Quality: a challenge for advanced user interfaces

Sophie Dupuy-Chessa
Univ. Grenoble Alpes, LIG
CNRS
41 rue des mathématiques
38400 Saint Martin d'Hères, France
Sophie.Dupuy@imag.fr

Gaëlle Calvary
Univ. Grenoble Alpes, LIG
CNRS
41 rue des mathématiques
38400 Saint Martin d'Hères, France
Gaelle.Calvary@imag.fr

ABSTRACT

In Human Computer Interaction, universal quality does not exist. Despite all the design efforts, there will always be users and situations the user interface (UI) will not be suitable for. This is particularly true for advanced UIs for which quality criteria are still ill-defined. This paper addresses the engineering of UIs from the end-user's point of view: it does not address the internal quality such as the software architecture. It reviews ways for integrating quality all over the development process and from different points of view: 1) the quality intended by designers thanks to flexibility and creativity in the design process as well as verification; 2) the quality perceived by end-users thanks to UI adaptation and self-explanation.

Keywords
Quality, Development process, Design time, Run time.

PROBLEM: THE MULTI FACES OF QUALITY

End-users often find problems while interacting with advanced user interfaces (UIs). Questions about where is an option on the mobile phone version of an application, what is the gesture to accomplish a task, or why did something happen in the UI naturally arise due to the imperfect quality of the UI.

This problem of insufficient quality can be due to the increasing difficulty of designing advanced UIs by adding parameters like the devices, the location, the user characteristics... As mentioned by [12], the difficulty to design systems has been moved up. Even if the designer intends to achieve a good quality level (intended quality), he/she cannot foresee all these problems and obstacles at design time because each single user has his/her own understanding of the UI and might be in specific situations. It becomes impossible to provide support for all the users at design time for all the situations they might be in.

But quality problems also exist because as the user is not the designer, the user has a different understanding of the UI. He/she can encounter different problems or obstacles during the interaction process, which can make him/her perceive a bad quality level.

These problems are graphically represented by a gulf (figure 1) between the intended versus perceived quality.

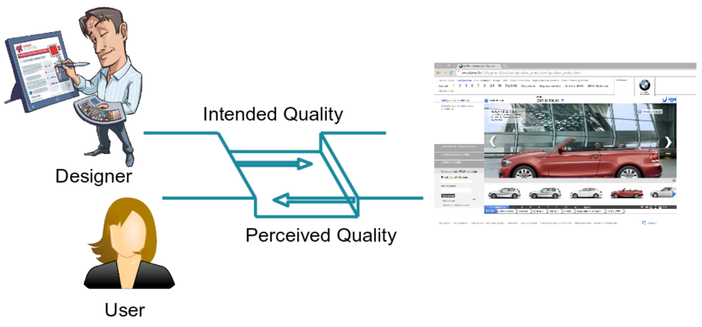


Fig. 1- Gulf of quality between intended and perceived quality.

Moreover perceived and intended quality is not static and can evolve with people, the context of use and the system itself. So the research question is:
What can be an engineering approach to continuously improve quality of advanced UIs?

MODELS: A UNIFYING APPROACH

As it is often the case in an engineering approach, we propose to use models as the basis for engineering quality. Models can serve as unifying approach that bridge the gap between the designer's and the end user's point of view:

- Models are created by developers along the development process. They aim at achieving the intended quality.
- Models created by developers can be reused to increase the quality perceived by end-users. They are "hyphens" between the intended and perceived qualities.

RESEARCH AXES: INTENDED AND PERCEIVED QUALITY

Considering the gulf of quality, we propose to structure the research axes thanks to designers' and end users' points of view. We need to study the intended and perceived quality.

Intended quality

Intended quality reflects the quality that the designer would like to achieve. It is mainly related to design time where design, but also quality verification activities usually take place.

