HAL
open science

# Self-Explanatory User Interfaces by Model-Driven Engineering

Alfonso García Frey, Gaëlle Calvary, Sophie Dupuy-Chessa

## To cite this version:

Alfonso García Frey, Gaëlle Calvary, Sophie Dupuy-Chessa. Self-Explanatory User Interfaces by Model-Driven Engineering. Proceedings of the CHI'10 Workshop on Model Driven Development of Advanced User Interfaces (MDDAUI'10), 2010, Atlanta, United States. pp.1-4. hal-00953332

**HAL Id: hal-00953332**

**https://inria.hal.science/hal-00953332v1**

Submitted on 28 Feb 2014

**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.

# Self-Explanatory User Interfaces by Model-Driven Engineering

**Alfonso García Frey, Gaëlle Calvary and Sophie Dupuy-Chessa**
University of Grenoble, CNRS, LIG
385, avenue de la Bibliothèque, 38400, Saint-Martin d'Hères, France
{Alfonso.Garcia-Frey, Gaelle.Calvary, Sophie.Dupuy}@imag.fr

## ABSTRACT
Modern User Interfaces (UI) must deal with the increasing complexity of applications as well as new features such as the capacity of UIs to be dynamically adapted to the context of use. The complexity does not necessarily imply a better quality. Thus, it becomes necessary to make users understand the UIs. This paper describes an on-going research about Self-Explanatory User Interfaces (SE-UI) by Model-Driven Engineering (MDE). Self-explanation makes reference to the capacity of a UI to provide the end-user with information about its rationale (which is the purpose of the UI), its design rationale (why is the UI structured into this set of workspaces?, what's the purpose of this button?), its current state (why is the menu disabled?) as well as the evolution of the state (how can I enable this feature?). Explanations are provided by embedded models. We explore model-driven engineering to understand why and how this approach can lead us to overcome shortcomings of UI quality successfully.

### Author Keywords
Self-Explanatory User Interfaces, UI quality, help, design rationale, model-driven engineering, model transformation.

### ACM Classification Keywords
H.5.2 User Interfaces: Theory and method.

## INTRODUCTION
### Motivation
On the one hand, most software is too hard to use."Modern applications such as Microsft Word have many automatic features and hidden dependencies that are frequently helpful but can be mysterious to both novice and expert users" [15]. Users may require assistance while interacting with a User Interface (UI). Ideally, the UI must guide the user in accomplishing a task the application was designed for. The user can request help about functionality, features, or any information about the process of the task that is being performed. The UI must be able to provide the correct answer, giving the necessary information to the user in an appropiate format. This can take place at any time in the whole interaction process between both the user and the UI. However, modern applications cover only a few questions the user may have, or provide a general help instead of a clear and concise answer to a given question. Furthermore, help is created ad-hoc, this is, it has been previously generated and it's not able to cover new questions at run-time because they were not considered by the designers. UI design problems are not covered at all because the designers are not aware of them.

Moreover, the UI must deal with users having different levels of expertise. Even many long-time users never master common procedures [6] and in other cases, users must work hard to figure out each feature or screen [6].

The problem is greater for Plastic UIs [5, 19]. Plastic UIs demand dynamic adaptation also for help systems because from now on, developers can't afford to consider all the different contexts of use one by one coding all possible ad-hoc solutions by hand. This complicates the prediction of the result and the final quality, making difficult the design choices.

As a result, dynamic solutions are required also for help systems. These help systems must now be aware of the context of use (user, platform and environment), the task, the structure and presentation of the UI.

### MDE and MB-UIDE approaches
On the other hand, Model-Driven Engineering (MDE) exists since long time ago and its recently applied to the engineering of UIs. It consists in describing different features of UIs (e.g., task, domain, context of use) in models from which a final UI is produced [18] according to a forward engineering process. MDE of UI is assumed to be superior to the previous Model-Based User Interface Development Environment versions since it makes the UI design knowledge explicit, and external for instance as model-to-model transformations and model-to-code compilation rules [2]. However, neither Model-Based User Interface Development Environment automatic generated UIs nor final UIs produced by MDE have enough quality, forcing designers to manually tweak the generated UI code [2]. Design knowledge can not be always explicitly represented into the models, but it has a potential to help final users. Some models as for instance the task model have this potential explicitly represented, and they can contribute also to guide and help the user.

Permission to make digital or hard copies of all or part of this work for personal or classroom use is granted without fee provided that copies are not made or distributed for profit or commercial advantage and that copies bear this notice and the full citation on the first page. To copy otherwise, or republish, to post on servers or to redistribute to lists, requires prior specific permission and/or a fee.
*CHI 2009*, April 4 - 9, 2009, Boston, Massachusetts, USA.
Copyright 2009 ACM 978-1-60558-246-7/09/04...$5.00.

