Self-Explanatory User Interfaces by Model-Driven Engineering

Alfonso García Frey
University of Grenoble, LIG
385, avenue de la Bibliothèque, 38400, Saint-Martin d'Hères, France
Alfonso.Garcia-Frey@imag.fr

ABSTRACT

Modern User Interfaces (UI) must deal with the increasing complexity of applications in terms of functionality as well as new properties as plasticity. The plasticity of a UI denotes its capacity of adaptation to the context of use preserving its quality. The efforts in plasticity have focused on the (meta) modeling of the UI, but the quality remains uncovered. We suggest a method for improving the quality of the UIs by providing explanations about the design of the UI itself: this is, by the use of the Self-Explanation. Self-Explanatory User Interfaces (SEUI) makes reference to the capacity of a UI to supply the end-user with all the information on the rational of the UI, about its constitution (for example, what is the purpose of this button?), its current state (why is the menu disabled?) as well as its evolution (how can I enable this feature?).

This thesis investigates the SEUI by Model Driven Engineering (MDE), where models are kept at run-time allowing the necessary techniques that maintain this link between design and execution.

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.

General Terms
Design, Human Factors

PROBLEM DESCRIPTION AND MOTIVATION

Most of the software is too hard to use. Modern applications such as Microsoft 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 appropriate 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.

The design of a UI remains a very difficult and complex task. The know-how of the designers, this is, the collection of decisions and choices the designers take at the design time, is only used to produced the final UI. However, even if this know-how seems to be a powerful source of knowledge to explain some of the features of the UI, it is not explicitly represented in the final UI.

A UI is normally designed for a particular kind of users that will find the UI easy and comfortable to use. Users out of this group, i. e., users who had not been considered in the design process of the UI, can express difficulties while interacting with it. 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]. This variety of users implies different contexts of use. Context of use refers to a set of observables that characterize the conditions in which a particular system is running. It covers three information spaces: the user model, the platform model, and the physical and social environment model [18]. This diversity of contexts of use is another well-known problem for plastic UIs [5]. In Human-Computer Interaction (HCI), plasticity refers to the ability of UIs to withstand variations of contexts of use while preserving usability. The adaptation of the UI has been addressed using many different approaches over the years, including Machine Learning [8], Model-Driven Engineering (MDE) [17], and Component-oriented services [16]. Regardless of the approach, the tendency has been to focus on functional aspects of adaptation, while neglecting the usability dimension.

Because plastic UIs adapt themselves dynamically to the whole context of use and not only to a specific group of users, help systems must be, from now on, able to adapt themselves dynamically to these different contexts of use as plastic UIs do. 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 decisions and choices at the design time. As a result, dynamic solutions are required also for help systems. These help systems must now be aware of the context of use as well as other factors like the task the user is accomplishing or the structure and the presentation of the UI.

Imperfections in the design of the UI must also be considered because the designers cannot cover at the design time all the different aspects that are involved at the run-time. Designers can also have different levels of expertise and again, the great variety of contexts of use don't allow to assure a good level of quality for each different case. Automatic generated UIs by MBUIDEs nor final UIs produced by MDE have not enough quality, forcing designers to manually tweak the generated UI code [2].

All these problems while designing UIs have also repercussions on the perceived quality from the user's point of view. End-users can have a comprehension of the UI different from the designers, so new questions about the elements of the UI or interactions while performing tasks can appear at any moment at run-time. Thus, it becomes necessary to make users understand the UIs.

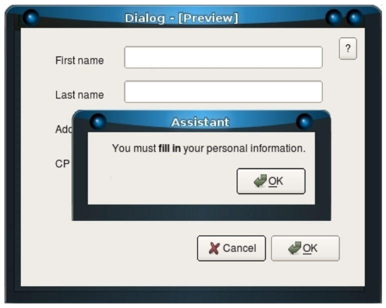
AIMS AND GOALS

This work is focused on an on-going research about Self-Explanatory User Interfaces (SEUI) by Model-Driven Engineering (MDE). Self-explanation makes reference to the capacity of a UI to provide the end-user with information about the rationale of the UI (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?).

In figure 1 a message is displayed by request when the user asks information about the window. SEUIs aim to generate these answers dynamically taking into account the current context of use in which the interaction between the end-user and the UI is taking place.