Quality by design: model-based approaches have been investigated for long for generating UIs from models. However, the resulting quality was rather limited: the UIs were simply made of basic widgets (e.g., input fields, radio buttons), far from supporting the advanced features promoted in ambient intelligence. Recently, **creativity** has been explored by models [10]: the point is no more to generate UIs for end-users, but UIs for developers as means for supporting the divergence and convergence processes in creativity [9]. The UIs are generated from a task model using different models transformations. The developer selects the UIs or parts of the UIs he/she really appreciates. Then genetic mutations are processed to make the models transformations evolve and thus give rise to new UIs. Figure 2 presents examples of UIs automatically produced for a Chat application. The UI variations (structure, widget, layout, colour, etc.) are intended to inspire the developer.

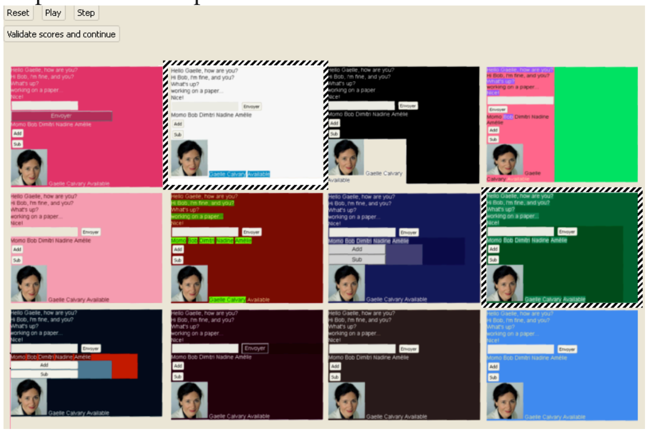


Fig. 2 – Examples of UIs generated by Magellan, a genetic algorithm-based environment for fostering developers' creativity

Another possible approach to improve the intended quality is to bring **flexibility for developers** so that to comply with the different practices in UIs development. Flexibility has been identified in the literature as one of the main research goals of method engineering [1]. For instance, [16] introduces flexibility in the design process for adaptive UI in order to decrease the threshold of use of models. We define three forms of flexibility:

1. Variability as the possibility of choosing one path in a set of equivalent variants. For example, instead of creating concrete user interfaces, they are generated from existing UIs, saving considerable part of the effort to be made for learning the CUI model and (re)modeling the UIs.
2. Granularability as the ability of a process model to support elements with different granularities, e.g. with different languages and/or quantities of details. We propose various levels of details when configuring and executing our tool for generating the domain model from a database. Expert designers just execute the tool whilst step by step explanations are provided for novice developers.
3. Completeness as the possibility of fulfilling or not the proposed process; some activities and/or artefacts are then optional or can be replaced by a predefined result or product. For instance, the activity "define the platforms model" can be avoided; in this case, the platform model can be replaced by "default" models that the developer picks up in a repository proposed by the process model.

Obviously, the UIs produced by such a flexible development process cannot be "perfect". However thanks to the process flexibility, designers and developers can reuse parts of their know-how and competencies, and are able to transfer some existing components into the paradigm of models: this makes it possible for them to create a first, albeit imperfect, version of their UIs, that they can iteratively improve, acquiring the needed competencies step by step.
But the solutions proposed are always limited; in particular they do not consider the enactment of the process, which is primordial when considering the need of rapid evolution when designing advanced UIs.

Quality by verification: Design should consider the verification of the intended quality. High quality of user interfaces, which can be ensured by several ways. For example, [13] proposes four ways of evaluation: *formally* by some analysis techniques, *automatically* by a computerized procedure, *empirically* by experiments with users and *heuristically* by simply looking at the UI and passing judgement according to one's own opinion. The automation of quality verification has been largely studied in the UI literature, however the usual techniques for UIs verification such as model checking [3] or testing [4] must be adapted to advanced UIs.