This research will study how Self-Explanatory User Interfaces (SE-UIs) can be built using the MDE. A SE-UI is a UI with the capacity of understanding its own rationale and consequently having the abilities of answer questions about it. We aim to provide a method for creating SE-UIs analyzing the relations between the different levels of abstraction in our MDE-compliant approach for developing UIs as well as the different models presented into the UsiXML specification and their relations. Complementary views of the UI are also considered into this research.

The rest of the paper presents the related work and our contribution to the field.

## RELATED WORK

The two major areas involved in our Self-Explanation approach are MDE and UI quality. The related works of the next two sections allow us to set up the bases of our contribution.

### MDE

The Cameleon Reference Framework [4] presented a MDE-compliant approach for developing UIs consisting of four different levels of abstraction: Task Model, Abstract User Interface, Concrete User Interface and Final User Interface. These levels correspond, in terms of MDE, to Computing-Independent Model (CIM), Platform-Independent Model (PIM), Platform-Specific Model (PSM) and the code level respectively. In the Model-Driven Development (MDD) many transformation engines for UI development have been created. Several researches have addressed the mapping problem for supporting MDD of UIs: Teresa [14], ATL [10], oAW [10] and UsiXML [17] among others. A comparative analysis can be found in [9]. Semantic Networks have been also covered for UIs [8]. The Meta-UI concept was firstly proposed in [7] and deeply explored later in many other works. In one of them [16], the concept of Mega-UI is studied introducing Extra-UIs, allowing a new degree of control by the use of views over the (meta-)models. We will focus on it later as these views are relevant for the explanation of the UI and consequently for the end-user's comprehension.

### UIs Quality

Help systems have been extensively studied. One of the most relevant works is the Crystal application framework [15]. Inspired by the Whyline research [11], "Crystal" provides an architecture and interaction techniques that allow programmers to create applications that let the user ask a wide variety of questions about why things did and did not happen, and how to use the related features of the application without using natural language [15]. Even if this approach does not cover the capacity of adaptation to different contexts of use, it represents an important improvement in quality for the end-user in terms of achieved value. Quality can be improved regarding not only the *achieved value*, but also from the perspectives of *software features* and *interaction experiences* [12]. The integration of Usability Evaluation Methods (UEM) [13] into a MDA process has been proved to be feasible in [1]. In particular, the evaluation at the PIM or PSM should be done in an interactive way until these models have the required level of usability. Different UEMs (e.g., heuristic evaluation, usability test, etc) can be applied iteratively until the concerned models have the required level of usability. A set of ergonomic criteria for the evaluation of Human-Computer Interaction (HCI) can be found in [3].

This research improves quality of help systems allowing a new range of questions. Adaptation to the context of use is now considered since SE-UIs understand their own rationale.

## CONTRIBUTION

### End-User's point of view

The goal of this work is to study how SE-UI can be built by MDE. One of the ways to explore SE-UI involves the task model and its rationale. A task model describes the user's task in terms of objectives and procedures. Procedures recursively decompose tasks into subtasks until one or more elementary tasks are reached, i.e., tasks which would be decomposable into physical actions only ("press the button"). A task model is well-defined then by the following terms:

**Nodes** Containing abstract tasks

**Leaves** Special nodes containing elementary tasks

**Branches** Expressing logical and temporal relations between tasks, subtasks and elementary tasks

The explicit information contained into the branches can help and guide the end-user answering questions related to different aspects of the UI. For instance, regarding the rationale of the UI questions like *which is the purpose of the UI?* can be successfully answered; also, questions as *why is the UI structured into this set of workspaces?* or *what is the purpose of this button?* can be explained understanding the relations of the design rationale. The current state of the UI and consequently the state of the application, can trigger a different kind of questions to the end-user as for instance *why is the menu disabled?*, as well as questions related to the overall progress of a task or questions about the evolution of the current state of the application as for example *how can I enable this feature?* Answers for all of them can be obtained exploring tasks and subtasks (nodes), elementary tasks (leaves) and relations between them (branches) in the task model.

This work will study also how different views of the model centered in extra-UIs, can help the end-user to understand the UI. A extra-UI [16] is a UI which represents and gives the control of a UI through a model. It is in a sense the UI of the configuration of a UI. These views can improve the end-user's comprehension as they are relevant for the explanation of the UI. Extra-UIs provide a new degree of control over the (meta-)models of the UI; both designer and end-user can see and understand how tasks are decomposed and how tasks are represented in a specific UI. In other words, how the UI is interfacing the interaction between the application and the own user. Designers can express this interaction in the form of relations between tasks and elements of the final UI with the method explained in the next section.

