Innovative Key Features for Mastering Model Complexity: FlexiLab, a Multimodel Editor Illustrated on Task Modeling

Nicolas Hili, Yann Laurillau, Sophie Dupuy-Chessa, and Gaëlle Calvary
Univ. Grenoble Alpes, LIG, F-38000 Grenoble, France
CNRS, LIG, F-38000 Grenoble, France
{firstname.lastname@imag.fr}

ABSTRACT

Modeling Human Computer Interaction (HCI) is nowadays practiced by IT companies. However, it remains a straightforward task that requires some advanced User Interface (UI) modeling tools to ease the design of large-scale models. This includes tackling massive UI models, multiplicity of models, multiplicity of stakeholders and collaborative editing.

This paper presents a UI multimodel editor for HCI, illustrated on task modeling. We present innovative key features (genericity, creativity, model conformity, reusability, etc.) to facilitate UI model design and to ease interaction.

ACM Classification Keywords

H.5.2 Information Interfaces and Presentation (e.g. HCI): User Interfaces; D.2.2 Software Engineering: Design Tools and Techniques

Author Keywords

Human Computer Interaction; Multimodel editors; Task Model

INTRODUCTION

In the context of the growing use of User Interface (UI) models by IT industry (e.g. UIs for nuclear power plant's control command [5]), we designed and implemented *FlexiLab*, a new UI multimodel editor to master the complexity of UI model design. It addresses several industrial concerns related to the complexity of UI model edition, such as massive models, multiplicity of models, multiplicity of stakeholders, and collaborative editing. Indeed, the primary goal of existing UI model editors (e.g. [8, 3, 6, 1, 7]) has so far always been to support research on novel notations, novel UI models or to illustrate novel approaches (e.g. Cameleon framework, CEDAR architecture [4, 1]). There is now a need to propose new editors and to invent new features to facilitate exploitation by companies for large-scale systems.

EICS'15, June 23-26, 2015, Duisburg, Germany
ACM 978-1-4503-3646-8/15/06.
http://dx.doi.org/10.1145/2774225.2775439

In order to address model multiplicity, scalability, reusability, readability, and collaborative editing, we present a set of innovative key features for model design and user interaction with massive models.

FLEXILAB, A MULTIMODEL EDITOR FOR HCI: OVERVIEW

FlexiLab is a multimodel editor implemented as a web-based application for Human Computer Interaction (HCI) and supports editing of multiple UI models. It covers most of the abstraction layers of the Cameleon framework [4]: task and domain models, abstract user interface model (AUI) and models of context of use. Its architecture is composed of a client side rendering and managing the UI on a web browser and a server side running a NodeJS server. We chose web technologies to ease the deployment of FlexiLab, to facilitate the development of the collaborative editing features thanks to the client-server architecture, and as they are mature enough to support model-driven engineering approaches. Multiplicity of models is supported thanks to internal data representations based on JavaScript Object Notation (JSON) combined with generic modules implementing metamodel compliance checking and model import/export for various formats and metamodels (e.g. eXtensible Markup Languages (XML)).

From a user perspective, Fig. 1 and 2 represent the general layout of the main user interface: in order to support multimodel edition, FlexiLab is designed to simultaneously display multiple editing areas ❶ (cf. Fig. 1), one per type of model (e.g. task and domain models) as well as to display a single editing area (cf. Fig. 2), using the entire screen space freed by the other editing areas. Separators between editing areas ❷ enable fast switching between editing areas as well as switching between a multiple editing area view and a sin-