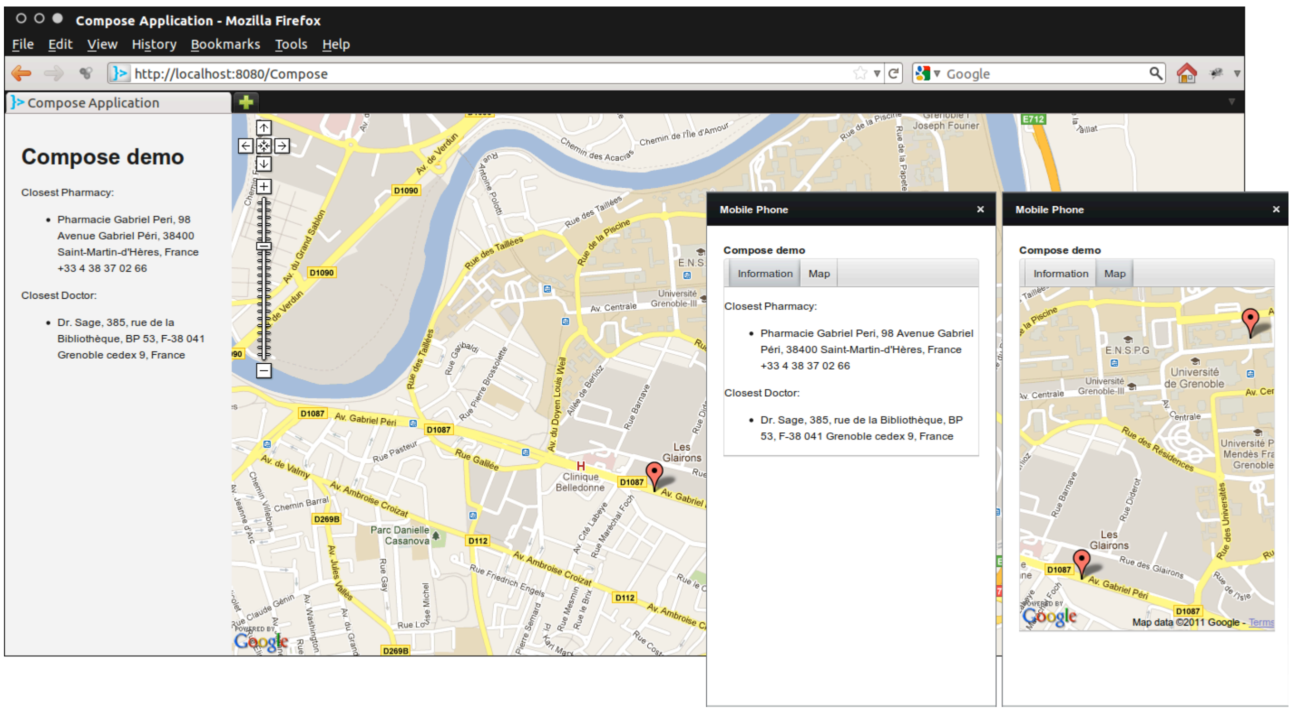
Perceived quality

Perceived quality corresponds to the end users' point of view under the system quality. So it is related to the runtime understanding of the system. At runtime, we consider that quality can be improved by composing existing UIs, by adapting UIs to their context of use or by explaining UIs.

Quality by reuse: Software composition is said to be one of the grand challenges for the coming years. In the engineering of human computer interaction, this means being capable of composing UIs from existing pieces of software. It has been addressed for different software development paradigms including models [11]. The problem is to succeed in composing without impairing the UI quality. As a matter of fact, composition can introduce some inconsistencies or discontinuities in the final UI.

Quality by adaptation: Plasticity refers to the capacity of UIs to withstand the variations of the context of use (user, platform, environment) while preserving user-centered properties [5]. User-centered properties clearly refer to the perceived quality. So to improve this perception, UIs can be technically adapted in two main manners: adaptation is either a remolding (e.g., replacing an image with a text) or a UI redistribution among the set of available platforms (e.g., migrating the inputs to a mobile device). Model driven adaptation has been intensively studied given rise to a reference framework [6]. To consider quality in this framework, usability is introduced: usability criteria complement transformations between models so as to choose an adaptation among others [14].

To implement this approach, [15] proposes UsiComp, an integrated and open framework which implements the principles of Cameleon by allowing designers to create models and to modify them at design time and at runtime. For instance, Figure 3 shows the two different UIs that are produced by UsiComp for two platforms, a PC and a mobile phone. In the background, we can see the UI adapted to the screen of the PC platform. Among others, the original screen from the PC platform has been split into two tabs due to the small resolution of the mobile phone screen. The zoom controller of the map widget has been removed as well. With such adaptations, UIs are adapted to devices thus providing usable UIs.


