participants. The frequency of the 'Populate and edit' workflow (385 instances) suggests that more effective mechanisms to populate the model need to be provided. For instance, the workarounds exhibited by users could be mimicked: they copied and pasted existing hierarchies and edited them in order to speed up the creation of task models. This suggests that duplicating hierarchies or allowing for the creation of templates would be beneficial in order to populate task models. Bulk editing could also be facilitated if the model was specified using spreadsheets and their typical functionalities including duplication, making minor modifications of entities effectively and arranging the elements of the model hierarchically [13]. Spreadsheets could then be loaded to populate the graphical model for later refinement and model checking purposes.

B. Facilitate a More Direct Manipulation of the Model

We refer to the possibility of moving a task to a different place of the model by replacing the current workaround of employing the 'Cut, paste and delete' workflow with a drag and drop functionality. This is not as straightforward as it may look like: the drag and drop functionality can be ambiguous since it can be used for improving the presentation of the task model and, also, for updating the position of a given task in the logical structure of the model. To give usable support in both cases, the tool should be able to correctly interpret the intention of the designer.

C. Embed Help and Guidance in the Workflows

After the 'Drawing area', the manual was the area that was looked for a longer time. Also, it took longer to complete the tasks to those who checked the manual more often. The screen recordings indicate that participants checked the manual before adding operators to tasks, before adding a task between two tasks and before adding categories to tasks. This suggests that a more effective support should be given to users. Note that CTTE already has a user guide, which is a large document whose consultation disrupts the current workflow. Further information about operators and categories could be implemented in two ways: guidance should be embedded in the 'Left menu'. Despite its central role, the drawing area is overrepresented and allocates a larger area than it should. Therefore, sacrificing this allocation in favor of the 'Left menu' would not be a hindrance. Alternatively, help could be provided as an option through a modal dialog that would open when mouse right clicking the operator or category.

D. Provide More Effective Task Insertion Modes

The 'Set task insertion mode' workflow suggests that a more effective way of indicating where to place the next task in the model is needed. Users sometimes were found to make mistakes when adding a new task in the intended position within the model. While the insertion mode carousel described earlier is to blame for this, these errors could be prevented if on hovering, tasks in the 'Drawing area' would show actionable outgoing arrows from the current task to the parent, sibling and child tasks. Clicking on the arrows would add a task to its corresponding location.

V. CONCLUSION

The lack of evidence about how designers use task modeling tools prevents us from knowing whether there are barriers to effective use. This seminal study addresses this gap by identifying five workflows that illustrate how software engineers go about their task modeling activities. These workflows inform implications for design for the next generation of task modeling tools and constitute hypotheses for future research.

REFERENCES

[1] F. Paterno, *Model-Based Design and Evaluation of Interactive Applications*, 1st ed. London, UK, UK: Springer-Verlag, 1999.

[2] G. C. van der Veer, B. F. Lenting, and B. A. Bergevoet, "GTA: Groupware task analysis modeling complexity," *Acta Psychologica*, vol. 91, no. 3, pp. 297 – 322, 1996, usage of Modern Technology by Experts and Non-professionals. [Online]. Available: http://www.sciencedirect.com/science/article/pii/0001691895000658

[3] D. Scapin and C. Pierret-Golbreich, "Towards a method for task description: MAD," *Work with display units*, vol. 89, pp. 371–380, 1989.

[4] G. Mori, F. Paterno, and C. Santoro, "CTTE: support for developing and analyzing task models for interactive system design," *IEEE Transactions on Software Engineering*, vol. 28, no. 8, pp. 797–813, Aug 2002.

[5] B. E. John, K. Prevas, D. D. Salvucci, and K. Koedinger, "Predictive human performance modeling made easy," in *Proceedings of the SIGCHI Conference on Human Factors in Computing Systems*, ser. CHI '04. New York, NY, USA: ACM, 2004, pp. 455–462. [Online]. Available: http://doi.acm.org/10.1145/985692.985750

[6] C. Martinie, P. Palanque, and M. Winckler, *Structuring and Composition Mechanisms to Address Scalability Issues in Task Models*. Berlin, Heidelberg: Springer Berlin Heidelberg, 2011, pp. 589–609. [Online]. Available: https://doi.org/10.1007/978-3-642-23765-2_40

[7] S. Caffiau, D. Scapin, P. Girard, M. Baron, and F. Jambon, "Increasing the expressive power of task analysis: Systematic comparison and empirical assessment of tool-supported task models," *Interacting with Computers*, vol. 22, no. 6, pp. 569–593, 2010. [Online]. Available: http://dx.doi.org/10.1016/j.intcom.2010.06.003

[8] M. Halbrügge, M. Quade, K.-P. Engelbrecht, S. Möller, and S. Albayrak, "Predicting user error for ambient systems by integrating model-based ui development and cognitive modeling," in *Proceedings of the 2016 ACM International Joint Conference on Pervasive and Ubiquitous Computing*, ser. UbiComp '16. New York, NY, USA: ACM, 2016, pp. 1028–1039. [Online]. Available: http://doi.acm.org/10.1145/2971648.2971667

[9] D. M. Hilbert and D. F. Redmiles, "Extracting usability information from user interface events," *ACM Comput. Surv.*, vol. 32, no. 4, pp. 384–421, Dec. 2000. [Online]. Available: http://doi.acm.org/10.1145/371578.371593

[10] C. Ehmke and S. Wilson, "Identifying web usability problems from eye-tracking data," in *Proceedings of the 21st British HCI Group Annual Conference on People and Computers: HCI...But Not As We Know It - Volume 1*, ser. BCS-HCI '07. Swinton, UK, UK: British Computer Society, 2007, pp. 119–128. [Online]. Available: http://dl.acm.org/citation.cfm?id=1531294.1531311

[11] C. Buchta, K. Hornik, I. Feinerer, and D. Meyer, "tau: Text analysis utilities. Available at http://cran.r-project.org/web/packages/tau/."

[12] H. Thimbleby, P. Cairns, and M. Jones, "Usability analysis with markov models," *ACM Trans. Comput.-Hum. Interact.*, vol. 8, no. 2, pp. 99–132, Jun. 2001. [Online]. Available: http://doi.acm.org/10.1145/376929.376941

[13] K. S.-P. Chang and B. A. Myers, "Using and exploring hierarchical data in spreadsheets," in *Proceedings of the 2016 CHI Conference on Human Factors in Computing Systems*, ser. CHI '16. New York, NY, USA: ACM, 2016, pp. 2497–2507. [Online]. Available: http://doi.acm.org/10.1145/2858036.2858430