A Development Process for Plastic User Interfaces

David Thevenin, Gaëlle Calvary, Joëlle Coutaz,
CLIPS-IMAG, BP 53, 38041 Grenoble Cedex 9
http://iihm.imag.fr
{ David.Thevenin, Gaelle.Calvary, Joelle.Coutaz}@imag.fr

1.Introduction

The increasing proliferation of computational devices has introduced the need for applications to run on multiple platforms in different physical environments. Tools such as UIML [Abrams99], and mechanisms based on the XML technology are technical attempts to address the problem. Although useful, they do not necessarily provide sufficient insight about the process by which adaptation may be performed. We propose a unifying framework that structures and helps understanding the development process of plastic user interfaces.

2. Plasticity

The term *plasticity* is inspired from the property of materials that expand and contract under natural constraints without breaking, thus preserving *continuous usage*. Applied to HCI, plasticity is the "capacity of an interactive system to *withstand variations* of *context of use* while *preserving usability*". By context of use, we mean two classes of physical entities:

- The physical and software platform(s), that is, the computational device(s) used for interacting with the system.
- The physical environment where the interaction takes place.

A *platform* is modeled in terms of resources which, in turn, determine the way information is computed, transmitted, rendered, and manipulated by users. The *environment* covers "the set of objects, persons and events that are peripheral to the current task(s) but that may have an impact on the system and/or the user's behavior, either now or in the future" [Thevenin99]. According to this definition, an environment may encompass the entire world. In practice, the boundary is set up by domain analysts whose role is to elicit the entities that are relevant to the case at hand. These include observation of users' practice as well as consideration for technical constraints. For example, surrounding noise should be considered in relation to sonic feedback. Lighting condition is an issue when it may influence the robustness of a computer vision-based tracking system. User's location provides context for information relevance: Tasks that are central in the office (e.g., writing a paper) may become secondary, or even irrelevant, in a train.

As any evolutive phenomenon, plastic adaptation is structured as a five step process:

- detection of the conditions for adaptation (here, variations in the context of use),
- identification of candidate user interfaces appropriate to the new context of use,
- selection of a user interface,
- transition from the current user interface to the newly selected solution (transition is a widely ignored issue!),

- execution of the new user interface until next conditions for adaptation occur.

Each of the five steps involved in the plasticity process (detection, identification, selection, transition, execution) is performed either by the system, or by the user, or as a mixture of both. At the two extremes,

- The system is able to handle the five steps without human intervention. In this case, the system is capable of *adaptative plasticity*.
- The user performs the five steps manually. The system supports *adaptable plasticity*.
- *Mixed plasticity* covers a combination of both human and system intervention.

Our framework covers all of these conditions.

3. The Framework

As shown in Figure 1, the framework:

- Builds upon known models such as the domain model and the tasks model, but improves them to accommodate the variation of context of use;
- Explicitly introduces new models and heuristics that have been overlooked or ignored so far to express different contexts of use: the platform, the interactors, the environment and the evolution models.