Compose Application - Mozilla Firefox
File Edit View History Bookmarks Tools Help
http://localhost:8080/Compose
Google
Compose Application
Mobile Phone
Information
Map
Mobile Phone
Information
Map

Compose demo

Closest Pharmacy:

- Pharmacie Gabriel Peri, 98 Avenue Gabriel Péri, 38400 Saint-Martin-d'Hères, France +33 4 38 37 02 66

Closest Doctor:

- Dr. Sage, 385, rue de la Bibliothèque, BP 53, F-38 041 Grenoble cedex 9, France

Figure 3. Generated UIs from the same task model. The UI in the background has been generated for a PC screen with higher resolution than the UI for the mobile phone in front of the figure.

However it still remains a challenge to guarantee that an adaptation has not impaired the perceived quality. Moreover adaptation generally does not take into account post-WIMP UIs.

Quality by repair: As universal quality is utopian, end-users are those who are the best to improve their UIs as soon as they grasp the purpose and design rationale of each UI element. So an interesting approach is to support end-users reprogramming thanks to models [7]. This approach is not restricted to repair, it can also be used for design. In this case, it introduces new problems in quality by design as end-users become designers.

A complementary approach can propose to provide help about the UIs thanks to models: models created at design time can be used at runtime to explain UIs. In particular, these self explanatory UIs can provide the end-users with the design rationale of the UI [8].

For instance, [8] presents a system that consists of using the design models to compute questions and answers at runtime to provide an help system (Figure 4). The design models are still those proposed by the Cameleon reference framework.

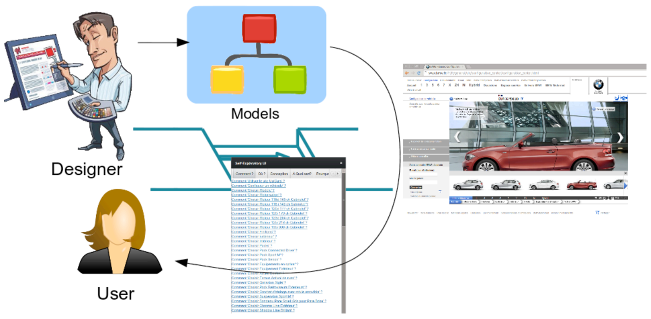


Figure 4. *Reducing the gap between intended quality and perceived quality by model-based explanations.*

The self-explanatory facilities generated with our approach are responsible for:

- Generating the set of questions. We consider those questions that the help system "knows" how to answer by inspecting the underlying models of the UI.
- Generating answers. Once the user asks a question to the help system, the system needs to compute an understandable explanation or answer. This is done through the following three steps:
    - Selecting the Explanation Strategy. In this phase the help system selects the explanation strategy according to the type of the question. For instance, a question about "how" to realize a task (e.g. how to choose packs when selecting a car) is associated to a strategy related to the task model.
    - Inspecting the models. Each explanation strategy inspects one or more models to retrieve the elements that have been defined for each strategy. For instance, to answer a "how" question, the strategy starts by inspecting the task model. The task related to the question is identified in the task model. Then the elements of the abstract UI mapped to the tasks are identified. Finally the elements of the concrete UI corresponding to the elements of the abstract UI are selected. Thanks to this chain of mapping, it is possible to obtain the final elements in the UI that can provide help. For instance, for the question "how to choose packs", the mapping between models allows the system to retrieve the « Packs » button in the UI.
    - Composing the answer. Once all the elements of the models have been retrieved, the answer is composed and prepared to be presented.
- Presenting the answer. The computed answer is provided to the user in an understandable way. For example, the system will propose to users to "Use the Packs button" if they want to choose a pack.

We conducted an experiment to evaluate the added value of model-based self-explanations. It shows that most of the users identify the help system as useful, in particular the How and the Where questions. However the study has also collected some interesting suggestions. First, we identified other types of questions (what is…, what if…) not explicitly supported by our system. Secondly usability of the help system, that was not our main concern, needs to be improved to facilitate interaction to select questions and to guide users thanks to the answer.