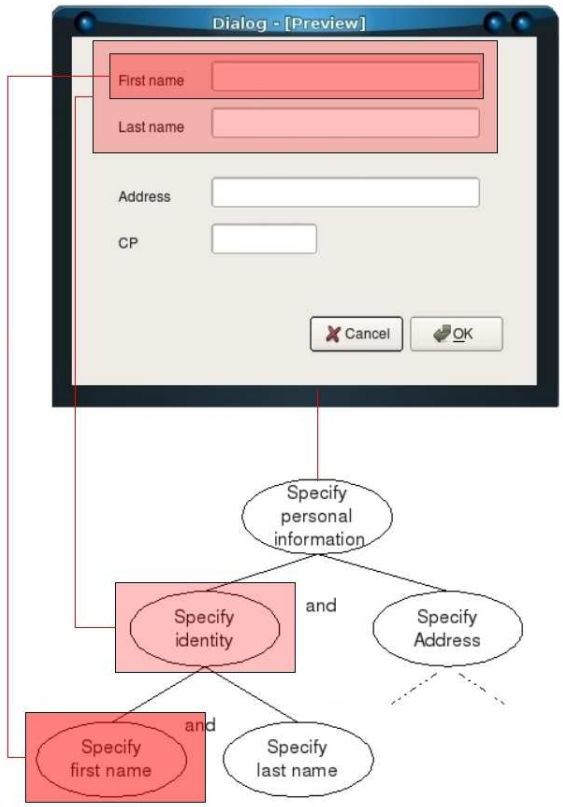


**Figure 1. Association between UI and a task model.**

### Designer's point of view

This work will explore a method to provide designers with a technique to add Self-Explanation to final UIs, specifying how end-user's tasks are directly related to the final UI level. The method consists in four steps:

1. Specify the final UI of the model-compliant application that it will be extended with SE functionality.
2. Define the task model of the application.
3. Specify the relations between both the task model and the final UI.
4. A new final SE-UI will be generated from these relations, adding SE functionality in real-time.

To support this method, we will supply designers with an editor in which tasks models and final UIs can be created. Both of them will coexist at the same time into the same workspace inside this editor. Once the task model and the final UI are represented, the designer will draw direct connections between elements of the task model and elements of the final UI, linking for instance, widgets with subtasks, as we can see in figure 1. Here, the task called *Specify identity* is visually connected to a group of widgets, containing two labels and two input fields. Then, the elementary task *Specify first name* which is also a subtask, is connected to a new subgroup of two widgets, one label and one input field.

**Figure 2. Help message derived from connections in Figure 1.**

The purpose of the method is to allow designers to specify direct relations between tasks and different elements of the final UI. The main advantage for designers is that from now on, there is no need of a deeply comprehension of all the model-to-model and model-to-code transformations between all the four levels of MDE. A visual representation gives direct information about these relations because connections are explicitly represented in a visual render, in which the final UI and the task model levels share the same workspace.

To allow end-user questions this study will consider a help button (figure 2) as a first approach. Other approaches can be considered as well. By clicking this help button, the application enters in a help mode where the end-user can ask about different elements of the UI just by clicking on them. Answers will be generated in real-time in different ways. The following section illustrates an example of this procedure.

### Answering questions

This work will study also how different questions can be answered. The first approach will associate a description to each element (tasks, relations, widgets, etc.) of figure 1. Other approaches like semantic networks [8] can be considered in the future. If the end-user asks himself, for instance, *Why is the OK button disabled?*, by clicking on this button using the special help mode, the system can say that the task is not completed. In figure 2 the message is dynamically derived from the relations of figure 1. For an edit box, the application can say *You must fill in + Description of the task*, where *your personal information* is the description. A more specific information can be generated exploring the task model. For instance, we can travel all the subtasks of the uncompleted task. In the example before, we can answer also that the user needs to fill in the first name and the last name, because these subtasks are both uncompleted.

## CONCLUSION

This research takes a significant step forward in the development of high quality UIs. It explores MDE of UIs to provide Self-Explanation at run-time, analysing the four levels of the MDE-compliant approach for developing UIs and the different models presented into the UsiXML specification and their relations. Complementary views of the UI are explored

in order to exploit these models, explaining the UI itself and giving to the user a new dimension of control by these views. This opens the work on End-User programming.


## ACKNOWLEDGMENTS

This work is funded by the european ITEA UsiXML project.


## REFERENCES


1. S. Abraho, E. Iborra, and J. Vanderdonckt. *Maturing Usability*, chapter Usability Evaluation of User Interfaces Generated with a Model-Driven Architecture Tool, pages 3–32. Human-Computer Interaction Series. Springer-Verlag, 2008.