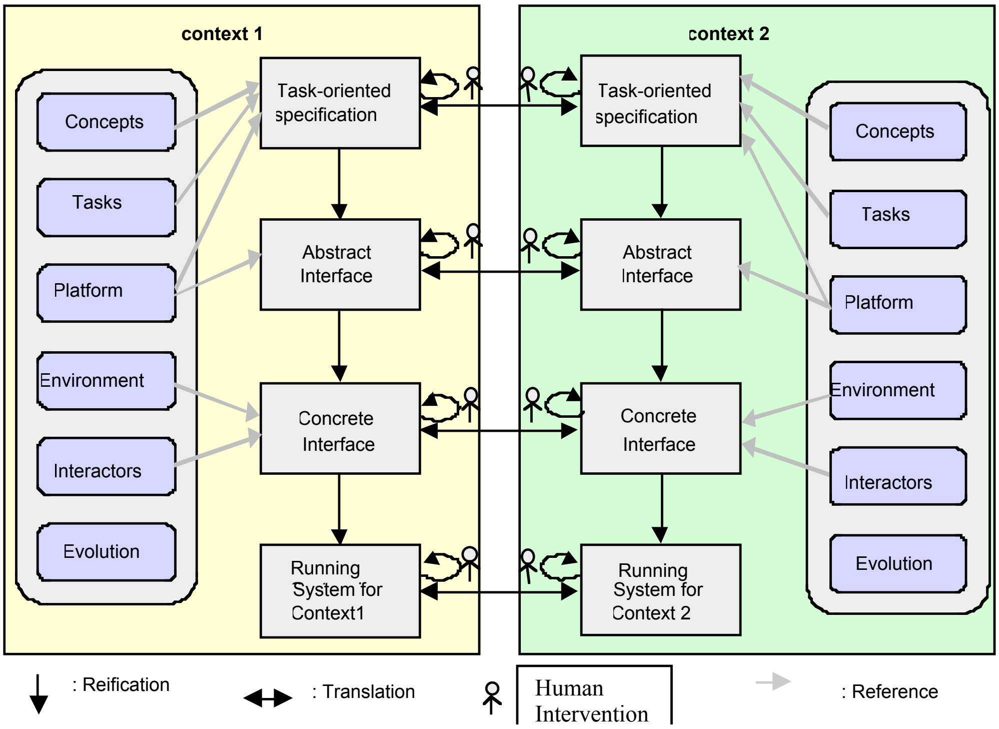

Fig.1. The reference development process for supporting plastic user interfaces. The picture shows the process when applied to two distinct contexts: context1 and context2.

The *Evolution model* specifies the change of state within a context as well as the conditions for entering and leaving a particular context. An interesting issue is to identify the boundary of a context! The *Interactors Model* describes "resource sensitive multimodal widgets" available for producing the concrete interface. Widgets may be functionally equivalent but may have very different costs (e.g., computational costs but also cognitive, conative, and physical costs for the user). The concepts model, the task model, the platform model, the environment model, the interactors model and the evolution model are the given.

The process is a combination of *vertical reification* and *horizontal translation*. Vertical reification covers the derivation process, from top level abstract models to run time implementation. Horizontal derivations, such as those performed between HTML and WML content descriptions, correspond to translations between models at the same level of reification. Reification and translation may be performed automatically from specifications, or manually by human experts. Because automatic generation of user interfaces has not found wide acceptance in the past [Myers00], our framework makes possible manual reifications and translations. Such operations are manual when the tools at hand cannot preserve the usability criteria set up for the particular system or when they simply do not exist.

Reification and translation may be combined in multiple ways giving rise to multiple instantiations of the reference framework.

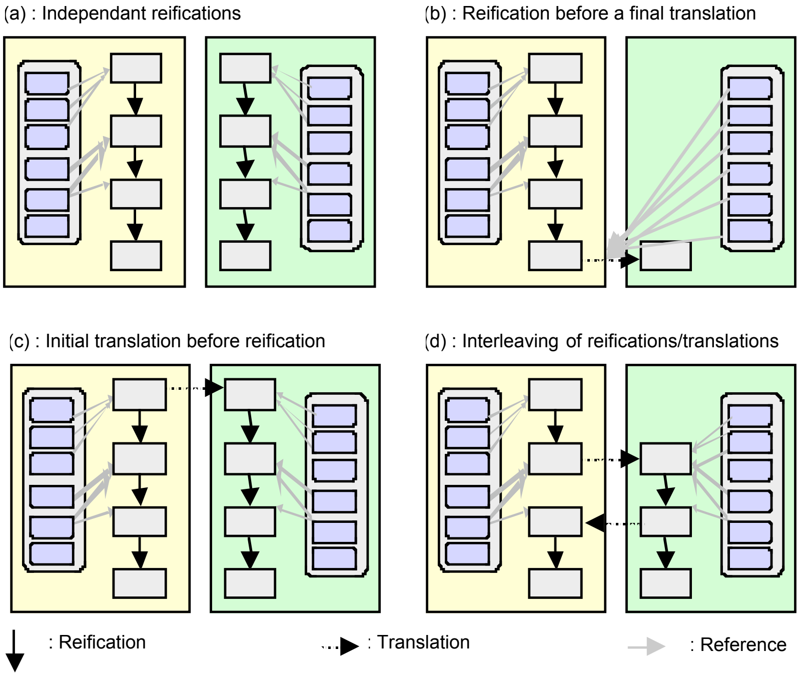
(a) : Independant reifications
(b) : Reification before a final translation
(c) : Initial translation before reification
(d) : Interleaving of reifications/translations
: Reification
: Translation
: Reference

