Adaptation and Plasticity of User Interfaces

David Thévenin, Joëlle Coutaz

CLIPS-IMAG
BP 53, 38041 Grenoble Cedex 9, France
Tel. (33) 4 76 41 91 57, Fax (33) 4 76 44 66 75
http://iihm.imag.fr/coutaz, email : joelle.coutaz@imag.fr

ABSTRACT. This paper introduces the notion of plasticity, a new property of interactive systems that denotes a particular type of user interface adaptation. It also presents a generic framework inspired from the model-based approach, for supporting the development of plastic user interfaces. This framework is illustrated with simple case studies.

KEYSWORDS. User interface adaptation, plasticity.

1. Introduction: A Design Space for Adaptation

In HCI, adaptation is modeled as two complementary system properties: adaptability and adaptivity. Adaptability is the capacity of the system to allow users to customize their system from a predefined set of parameters. Adaptivity is the capacity of the system to perform adaptation automatically without deliberate action from the user's part. Whether adaptation is performed on human requests or automatically, the design space for adaptation includes three additional orthogonal axes (see Figure 1):

- *The target for adaptation*. This axis denotes the entities for which adaptation is intended: adaptation to users, adaptation to the environment, and adaptation to the physical characteristics of the system. The physical characteristics of a system can be refined in terms of interactional devices (e.g., mouse, keyboard, screen, video cameras), computational facilities (e.g., memory and processing power), and communicational facilities (e.g., bandwidth rate of the communication channels with other computing facilities).
- *The means of adaptation*. This axis denotes the components of the system involved in adaptation: typically, the system task model, the rendering techniques, and the help subsystems, may be modified to adapt to the targeted entities. The system task model is the system implementation of the user task model specified by human factor specialists. The rendering techniques denote the observable presentation and behavior of the system. The help subsystems include help about the system and help about the task domain.
- *The temporal dimension of adaptation*. Adaptation may be static (i.e., effective between sessions) and/or dynamic (i.e., occurring at run time).

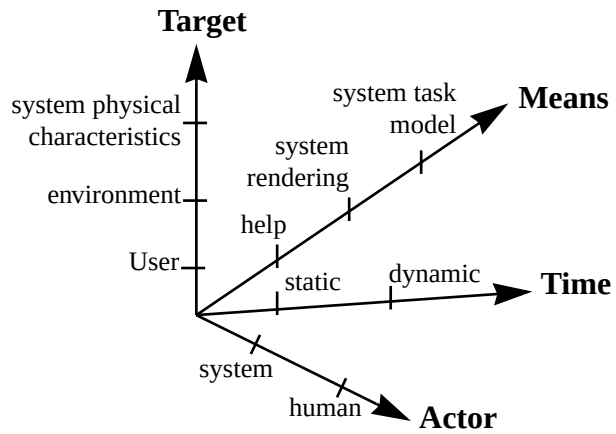

Figure 1: A design space for adaptation at a high level of reasoning.

2. Plasticity of User Interfaces

The term "plasticity" is inspired from the property of materials that expand and contract under natural constraints without breaking, thus preserving continuous usage. By analogy, plasticity is the capacity of a user interface to withstand variations of both the system physical characteristics and the environment while preserving usability. In addition, a plastic user interface is specified once to serve multiple sources of variations, thus minimizing development and maintenance costs. Plasticity may be static and/or dynamic. It may be achieved automatically and/or manually.

Within the design space of adaptation presented in Figure 1, plasticity is characterized in the following way:

- Along the target axis, plasticity is concerned with the variations of the system physical characteristics and/or the environment. It does not, therefore, cover adaptation to users' variations;
- Along the means axis, plasticity involves the modification of the system task model and/or of the rendering techniques;
- The temporal and automaticity axis are left opened.

