

# UML for Interactive Systems: What is Missing

*Philippe Palanque*

*Rémi Bastide*

LIHS-IRIT, Université Paul Sabatier  
118, route de Narbonne,  
31062 Toulouse Cedex, France  
palanque@irit.fr

LIHS-IRIT, Université Toulouse 1  
Place Anatole France,  
31042 Toulouse Cedex, France  
bastide@irit.fr

## **Abstract**

This position paper explores several design approaches put forward by HCI research, and investigates how the UML could or should support them.

## **1 Introduction**

Over the years, the HCI research community has developed a set of techniques and approaches to the design of interactive systems. This trend of research puts the user at the center of the design process, and regards usability as the ultimate goal of design.

Meanwhile, the Software Engineering community was developing a set of notations and processes to support software design. The software system itself is at the center of this research, and software quality (dependability, reusability, etc.) is its ultimate goal. This research has matured into the UML [3], which is now an accepted standard.

It is unclear how these two trends of research can complement each other in order to keep the best of both worlds. This position paper explores this issue by looking at several design philosophies from the HCI side, and how much support is found for them in the UML (if any).

## **2 Engineering Interactive Systems**

Several design philosophies emerge from HCI research, and can be classified according to the artefact that is put in the forefront:

- Abstraction first : architectures at the forefront
- Semantic first : metaphors at the forefront
- Implementation first : toolkits at the forefront
- Process first : User Centred Design
- Model first : Model-Based approaches

Each of these design philosophies is detailed below, along with its relationship to the UML

### **2.1 Abstraction first: architectures**

The well known Seeheim and Arch [1] models are the result of an in-depth reflection on the very nature of interactive systems. They describe the typical structure of an IS in terms of abstract functionalities to be covered (e.g. Presentation, or Functional Core Interface).

These two models can be considered as an abstract software model of an Interactive System. They provide a very useful framework for the software designer to work in, since they provide a guide for a clear separation of concerns.

What kind of support can be found in the UML to support this architectural view of IS? A naïve idea is to use the notion of “package”, provided by the UML as well as by most Object-Oriented programming languages. For instance, the topmost decomposition level of the software could mimic the Seeheim model by featuring four packages, Presentation, Dialogue, Functional Core Interface, and Functional core. This strategy, however, seems rather naïve since the “Dialogue” component of Seeheim cannot usually be conveniently isolated into a single package, but is usually distributed over several software components. Fined-grained dialogue (such as the answer to mouse click and key presses) is usually embedded into the code of widgets that are part of the “Presentation” toolkit. Coarse-grained dialogue is defined as the business logic of the application, usually in objects from the “Functional core” part.

Another approach to support the architectural view of IS would be a systematic use of the extension mechanisms of the UML such as stereotypes of tagged values.

## **2.2 Semantic first: metaphors**

Designers often favour the use of metaphors to provide a unifying vision to their software artefact. The desktop or room metaphor has extensively been used for file managers, the “Village” metaphors is used for web sites or interactive TV, etc.

It is unclear how the UML could (or should) support the use of metaphors in the design. Maybe some support could be provided within the RUP to this end, but this requires further study.

## **2.3 Implementation first: toolkits**

The HCI community has devoted extensive work and research on the design and implementation of UI toolkits, aiming at making the life of the software designer easier.

Although it would seem that UML is ideally suited to this kind of work, it is surprising to notice that no widely-used toolkit is described in terms of the UML (beyond providing an inheritance tree), while a UML description of a toolkit would be very useful for the software designer, especially interaction diagrams and StateCharts of the widget set. We can guess that 1) UML has not been used when designing these toolkits and 2) UML has been considered to costly for providing the documentation.

Has a result, most UI toolkits are hard to master and poorly documented, providing no or few rationale on their design.

## 2.4 Process first: iterative, UCD process

HCI people promote user-centred, iterative processes based on early evaluation of low-fidelity prototypes, incrementally evolved towards higher-fidelity prototypes. The UML users' guide Software development life cycle p. 33-34 states that "the UML is largely process-independent [...]. However, [...] you should consider a process that is 1) Use case driven, 2) Architecture-centric 3) Iterative and incremental". Prototyping per-se is not covered in the RUP. Proponents of eXtreme Programming (XP) also promote light-weight, incremental design, but the focus is on the incremental production and test of software, and the users are not usually formally involved in XP.

## 2.5 Model first: model-based approaches

Many models can be considered for model-based UI Design [4], [2]:

- Domain model (including data model)
- Task model and scenarios
- User model
- Platform model (link to toolkit and environment)
- Dialogue model (behaviour of the application I/O)
- Presentation model (appearance of the application)
- Application model (commands and data the application provides)
- ...

Several of these models are supported in the UML:

- Domain model (including data model) are supported as class and object diagrams (organisational charts, ...)
- Application model (commands and data the application provides) are the main focus of UML

Some models are only partially accounted for:

- Task model and scenarios can be described informally in UML use cases (Use Cases diagrams describe the **relationships** between use cases)
- Dialogue model state charts (state and events) and sequence diagrams

Several models are not considered at all in the UML

- User model
- Platform model (link to the toolkit and )
- Presentation model (appearance of the application)

## 3 Conclusions and Perspectives

For the team of methodologists (Rumbaugh, Jacobson, Booch) that shaped the UML, User Centred Design was not a central concern.

However it is clear that, in the present landscape of software development, Non-Interactive Systems are a special rare kind of system and not the other way round.

Our claim is that further work is needed to merge the achievement of HCI research into mainstream UML-based software design methodologies, and that this would be beneficial to the software industry as a whole.

## 4 References

1. Bass, Len, Little, R., Pellegrino, R., Reed, S., Seacord, R., Sheppard, S., and Szezur, M. R. "The Arch Model: Seeheim Revisited." *User Interface Developers' Workshop*. Version 1.0 (1991)
2. Bodart, François, Hennebert, A-M., Leheureux J-M., Provot, I., and Vanderdonckt, Jean. "A Model-Based Approach to Presentation: A Continuum From Task Analysis to Prototype." *1<sup>st</sup> Eurographics Workshop on Design Specification and Verification of Interactive System (DSV-IS'94)*, Carrara, Italy. (1994)
3. Rational Software Corporation. *UML Summary*. 1.1 ed.1997.
4. Wilson, S., Johnson, P., Kelly, C., Cunningham, J., and Markopoulos, P. "Beyond Hacking: a Model Based Approach to User Interface Design." *HCI'93*, Loughborough, U.K. Cambridge University Press (1993) 217-31.