Position paper for the End-User Software Engineering Dagstuhl Workshop (Feb. 2007)

Meta-User Interfaces for Ambient Spaces: Can Model-Driven-Engineering Help?

Joëlle Coutaz
Université Joseph Fourier, Lab.Informatique de Grenoble (LIG)
385 rue de la Bibliothèque, BP 53, 38041 Grenoble Cedex 9, France
joelle.coutaz@imag.fr

PERSONAL WORK RELEVANT TO THE WORKSHOP

My goal is to develop concepts and techniques that allow users to control and understand the ambient interactive spaces in which they live. With ambient computing, we are shifting from the control (and understanding) of systems and applications confined to a single computer to that of a dynamic computational aura where the boundaries between the physical and the digital worlds are progressively disappearing, where everything is highly dynamic and adaptive.

As a result, the pre-packaged well-understood solutions provided by shells and desktops that allow end-users to control their computing environments are inadequate for a continuous moving universe. To address this problem, I propose the concept of *meta-UI*. In addition, user interfaces that used to be defined once for ever for a well-identified context of use, must evolve dynamically. In my research group, we are addressing this problem under the umbrella of *UI plasticity*. Our approach to UI plasticity brings together MDE (Model Driven Engineering) and SOA (Service Oriented Architecture) within a unified framework that covers both the development stage and the runtime phase of interactive systems.

META-UI

A meta-UI is a special kind of end-user development environment whose set of functions is necessary and sufficient to control and evaluate the state of an interactive ambient space. This set is *meta-* because it serves as an umbrella *beyond* the domain-dependent services that support human activities in this space. It is *UI*-oriented because its role is to allow users to control and evaluate the state of the ambient interactive space. By analogy, a meta-UI is to ambient computing what desktops and shells are to conventional workstations.

As shown in Fig. 1, a meta-UI is characterized by its *functional coverage* in terms of *services* such as object discovery and coupling, and *object types*. *Objects discovery* allows users (and the system) to be aware of the objects that can be coupled. By coupling objects, users (and the system) build new constructs whose components play a set of roles (or functions). In conventional computing, roles are generally predefined. In ambient computing, where serendipity is paramount, *assigning roles to objects* becomes crucial. For example, Bob and Jane meeting in a café use spoons and lumps of sugar to denote the streets and buildings of the city they are talking about. Bob couples a spoon with the table by laying it down on the table while uttering "this is Champs-Elysées". The system can then discover the presence of the spoon and assign it the role of interaction resource (phicon). By doing so, Bob has dynamically defined a *mixed-by-contruction object*.

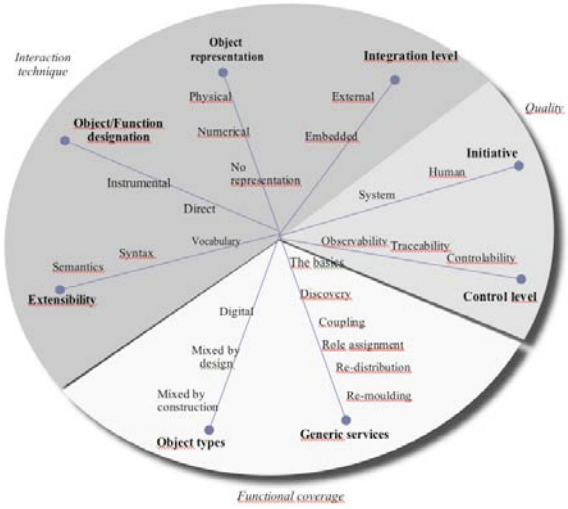

Fig. 1. A dimension space for meta-UI's.

UI re-distribution is another important generic service to be provided in ambient spaces. It denotes the re-allocation of UI elements of the interactive space to different interaction resources. For example, the GUI of a web site may dynamically switch from a centralized rendering on a PC screen to a distributed UI between a PDA and a wall-mounted display. In turn, UI re-distribution may require UI *re-moulding*, that is the capacity of the UI to reconfigure itself or to be reconfigured (under end-user's control) by suppressing, adding, and/or re-organizing UI elements.

Dagstuhl Seminar Proceedings 07081
End-User Software Engineering
http://drops.dagstuhl.de/opus/volltexte/2007/1082

Services and objects are invoked and referenced by the way of an *interaction technique* (i.e. a UI) that provides users with some *level of control* (observability only, traceability over time, and controllability or programmability). An interaction technique is a language (possibly *extensible*) characterized by the *representation* (vocabulary) used to denote objects and functions as well as by the way users construct sentences and assemble them into programs (including how they select/*designate* objects and functions).

Given the role of a meta-UI, the elements of the interaction technique of the meta-UI cohabit with the UI's of the domain-dependent services that it governs. The *integration level* expresses this relationship: all or parts of the UI elements of the meta-UI are *embedded* with (or weaved into) the UI components of the domain-dependent services. For example, Collapse-to-zoom uses the weaving approach. Alternatively, UI elements of the meta-UI services may be *external*, i.e. not mixed with the UI components of the domain-dependent services.

MDE and SOA

MDE aims at integrating different technological spaces using models, models transformations and mappings as key mechanisms. SOA defines the appropriate meta-model for a particular class of models: the runtime components. The flexibility offered by SOA fits our requirements for dynamic UI re-distribution and UI re-molding.