Technically, plasticity requires software portability. However, platform independent code execution is not a sufficient condition. Virtual toolkits, such as the Java abstract machine, offer very limited mechanisms for the automatic reconfiguration of a user interface in response to variations of interactional devices. All of the current tools for developing user interfaces embed an implicit model of a single class of target computers (typically, a keyboard, a mouse and at least a 640x480 color screen). As a result, the rendering and responsiveness of a Java applet may be satisfactory on the developer's workstation but not necessarily usable for a remote Internet user. Experience shows that portability does not guarantee usability continuity. In addition, the iterative nature of the user interface development process, as well as code maintenance, make it difficult to maintain consistency between the multiple target versions.

In the absence of appropriate software tools, we need a framework for reasoning about the design and development of plastic user interfaces.

3. A Framework for Supporting Plasticity

Going one step further than the slogan for portability (i.e., "write it once and run it multiple times"), a framework for supporting plasticity should allow developers to specify once and produce usable multiple times.

Model-based user interface generators, which promote the specification of high level models, provide an approach consistent with our requirements. Our framework, inspired from such generators, is illustrated in Figure 2 [Thevenin 98].

- A System Task Model is specified from the description of the User Task Model. The User Task Model results from the activity analysis performed by human factor specialists. It informs the System Task Model that describes the tasks supported by the computer system.
- The Abstract User Interface is the canonical expression of an abstract rendering of the domain concepts and domain functions in conformance to the System Task Model. At this level of abstraction, rendering is interactor independent. It is similar in spirit to the notion of common information representation proposed in [Hüther 98].
- The Interactors Model describes the interactors available for physical rendering. Graphical user interface widgets, speech sentences are examples of interactors. More generally, an interactor belongs to a modality. As discussed in [Coutaz 95], a modality m is defined as the couple m = <r, d> where r denotes a representational system, and d the physical I/O device used to convey expressions of the representational system. Both r and d can be characterized with a rendition cost

Cr and Cd.

- The Platform Model describes the physical characteristics of the target platforms.
- The Environment Model specifies the context of use. The scope of the notion of environment is still unclear in the literature. In this discussion, the environment roughly covers objects, persons and events that are peripheral to the current task but that may have an impact on the system and/or the user's behavior. For example, a mobile phone that knows it is used in a public area will automatically switch sonic calls to vibrations.
- The Physical User Interface results from the constraints resolution and/or constraints propagation expressed in the Platform Model, the Abstract User Interface Model, the Interactors Model and the Environment Model.
- The models related to the physical world (i.e., the Platform, Interactors, and Environments models) may in turn have an impact on the System Task and the User Task models.

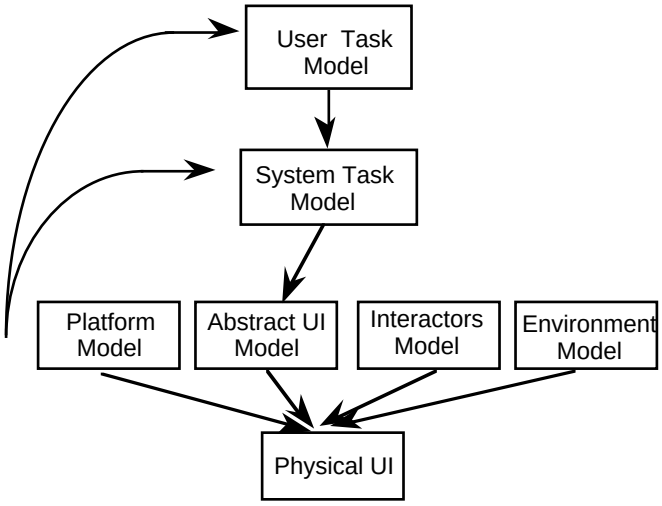

Figure 2: A framework for the development of plastic user interfaces.

The framework of Figure 2 sets the foundations for the development of plastic user interfaces. To be operational, the models it involves must be formalized. Next section provides preliminary requirements and heuristics in that direction.

4. Preliminary Requirements and Heuristics

An early analysis of an electronic agenda has led us to formulate a number of requirements and heuristics in relation to the User Task Model, the System Task Model, and the Interactors Model. The electronic agenda runs both on a workstation and the Palm Pilot.