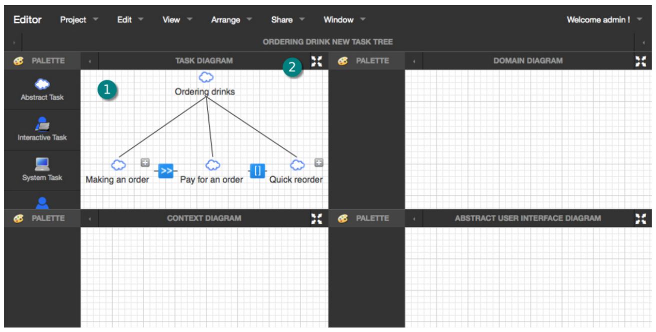
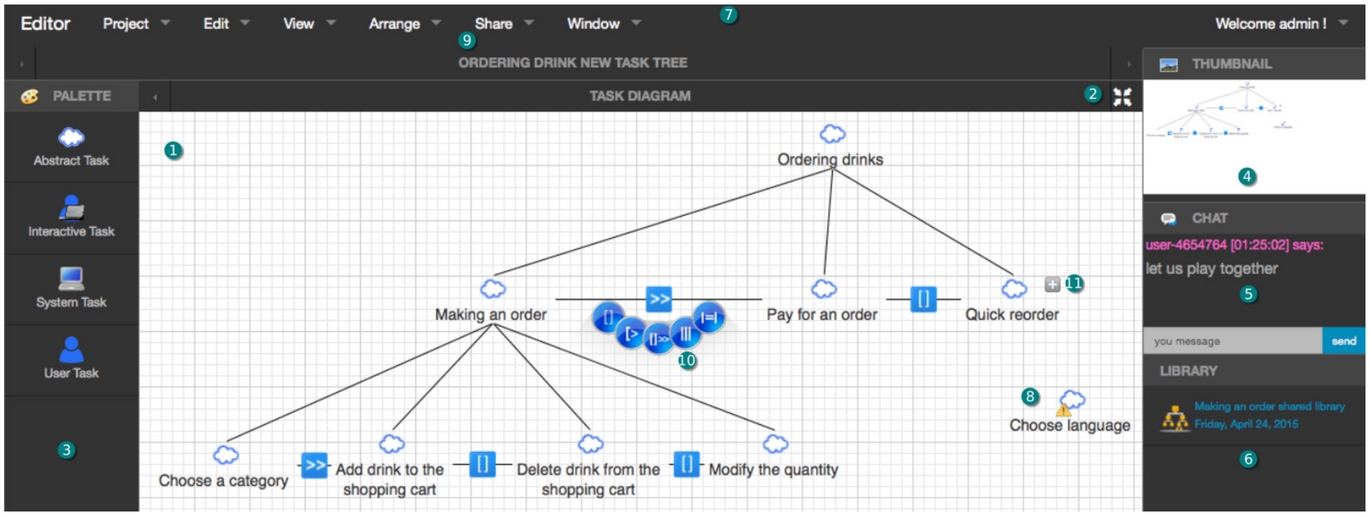


Figure 1: Overview of FlexiLab



user-4654764 [01:25:02] says:
let us play together

Figure 2: Overview of the FlexiLab Task Model Editor (single editing area view)

gle editing area view. On the left side of each editing area, a contextual tool palette 3 is available and offers specific tools depending on the type of model. Specific views (4, 5 and 6) are displayed on the right side of the editing area (we will further detail them). Finally, general features (e.g. open/save, import/export, copy/paste, etc.) are accessible through standard menus in a menubar 7.

In the following, we illustrate our approach on the task model editor (cf. Fig. 2).

INNOVATIVE KEY FEATURES FOR DESIGN

When designing huge models in IT companies, designers have to face several requirements like model scalability, reusability, readability and so forth. To this aim, we propose some innovative key features to address them.

Metamodel-tolerant Approach for Combining Genericity, Creativity and Conformity

Since HCI models can be used as support for design as well as communication, bridging the gap between creativity and conformity is a main requirement for HCI editors.

For this reason, we favored a "*metamodel-tolerant*" approach where generic models are loosely tied to a metamodel definition and thus not constrained by this latter. The main benefit of this approach is to not hinder the creativity of the designer by constraining him to a particular definition of a metamodel. For example, he or she can add a new task without binding it to another one 8 and save the model in a non-valid state.

In order to combine creativity and conformity, we deciced to support a loose model conformity control. Loose model conformity means is "*metamodel-tolerant*" and is ensured by rules handwritten inside the tool. Warning and error icons can warn the designer of a non-valid state and guide him to correct the problem. The main benefit of this approach is to not prevent him to save the model in a non-valid state and therefore not reduce creativity.