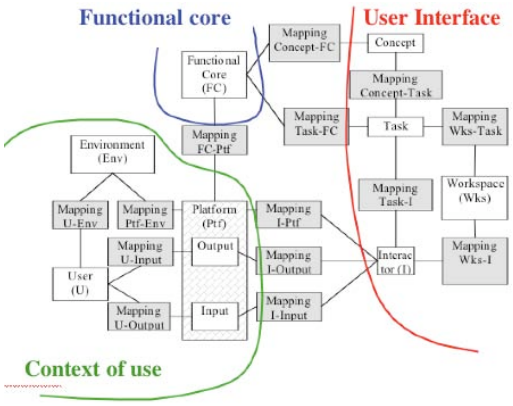

Fig. 2 An interactive system is a graph of models related by mappings and transformations.

As shown in Fig. 2, an interactive system is a graph of models that expresses and maintains multiple perspectives on the system. As opposed to previous work, an interactive system is not limited to a set of linked pieces of code. Models developed at design-time, which convey high-level design decision, are still available at runtime. A UI may include a task model, a concept model, an Abstract UI model (expressed in terms of workspaces), and a Concrete UI model (expressed in terms of interactors) all of them linked by mappings. Tasks and Concepts are mapped to entities of the Functional Core of the interactive system, whereas the Concrete UI interactors are mapped to I/O devices (interaction resources) of the platform. Mappings between interactors and I/O devices support the explicit expression of centralized versus distributed UIs.

Transformations and Mappings are models as well expressed in ATL (QVT could be an option as well). In the conventional model-driven approach to UI generation, transformation rules are diluted within the tool. Model transformers are encapsulated as services within a middleware infrastructure that includes services to support context awareness, UI re-moulding and UI re-distribution: The *situation synthesizer* computes the current situation from the information provided by observers. An *evolution engine* elaborates a reaction in response to the new situation. For example, "if a new PDA arrives, move the control panel to the PDA". The evolution engine identifies the components of the UI that must be replaced and/or suppressed and provides the configurator with a plan of actions. The *Configurator* executes the plan. If new components are needed, these are retrieved from the *storage space* by the *component manager*. Components of the storage space are described with conceptual graphs and retrieved with requests expressed with conceptual graphs. By exploiting component reflexivity, the configurator stops the execution of the "defectuous" components specified in the plan, gets their state, then suppresses or replaces them with the retrieved components and launches these components based on the saved state of the previous components. The components referred to in the action plan do not necessarily exist as executable code. They may instead be high-level descriptions such as task models. If so, the configurator relies on *models transformers* to produce executable code.

We are currently experimenting the flexibility provided by the interplay between modeling an interactive system as a graph of models, the existence of a meta-UI and of UI transformers encapsulated as OSGi services. In our example of a Home Control Heating System (HHCS), the user's task is to set the temperature of the rooms of the home. The meta-UI provides the end-user with access to the task and the platform models. For example, the platform model indicates that a PC HTML and a PC XUL are currently available in the home. By selecting a task of the task model then selecting the platform(s) on which the user would appreciate to perform the selected task, the UI is re-computed and redistributed on the fly.

ISSUES TO BE DISCUSSED

Programming (and debugging) ambient spaces is yet another challenge. Embracing this challenge as a whole may be too complex. Shall we study it based on a classification of ambient spaces (e.g., domestic, public, mobile settings, a day of "my" life, etc.). By extension, what is the problem space of EUSE? How does current approaches cover the problem space? And then, what is the solution space?

REFERENCES

In addition to the classics (A. Cypher, B. Myers, H. Lieberman, etc.), I would like to suggest the following ref. related to ambient spaces as well as to our own work on UI plasticity and meta-UI.

1. L. Balme, A. Demeure, N. Barralon, J. Coutaz, G. Calvary. CAMELEON-RT: a Software Architecture Reference Model for Distributed, Migratable, and Plastic User Interfaces. In Proc. *second European Symposium on Ambient Intelligence*, EUSAI 2004, LNCS 3295, Markopoulos et al. pp. 291-302

2. G. Calvary, J. Coutaz, D. Thevenin, Q. Limbourg, L. Bouillon, J. Vanderdonckt. A Unifying Reference Framework for Multi-Target user interfaces. *Interacting with Computers*, Special Issue on Computer-Aided Design of User Interface, 15(3), Elsevier Publ., June 2003, pp. 289-308.

3. Coutaz J. Meta-User Interface for Ambient Spaces, Invited talk. In proceedingd *TAMODIA'06*, Hasselt, Belgium, October 2006, Springer LNCS publ., p. 1-15.

4. Dey, A., Hamid, R., Beckmann, C., Li, Y., Hsu, D., a CAPpella: Programming by Demonstration of Context-Aware Applications, In *Proceedings of the ACM SIGCHI'04*, Vienne, 33-40.

5. Sohn, T.Y., Dey, A.K., iCAP: An Informal Tool for Interactive Prototyping of Context-Aware Applications. In *Proceedings of the International Conference on Pervasive Computing 2006*. Dublin, Ireland, May 2006, 974–975.