To accomplish this research we study how SEUIs can be built by MDE. The MDE approach allows us to separate the different concerns of the UI in several levels of abstraction. The Cameleon Reference Framework [4] illustrates this process. Some models as for instance the task model have part of the design knowledge explicitly represented, and they can contribute also to guide and help the user.

At the end we hope to develop a method to automatically generate dynamic help systems that are aware of the context of use, the current task the user is performing, the structure of the UI and its representation to the user. An editor will be developed to build SEUIs.



You must **fill in** your personal information.

Figure 1. A help message leads the user through the UI.

This goal has different impacts on the user and the designer as explained in the following sections.

End-User's point of view

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 thesis will study also how different views of the model, in form of extra-UIs, can help the end-user to understand the UI. A extra-UI [19] is a UI which represents and gives the control of a UI through a model. 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

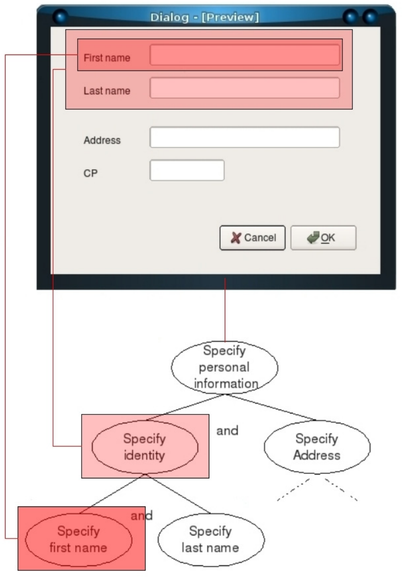


Figure 2. Association between UI and a task model.

end-user can see and understand how tasks are decomposed and 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.

Designer's point of view

This work will explore a method to provide designers with a technique to add Self-Explanation to 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 coexist at the same time into the same workspace inside the editor. 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 2. 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.

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 approach. A visual representation gives direct information about these relations because connections are explicitly represented in a visual render.

RELATED WORK

Two major areas are involved in our Self-Explanation approach, MDE and UI quality.

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 [11], oAW [11] and UsiXML [20] among others. A comparative analysis can be found in [10]. Semantic Networks have been also covered for UIs [9]. The Meta-UI concept was firstly proposed in [7] and deeply explored later in many other works. In [19], 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.

UIs Quality

Help systems have been extensively studied. One of the most relevant works is the Crystal application framework [15]. 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]. A set of ergonomic criteria for the evaluation of Human-Computer Interaction (HCI) can be found in [3].

WORK IN PROGRESS

First, the state of the art is still being covered. Second, we are studying the models of the UsiXML project in which this research is founded. Third, a prototype of the editor is being developed in order to show the concept in figure 2 with real UIs and study how to improve the explained method. However, this prototype does not support models at the moment and the designer must create the associated graphs by hand. Models will be integrated in the future. Finally, we are also exploring cognitive and knowledge engineering domains in order to let the user ask about the UI and how to deal with the possible answers.

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, analyzing the four levels of the MDE-compliant approach for developing UIs and the different models presented into the UsiXML specification and their relations. An editor is being developed to support designers at the design time. 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. J. Coutaz, L. Balme, G. Calvary, A. Demeure, and J.-S. Sottet. An MDE-SOA approach to support plastic user interafces in ambient spaces. In *Proc. HCI International 2007*, pages 152–171, 2007. Beijing, July 2007.

9. 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.

10. 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.

11. 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.

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. P. D. S. Paulo. User interface declarative models and development environments : A survey. 2007.

17. S. N. G. F. Sheshagiri, M. Using semantic web services for context-aware mobile applications , in proceedings of acm mobisys2004 workshop on context awareness, boston, massachusetts, usa, june 2004. 2004.

18. J.-S. Sottet, G. Calvary, J. Coutaz, and J.-M. Favre. A model-driven engineering approach for the usability of user interfaces. In *Proceedings of Engineering Interactive Systems 2007*, pages 140–157, 2007. Joint conference of IFIP WG2.7/13.4 10th Conference on Engineering Human Computer Interaction, IFIP WG 13.2 1st Conference on Human Centred Software Engineering, DSVIS - 14th Conference on Design Specification and Verification of Interactive System, University of Salamanca, Spain, March 22-24, 2007, J. Gulliksen et al. (eds), LNCS 4940.

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

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