In addition, we plan to implement a *complementary* strict model conformity control that could be performed on demand or during an export to ensure that the model is conformed to a specific metamodel. We currently investigate existing model validators, like EMF Validation Framework [1] or moddle [2].

To support model genericity, we implemented export features in several formats. Currently, FlexiLab can export in JSON and XML formats. We chose to support both as JSON is easier to manipulate on the web while XML is more structured and favored in modeling environments like Eclipse. This also ensures the interoperability at syntactical level and allows the designer to export the models according to different metamodel definitions. In addition, FlexiLab also supports graphical exports. Two formats are supported, Scalable Vector Graphics (SVG) and Portable Network Graphics (PNG). This permits to use the model as support for communication.

Modular Design for Reusability

When designing huge models, there is a strong interest in favoring a modular approach. We propose the concept of *fragment*, a part of model designed in a previous project to be reused. Fig. 2 illustrates this feature on the task editor. A library view 6 was implemented to import reusable fragments on the current model. Fragment import is performed with a Drag and Drop action in the same way as adding a graphical element from the palette. For example, the designer can import a task tree fragment in a single action. The tree is collapsed by default and the designer can expand it on the editor.

Sharing and Communication for Collaborative Work

To address the model scalability requirement and take advantage of a modular design of huge models, collaboration is usually addressed at two different levels in existing collaborative editors[3]: at informational level first, through communication features (e.g. chat) between all the stakeholders of a project,

[1]https://wiki.eclipse.org/EMF/Validation
[2]https://github.com/bpmn-io/moddle/
[3]e.g. LucidChart (http://lucidchart.com)

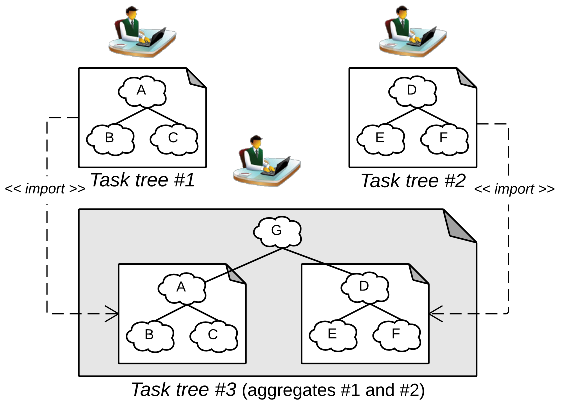


Figure 3: Aggregation of several task tree fragments

but also at design level with model sharing features. However, these features are currently not available in existing HCI editors. In FlexiLab, we chose to cover both levels.

Communication between stakeholders is possible in real-time through an interactive chat 5. Besides, we implemented a *share* feature that enables a designer to share a fragment of model. This feature is accessible from the menubar 9. Combined with the library view, it allows several designers to work on separate fragments of the same model. Fig. 3 illustrates this feature. Two designers can simultaneously design *task tree #1* and *task tree #2* that will be further aggregated by a third designer in the *task tree #3*.

Alongside the innovative key features for design, we also focused on implementing key features to ease interaction.

KEY FEATURES FOR INTERACTION

We essentially address the *guidance*, *information density* and *explicit control* ergonomic criteria [2] by providing features that ease the direct manipulation of models. These features were validated through an experiment with two groups of twenty students.

Contextual Toolbar

The first key feature we implemented is a *contextual toolbar* for recurrent actions and properties. Used alongside the palette and the property menu for sporadic property editing, it efficiently helps the designer who can favor it to perform the most frequent actions and edit the most frequent properties.

The contextual toolbar is opened by a single click on a graphical element. It is composed of several icons corresponding to specific actions. For example, Fig. 4 illustrates the contextual toolbar for a task. From left to right, the five actions are: *rename the task*, *add subtasks*, *connect to a subtask*, *edit the properties* and *delete the task*. The contextual toolbar for the temporal relationship is illustrated on Fig. 2 10. It offers a quick way to change the temporal operator between two subtasks without rebuilding the link from the palette.