User task Model: Domain concepts involved in a task should be ranked according to their level of importance in the task domain. For example, in the agenda application, the day of the month and the month of the year are first class objects. Conversely, the day of the week (e.g., whether the 15th of October is a Wednesday or a Friday) is not necessary to carry most frequent tasks. Current User Task and System Task models do not specify the relative importance of domain concepts. The availability of such information, however, would help the rendering process using general heuristics such as the following one: First class objects should be observable [WG2.7 96] whereas second class objects may be browsable if observability cannot be guaranteed due to the lack of physical resources. In the agenda example, a date is displayed as "Wednesday October 15, 1998" on the workstation, and as "15 Oct 98" on the Palm Pilot where screen resources are scarce.

The System Task Model: Generic articulatory tasks should be identified in a systematic way and inserted appropriately in the System Task Model. An articulatory task is domain independent. As such, it does not appear in the User Task Model and its execution does not modify the value of

domain concepts. Scrolling and navigating are typical articulatory tasks. In the context of plasticity, browsability, when used for accessing second class objects, induces articulatory tasks. For example, opening the agenda at a certain date requires selecting the month and the day in the calendar. On the PC version, the calendar is always visible whereas on the Palm Pilot, the calendar must be opened on demand. The "GoTo" command is an articulatory task introduced in the System Task Model of the Palm Pilot version to access the calendar, a browsable domain concept.

The Interactors Model: Interactors should explicitly describe the physical resources needed for their instantiation. They should also specify the abstract data types they are able to handle. The abstract data type description provides the information necessary to perform the mapping between a domain concept and a set of candidate interactors. The rendering cost drives the selection of the final interactors set for presenting domain concepts. For example, the display footprint is a way to express the rendition cost of a graphical interactor. Temporal footprint, i.e., the time to enunciate a vocal output message, expresses the rendition cost of a speech output interactor. Figure 3 shows the thumb index interactor used on the PC for setting the visualization mode of the agenda, whereas the Palm Pilot version deploys a tiny set of three icons. The visualization mode is an enumerated data type (day, month, and year) that can be mapped to a set of three icon interactors. Both sets use the same physical output device (i.e., the screen) but two different representational systems (text and graphics) whose rendition costs Cr (their display footprint) are different. In the example of the date ("October 1998" versus "Oct 98") the same representational system is used (text) using distinct pragmatic values that convey the same semantics at a lower footprint cost.

Figure 3: On the left, the thumb index used for the workstation. On the right, the icons set for the Palm Pilot.

Having presented general requirements and heuristics for the components of our framework, we now discuss the principles for producing a physical user interface from an Abstract User Interface and an Interactors model.

5. From the Abstract User Interface and Interactor Models to the Physical User Interface

The notion of Presentation Unit (PU) used in TRIDENT [Vanderdonckt 95] sets the foundations of a canonical representation for the Abstract Interface Model. A PU is a hierarchical structure that models information containers such as workspaces, as well as elementary units that correspond to domain concepts at the appropriate level of granularity. In addition to the structural static description of a PU, PU's have a behavior and are linked by navigational relationships. These relationships reify task ordering as expressed in the System Task Model.

The Physical User Interface is obtained by associating an interactor to every PU. Additionally, an interactors configuration is performed for PU's composed of sub-PU's.

For the selection of interactors, we advocate a two step process :

1. correspondence matching between PU's and interactors,
2. correspondence matching between the interactors selected in Step 1 and the physical resources provided by the system.

Correspondence matching between PU's and Interactors is based on consistency checking between the information flows that the PU's and the Interactors support respectively. An information flow is characterized with the following attributes :

- Direction of information: one way input, one way output, two ways input and output,
- Information data type: type, domain of values, cardinality. For a PU, the information data type corresponds to the domain concept supported by the PU. For an interactor, the information data type denotes the abstract domain the interactor is able to render.

For example, the information flow for a PU that corresponds to the concept of Temperature, is described as *direction of information: one way output; type: integer; domain of values: [-50, +100]; cardinality:1*. Interactors, such as the graphic thermometer, the graphic bar chart, and the graphic/ audio real number support information flows that are compatible with PU Temperature. Their information flow is defined as *direction of information: one way input; type: real; domain of values: unspecified; cardinality:1*. Interactors, such as the check box and the radio button, are not consistent with our PU. They are discarded in step1 of the selection process.