Fig.2. Instantiations of the reference framework.

4. Instantiation of the Framework

As shown in Figure 2, the reference framework can be instantiated in many ways:

- In 2a), two running user interfaces are reified in parallel where the initial models are specified for each context of use. This configuration, which depicts current practice, forces to maintain consistency between the multiple versions.
- 2b) corresponds to the ideal situation: reification is used until the very last step. Consistency maintenance is here minimal. This approach has been used for the development of the user interface of a Heating Control System for Java-enabled target platforms ranging from a PC platform to a PDA. These interfaces have been derived automatically using ARTStudio, a tool developed along the lines of our framework [Calvary01].
- In 2c), the task-oriented specification is translated to fit another context. From there, reifications are performed in parallel. This approach has been adopted for the Heating Control System for WAP mobile phones. Sub trees that correspond to infrequent tasks have been pruned from the original task tree developed for the Java-enabled platforms.

Because ARTStudio does not support Web-based techniques yet, the reification steps have been done manually by a human expert.

- 2d) shows a mix of interleaving between reification and translation.

4. Conclusion

Although the prospective development of interactive systems may be fun and valuable in the short run, we consider that the principles and theories developed for the desktop computer should not be put aside [Eisenstein00]. Instead, our reply to the technological push is to use current knowledge as a sound basis, question current results, improve them, and invent new principles if necessary. This is the approach we have adopted for supporting plasticity by considering model-based techniques from the start [Paterno99, Szekely 96]. These techniques have been revised and extended to comply with a structuring reference framework. In particular, we think that the combination of reification and translation is an interesting issue to discuss in the light of current practice and tools. Given a particular system and development context, what is the right mix? When is reification appropriate, when is translation appropriate?

5. References

[Abrams 99] M. Abrams, C. Phanouriou, A.L. Batongbacal, S. Williams, and J. Shuster. UIML: An Appliance-Independent XML User Interface Language. In Proceedings of WWW8, Toronto, May 11-14, 1999.

[Calvary 01] A Unifying Reference Framework for the Development of Plastic User Interfaces. Engineering Human Computer Interaction, EHCI2001,To Appear

[Eisenstein 00] Eisenstein J., Vanderdonckt, J. Puerta, A. Adapting to Mobile Contexts with User-Interfaec Modeling. Proc. of 3rd IEEE Workshop on Mobile Computing Systems and Applications WMCSA 2000 (Monterey, December 7-8, 2000), IEEE Press, Los Alamitos, 2000.

[Myers 00] B. Myers, S. Hudson, R. Pausch. Past, Present, Future of User Interface Tools. Transactions on Computer-Human Interaction, ACM, 7(1), March 2000, pp. 3-28.

[Paterno 99] F. Paternò. Model-based Design and Evaluation of Interactive Applications, Springer Verlag, November 1999.

[Szekely 96] P. Szekely. Retrospective and Challenges for Model-Based Interface Development, Computer-Aided Design of User Interfaces. In *Proceedings of CADUI'96*, J. Vanderdonckt (eds), Presses Universitaires de Namur, 1996.

[Thevenin 99] D. Thevenin, J. Coutaz. Plasticity of User Interfaces: Framework and Research Agenda. In Proc. Interact99, Edinburgh, A. Sasse & C. Johnson Eds, IFIP IOS Press Publ.. , 1999, pp.110-117.