Smart Connection and Disconnection

We implemented a smart connection and disconnection engine in the multieditor. It ensures the correct ordering and

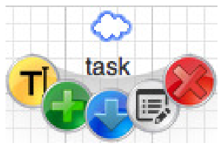


Figure 4: Contextual Toolbar for a Task

arrangement of subtasks sharing the same ancestor, during a connection or disconnection action. Its main benefit is to automatically handle the creation and deletion of the temporal relationships between tasks. In most exiting tools, the designer has to manually create the relationships between tasks. When a task is added or removed, he or she has to manually reconnect the link in the best case scenario, or remove it and create a new one in the worst case. In our tool, the smart connection and disconnection engine automatically handles this, thus reducing the effort of the designer.

Fig. 5a illustrates the disconnection of a task from its ancestor. When disconnecting the task, the two temporal relationships are automatically merged in order to reconnect the two distant tasks (see Fig. 5b). Conversely, when a task needs to be added between two connected tasks (see Fig. 5c), the engine is able to automatically split the temporal relationship in order to connect the three tasks together (see Fig. 5d).

Additional Features

We also implemented some common features we can find in other tools. *Rearrangement* helps the designer to manually or automatically rearrange a task tree. Automatic rearrangement optimizes the rendering and prevents graphical element overlaps. If the designer wants to manually rearrange a task tree, visual guidelines can guide him to align graphical elements for readability purpose. We also provided some features to address the problem of huge models that cannot be entirely displayed on the screen: we implemented a view to display an

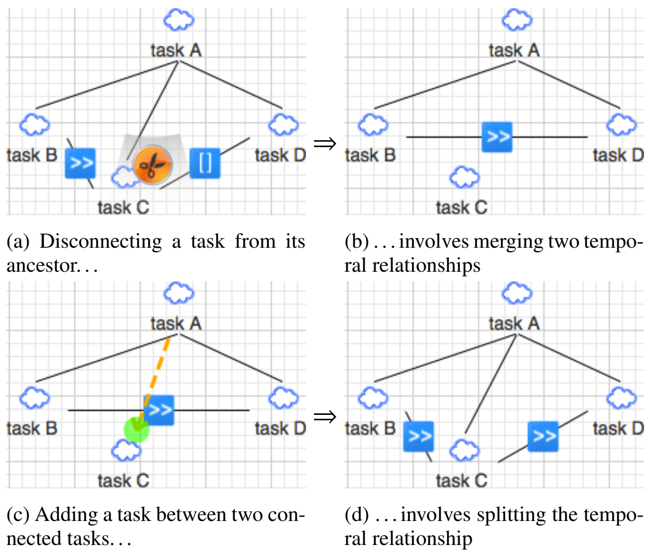

task A
task B
task C
task D
>>
[]
⇒
task A
task B
task C
task D
>>
(a) Disconnecting a task from its ancestor. . .
(b) . . . involves merging two temporal relationships
task A
task B
task C
task D
>>
⇒
task A
task B
task C
task D
>>
>>
(c) Adding a task between two connected tasks. . .
(d) . . . involves splitting the temporal relationship

Figure 5: Smart Connection & Disconnection Features

Table 1: Comparison between Task Editors

Editor	Genericity	Creativity	Reuse	Collaboration
CTTE	No	No	Import	No
KMADE	Domain	Yes	No	No
e-COMM	Context (basic)	No	Basic	No
HAMSTERS	Yes	Yes	Yes	No

outline 4 of the currently edited model. In addition, the designer can also collapse and expand some tasks to reduce the size of the task tree and increase the readability of the model. A collapsed task is illustrated with a "+" symbol alongside its icon 11.

IMPLEMENTATION

We chose to develop FlexiLab as a web application to facilitate its use. We divided its implementation into two parts: a client side displayed on a web browser and a server side running on a NodeJS server. JavaScript is used for both sides and a SQL database is used to store the models on the server side.

On the client side, we use the *Foundation*[4] framework to design the multimodel editor. The editor itself is based on the HTML5 Canvas library.

Communication between the client side and the server side is asynchronous. We mainly use *Asynchronous JavaScript and XML* (AJAX) for passive communications between both sides, and *Web Sockets* for the chat and the library views. JSON was favored for data exchanges.