Concept	Direction	Data Type	Cardinality	Domain
Person name	Out	ASCII	1	Undefined
Workstation name	Out	ASCII	1	Undefined
Level of activity	Out	Integer	1	[0, 100], step =1
Quit command	In	ASCII	1	{Quitter}
System Purpose	Out	ASCII	1	Undefined

Table 1: Information flow of the PU's for the MMS mediaspace

Correspondence matching between the interactors selected in step 1 and the physical resources of the system, is based on constraints resolution between the rendering costs of the interactors (<Cr, Cd>) and the availability of the physical resources of the computer system. For example, the thermometer and the bar chart interactors both require a screen (Cd=Screen) whereas, for the real number, Cd=(Screen or Loudspeaker). In addition, their representational cost Cr (Cr=Screen footprint) is higher than that of the graphic real number. On a mobile phone with no screen or with a small screen, the audio/graphic real number satisfies the constraints.

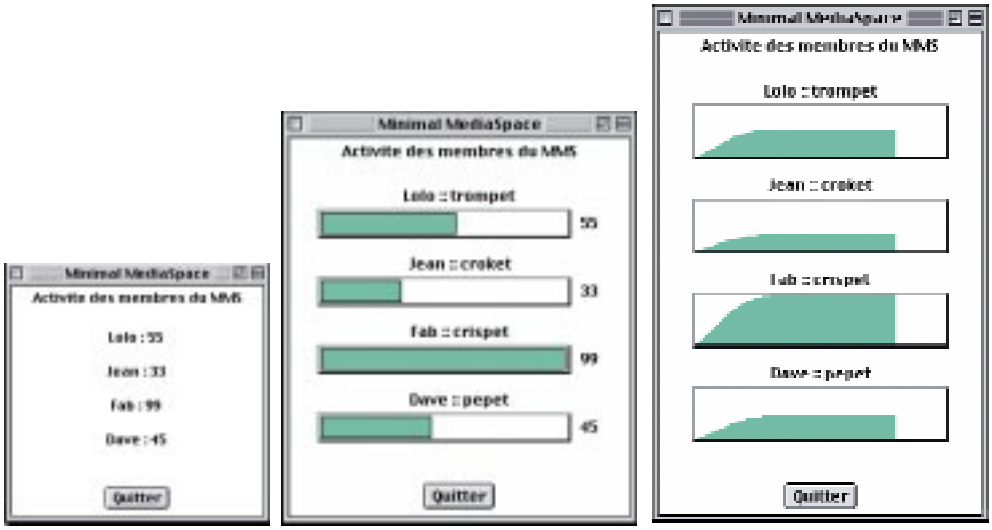

Fig 4.a Fig 4.b Fig 4.c

Figure 4: Dynamic rendering of MMS in response to screen space variations :
a) Rendering of MMS when using a small window. Only users' name and their level of activiy are observable using an integer value. Since workstation names are second class objects, they are not observable. Therefore, the System Task Model should have been modified to make them browsable.
b) Rendering of MMS using bar charts when using a medium range window size.
c) Rendering of MMS when using a window large enough to accommodate plot interactors.

We have applied this process to MMS, a simple mediaspace that conveys the level of activity of the community members of our lab. The activity level is computed from the use of the mouse and the keyboard. Image differences from video cameras could have been used as well as a source of input. In this example, a single compound PU contains three elementary PU's: the purpose of the system, a

quit command, the list of users currently connected to MMS, their activity level and the name of their workstation. All of the concepts are first class objects except the name of the workstation the user is currently connected to. Table 1 shows the definition of the elementary PU's.

The interactors that satisfy the PU's information flow are the following: the string interactor for the purpose of the system, and for the users and workstations name; the button interactor fits the quit command; the integer, the bar chart and the plot interactors match the level of activity. In our demo, interactors are selected dynamically according to the size of the window that corresponds to the compound workspace PU. The sequence in Figure 4 shows the renderings as the user enlarges/shrinks the workspace window. Figure 5 illustrates the internal representation used to compute the final rendering. Each node is characterized with a cost and the set of alternatives for rendering a PU.