2. N. Aquino. Adding flexibility in the model-driven engineering of user interfaces. In *EICS '09: Proceedings of the 1st ACM SIGCHI symposium on Engineering interactive computing systems*, pages 329–332, New York, NY, USA, 2009. ACM.

3. J. C. Bastien and D. L. Scapin. Ergonomic criteria for the evaluation of human-computer interfaces. 0 RT-0156, INRIA, 06 1993.

4. G. Calvary, J. Coutaz, D. Thevenin, Q. Limbourg, L. Bouillon, and J. Vanderdonckt. A unifying reference framework for multi-target user interfaces. *Interacting With Computers Vol. 15/3*, pages 289–308, 2003.

5. B. Collignon, J. Vanderdonckt, and G. Calvary. Model-driven engineering of multi-target plastic user interfaces. In *Proc. of 4th International Conference on Autonomic and Autonomous Systems ICAS 2008*, pages 7–14, 2008. D. Greenwood, M. Grottke, H. Lutfiyya, M. Popescu (eds.), IEEE Computer Society Press, Los Alamitos, Gosier, 16-21 March 2008.

6. M. Corporation. Microsoft inductive user interface guidelines, 2001. http://msdn.microsoft.com/en-us/library/ms997506.aspx.

7. J. Coutaz. Meta-user interfaces for ambient spaces. In *Tamodia'06*, 2006. 8 pages.

8. A. Demeure, G. Calvary, J. Coutaz, and J. Vanderdonckt. Towards run time plasticity control based on a semantic network. In *Fifth International Workshop on Task Models and Diagrams for UI design (TAMODIA'06)*, pages 324–338, 2006. Hasselt, Belgium, October 23-24, 2006.

9. J. González Calleros, A. Stanciulescu, J. Vanderdonckt, D. J.P., and M. Winckler. A comparative analysis of tranformation engines for user interface development. In *Proc. of the 4th International Workshop on Model-Driven Web Engineering (MDWE 2008)*, pages 16–30, Tolouse, France, 2008. CEUR Workshop Proceedings.

10. F. Jouault and I. Kurtev. Transforming models with atl. In *Satellite Events at the MoDELS 2005 Conference*, volume 3844 of *Lecture Notes in Computer Science*, pages 128–138, Berlin, 2006. Springer Verlag.

11. A. J. Ko and B. A. Myers. Designing the whyline: a debugging interface for asking questions about program behavior. In *CHI '04: Proceedings of the SIGCHI conference on Human factors in computing systems*, pages 151–158, New York, NY, USA, 2004. ACM.

12. E. Lai-Chong Law, E. T. Hvannberg, and G. Cockton. *Maturing Usability. Quality in Software, Interaction and Value*. Human-Computer Interaction Series. Springer-Verlag, 2008.

13. E. L. Law, E. T. Hvannberg, G. Cockton, P. Palanque, D. Scapin, M. Springett, C. Stary, and J. Vanderdonckt. Towards the maturation of IT usability evaluation (MAUSE). In *Human-Computer Interaction - INTERACT 2005*, pages 1134–1137. 2005.

14. G. Mori, F. Paterno, and C. Santoro. Design and development of multidevice user interfaces through multiple logical descriptions. *IEEE Trans. Softw. Eng.*, 30(8):507–520, 2004.

15. B. A. Myers, D. A. Weitzman, A. J. Ko, and D. H. Chau. Answering why and why not questions in user interfaces. In *CHI '06: Proceedings of the SIGCHI conference on Human Factors in computing systems*, pages 397–406, New York, NY, USA, 2006. ACM.

16. J.-S. Sottet, G. Calvary, J.-M. Favre, and J. Coutaz. *Megamodeling and Metamodel-Driven Engineering for Plastic User Interfaces: Mega-UI*. 2007.

17. J. Vanderdonckt. A MDA-Compliant environment for developing user interfaces of information systems. In *Advanced Information Systems Engineering*, pages 16–31. 2005.

18. J. Vanderdonckt. Model-driven engineering of user interfaces: Promises, successes, failures, and challenges. In *Proc. of 5th Annual Romanian Conf. on Human–Computer Interaction ROCHI'2008, (Iasi, 18–19 September 2008), pp. 1—10. Matrix ROM, Bucarest*, 2008.

19. J. Vanderdonckt, J. Coutaz, G. Calvary, and A. Stanciulescu. *Multimodality for Plastic User Interfaces: Models, Methods, and Principles*, chapter 4, pages 61–84. 2008. D. Tzovaras (ed.), Lecture Notes in Electrical Engineering, Springer-Verlag, Berlin, 2007.