RELATED TOOLS: KEY FEATURES FOR DESIGN

Many tools and notations were proposed to design hierarchical task tree models [8, 3, 6, 7]. Based on CTT notation, ConcurTaskTrees Environment (CTTE) [8] is a single-user monomodel editor supporting task model design and analysis, available as a Java standalone application. It supports reuse through import/export of sub-tree fragments. For the design of cooperative task trees, the editor supports parallel editing of task trees (one per role plus the cooperative task tree).

K-MADe is an editor supporting the K-MAD notation [3], available as a Java standalone application. It supports joint editing of task and domain models. The editor supports creativity as an added task may be orphan: an alert box indicates whether the model is compliant with the K-MAD notation.

e-COMM [6] is an online web-based editor using the Collaborative and MultiModal (COMM) notation, implemented in C#/Silverlight. It addresses specification of multi-user multimodal systems. It allows the designer to create task trees, business roles and contexts of use (basic description through a texfield). Reuse is basically supported by separated and parallel editing of task trees or sub-trees.

HAMSTERS [7] is a task tree notation related to ICO. The editor is integrated in PetShop and handles: tasks, domain, petri nets. It favours creativity rather than guidance (e.g. a temporal operator may exist independently of any task connector). Collaboration is covered by sub-tree fragments that can be added by reference in a task tree: if a fragment is modified, each reference is also automatically modified.

[4]http://foundation.zurb.com

Compared to existing tools, FlexiLab supports cooperation thanks to communication, fragment sharing and multiple models. It also provides advanced features to support creativity and model conformity checking on demand.

CONCLUSION

This paper presents innovative key features to address the complexity of HCI model design. Those features were implemented in a multimodel editor for HCI.

In future works, we plan to generalize those features to other models, with a special focus on collaborative development, through the library and chat.

ACKNOWLEDGMENTS

This work is funded by the French Connexion Cluster (Programme d'Investissements d'avenir / Fonds national pour la société numérique / Usages, services et contenus innovants).

REFERENCES

1. P. Akiki, A. Bandara, and Y. Yu. 2013. RBUIS: Simplifying Enterprise Application User Interfaces Through Engineering Role-based Adaptive Behavior. In *Proceedings of EICS'13*. ACM, New York, NY, USA.
2. C. Bastien and D. Scapin. 1993. Ergonomic criteria for the evaluation of human-computer interfaces. (1993).
3. S. Caffiau, D. Scapin, P. Girard, M. Baron, and F. Jambon. 2010. Increasing the expressive power of task analysis: Systematic comparison and empirical assessment of tool-supported task models. *Interacting with Computers* 22, 6 (Nov. 2010), 569–593.
4. G. Calvary, J. Coutaz, D. Thevenin, Q. Limbourg, L. Bouillon, and J. Vanderdonckt. 2003. A unifying reference framework for multi-target user interfaces. *Interacting with Computers* 15, 3 (2003), 289–308.
5. E. Céret, G. Calvary, and S. Dupuy-Chessa. 2013. Flexibility in MDE for scaling up from simple applications to real case studies: illustration on a Nuclear Power Plant. In *25ème conférence francophone sur l'Interaction Homme-Machine, IHM'13*. AFIHM, ACM, Bordeaux, France.
6. F. Jourde, Y. Laurillau, and L. Nigay. 2010. COMM Notation for Specifying Collaborative and Multimodal Interactive Systems. In *Proceedings of EICS'10*. ACM, New York, NY, USA, 125–134.
7. C. Martinie, E. Barboni, D. Navarre, P. Palanque, R. Fahssi, E. Poupart, and E. Cubero-Castan. 2014. Multi-models-based Engineering of Collaborative Systems: Application to Collision Avoidance Operations for Spacecraft. In *Proceedings of EICS'14*. ACM, New York, NY, USA, 85–94.
8. G. Mori, F. Paternò, and C. Santoro. 2002. CTTE: support for developing and analyzing task models for interactive system design. *Software Engineering, IEEE Transactions on* 28, 8 (2002), 797–813.