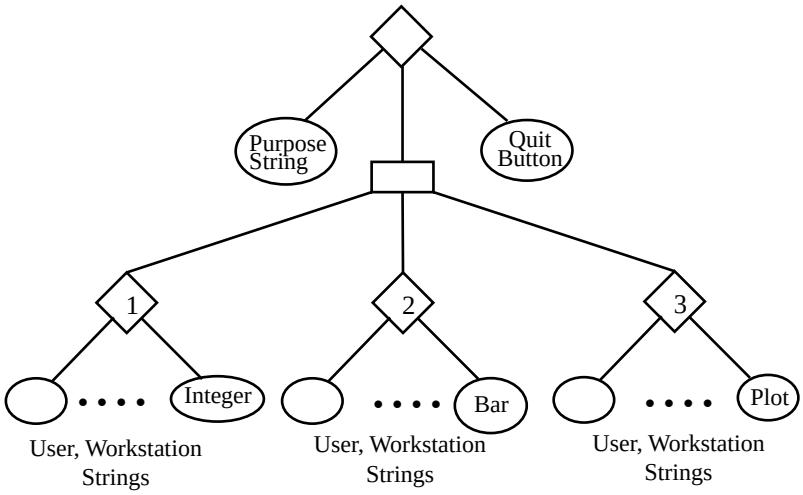

Figure 5: Internal representation for the solution space of MMS. Diamond nodes represent interactors composition with layout rules. Rectangle nodes denote rendering alternatives. Oval nodes correspond to interactors. The final representation is computed using a constraint resolution algorithm based on a cost value associated to each node and resource availability.

6. Conclusion

In this paper, we have defined the notion of plasticity for user interfaces and presented a generic framework for developing plastic user interfaces. This framework defines the boundary of the problem space. In that space, we have addressed the selection of interactors in conformance to the availability of physical resources. But we have considered one single such resources: the screen. Although MMS demonstrates the soundness of the approach, we are far from a general versatile solution.

As shown in Section 3, Task models need to be improved for addressing the problem of plasticity. At the other end of the spectrum, interactors must draw upon the research work on multimodality. The Environment and the Physical Resources models have never been addressed explicitly so far. Finally, the canonical representation that we advocate for the Abstract User Interface model may have to be specialized by domain. We hope to be able to discuss these issues at the workshop on Adaptive Design of Interactive Multimedia Presentations for Mobile Users. If adequate for the audience, we will be able to show a movie demo of MMS.

7. References

[Coutaz 95]
J. Coutaz, L. Nigay, D. Salber, A. Blandford, J. May, R. Young, Four Easy Pieces for Assessing the Usability of Multimodal Interaction: The CARE properties, Proceedings of the INTERACT'95 conference, S. A. Arnesen & D. Gilmore Eds., Chapman&Hall Publ., Lillehammer, Norway, June 1995, pp. 115-120.

[Hüther 98]
M. Hüther, T. Rist, Entering a shared Information Space through Heterogeneous Communication Devices, ECAI98 workshop on Interactive Multimedia, 1998.
http://www.dfki.uni-sb.de/imedia/workshops/ecai98/#Program

[Thevenin 98]
D. Thévenin, Interfaces Homme-Machine autoconfigurables, DEA Informatique, Université Joseph Fourier, France, 1998.

[Vanderdonckt 95]
J. Vanderdonckt, Knowledge-Based Systems for Automated User Interface Generation; The TRIDENT Experience. Rapport technique RP-95-010, Facultés Universitaires de Notre-Dame de la Paix, Institut d'Informatique, Namur, Belgique, mars 1995.
http://www.info.fundc.ac.be/cgi-bin/pub-spec-paper?RP-95-010.

[WG2.7 96]
Design Principles for Interactive Software, C. Gram & G. Cockton Eds. , Chapman&Hall Publ., 1996.