However they are limited to graphical UIs and there is a clear need to increase models so as to provide explanations about advanced UIs.

CONCLUSION

The multi-faces quality of advanced UIs requires continuous amelioration. This challenge motivates needs for co-evolution between actors and systems: roles are unified between end users and developers; the gap between design, run and evaluation times are erased so as to improve quality at any time.

REFERENCES

1. Harmsen F., Brinkkemper S., and Oei J., Situational method engineering for informational system project approaches, in *Methods and Associated Tools for the Information Systems Life Cycle*, 1994, p. 169- 194.

2. Nielsen, J., and Molich, R. Heuristic evaluation of user interfaces. In *Proceedings of the SIGCHI conference on Human factors in computing systems*, ACM (1990), 249–256.
3. Paternò, F., and Santoro, C. Support for reasoning about interactive systems through human-computer interaction designers' representations. *Comput. J. 46*, 4 (2003), 340–357.
4. Hjort, U. H., Illum, J., Larsen, K. G., Petersen, M. A., and Skou, A. Model-based gui testing using Uppaal at Novo Nordisk. In *FM 2009: Formal Methods*. Springer, 2009, 814–818
5. Thévenin D. and Coutaz J., Plasticity of user interfaces: Framework and research agenda ; in Proceedings of Interact'99, A. Sasse & C. Johnson (réds), IFIP IOS Press Publ., 1999, 110–117.
6. Calvary, G., Coutaz J., Thevenin, D., Limbourg, Q., Bouillon, L., Vanderdonckt, J. A Unifying Reference Framework for Multi-Target User Interfaces, Interacting With Computers, Vol. 15/3, 2003, 289-308.
7. Sottet, J.-S., Calvary, G., Favre, J.-M., and Coutaz, J. Megamodeling and metamodel-driven engineering for plastic user interfaces: Mega-ui. In *Human-Centered Software Engineering*. 2009, 173–200.
8. Garcia Frey, A., Calvary, G., Dupuy-Chessa, S., and Mandran, N. (2013). Model-based self-explanatory UIs for free, but are they valuable? In *Proceedings of the 14th IFIP TC13 Conference on Human-Computer Interaction (INTERACT'13), 2-6 September 2013, Cape Town, South Africa*. Springer.
9. Buxton, B. Sketching User Experiences: Getting the Design Right and the Right Design, Morgan Kaufmann Publishers Inc. San Francisco, CA, USA, 2007
10. Masson, D., Demeure, A., **Calvary**, G. Magellan, an Evolutionary System to Foster User Interface Design Creativity, Proceedings of the second ACM SIGCHI Symposium on Engineering Interactive Computing Systems (EICS 2010), ACM Press, Berlin, pp 87-92
11. Gabillon, Y., Petit, M., **Calvary**, G., Fiorino, H. Automated planning for User Interface Composition, 2nd SEMAIS Workshop, IUI conférence, Feb. 2011, Palo Alto, USA
12. Palen, L. Beyond the Handset: Designing for Wireless Communications usability, ACM Transactions on Computer-Human Interaction, 9(2), pp. 125-151, june 2002.
13. Nielsen, J., and Molich, R. Heuristic evaluation of user interfaces. In *Proceedings of the SIGCHI conference on Human factors in computing systems*, ACM (1990), 249–256.
14. Sottet, J.-S., Calvary, G., Coutaz, J., and Favre, J.-M. A model-driven engineering approach for the usability of plastic user interfaces. In *Proc. of EIS '07*, Springer-Verlag, 2008, 140–157.
15. Garcia-Frey A., Ceret E., Dupuy-Chessa S., Calvary G., and Gabillon Y. UsiComp: an extensible model- driven composer. EICS, (2012), 263–268.
16. Céret E., Calvary G., Dupuy-Chessa S., Flexibility in MDE for scaling up from simple applications to real case studies: illustration on a Nuclear Power Plant, 25ème conférence francophone sur l'Interaction Homme-Machine, IHM'13, Bordeaux, France, 2013.