REQUIREMENTS FOR AN ABSTRACT INTERACTION MACHINE

Christophe Lachenal, Joëlle Coutaz

CLIPS-IMAG
385 rue de la Bibliothèque
F-38041 Grenoble cedex 9, France
{Christophe.Lachenal, Joelle.Coutaz}@imag.fr

ABSTRACT

In this position paper, we argue that current windowing systems and toolkits, which set the foundations for the development of interactive systems, do not satisfy the software requirements for implementing state of the art user interfaces. New styles of interaction require the dynamic configuration of multiple processors and interaction resources into a uniform computational and interactional space. We propose the notion of Abstract Interaction Machine (AIM) as an approach to address this problem. An AIM is a basic software infrastructure that hides the complexity of the underlying physical infrastructure on top of which distributed UI's can be easily developed.

KEYWORDS: Abstract machine, Human Computer Interaction, ubiquitous computing.

1. INTRODUCTION

With the advent of digital cell phones, personal organizers, and wall-size augmented surfaces, users expect to interact with the same interactive system both *in the small* and *in the large* using multiple modalities. In addition, wireless connectivity offers new opportunities for using interactive systems in different environments (e.g., at home, in the street, at work) putting new demands on software functionalities such as context-awareness and dynamic discovery of interaction resources.

Meanwhile, design principles, methods and software development tools have been devised to increase systems usability while reducing development costs. But today's tools have just caught up with the static "gray box" connected to the Ethernet. Although they have achieved a high level of sophistication, current methods and tools make a number of implicit assumptions. In general, implicit assumptions simplify implementation issues and favors performance. On the other hand, they are limiting factors for innovation.

Typical examples of limitations in foundational tools such as windowing systems are discussed next. Based on the analysis of state of the art User Interfaces, we observe that the notion of platform as exemplified by the usual workstation, must be revised and extended. We then propose a formal definition for the notion of platform that fits the spirit of ubiquitous computing. From there, we suggest a number of requirements for the low-level software that would replace current windowing systems and toolkits: an abstract interaction machine that would support the dynamic reconfiguration of underlying physical infrastructure, as well as the dynamic distribution of user interfaces across multiple interaction resources.

2. LIMITING FACTORS

The development of new types of user interfaces is limited by the simplicity of the models that current windowing systems maintain for both output and input devices. For output, windows models are screen centric and screens topology is simplistic. As for input, windowing systems handle a single instance for each class of input devices. These issues are discussed next in more detail.

2.1. Screen Centricity of Windows Models

Windowing systems model windows as rectangular drawables whose borders are constrained to be parallel to that of the display. This model is based on the assumption that users keep facing the screen. With the proliferation of video beamers, it is popular to project window contents on large surfaces such as tables. In this situation, users should be able to rotate windows so that everyone can share window contents without twisting their neck. Today, rotating windows must be implemented from scratch from low-level graphics primitives as in Jazz/PAD++ [Bederson 00].

2.1. Limited topology for supporting multiple display surfaces

Windowing systems are able to support multiple screens provided that the display surfaces are kept vertical facing the user. In addition, the relative location of the display screens must be set up by the user.

Again, because they are screen-centric, windowing systems do not support topologies that include the surrounding environment (e.g., walls, tables, users' location with respect to interactive surfaces, etc.). However, for many interactive systems, the topology of the rendering surfaces matters. For example, in Built-IT [Rauterberg 98] and Rekimoto's augmented surfaces [Rekimoto 99], objects are represented as 3D graphics interactors on laptops, whereas 2D rendering is used for objects placed on an horizontal surface.

2.3. A single instance for each class of input devices

In current windowing systems, the reference workstation is supposed to have a single mouse and keyboard. As a result, multi-user applications such as MMM [Bier 92] and Kidpad

[Benford 00] whose users share the same screen with multiple mice, require the underlying toolkit and event manager to be revisited as in MID [Hourcade 99]. In Pebbles, multiple PDA's can be used as input devices to control information displayed on a wall-mounted electronic board [Myers 98].

2.4. Synthesis

Our brief analysis shows that windowing systems, which set the foundations for the UIMS technology, primarily address the development of user interfaces confined to a single PC, with limited models for display surfaces and input devices. At the opposite, the state of the art shows that some user interfaces are aimed at simple platforms such as laptops and PDA's, while others, such as I-land [Streitz 99] and Rekimoto's work, require the aggregation of multiple computers. In addition, input and output devices can be connected to, or disconnected from, a computer at any time. We synthesize the variety of situations with the following notion of platform.

2. THE PLATFORM REVISITED

We make a distinction between elementary platforms, which are built from core resources and extension resources, and clusters, which are built from elementary platforms.

3.1. Definitions

An *elementary platform* is a set of physical and software resources that function together to form a working computational unit whose state can be observed and/or modified by a human user. None of these resources per se is able to provide the user with observable and/or modifiable computational function. A personal computer, a PDA, or a mobile phone, are elementary platforms. On the other hand, resources such as processors, central and secondary memories, input and output interaction devices, sensors, and software drivers, are unable, individually, to provide the user with observable and/or modifiable computational function.

Some resources are packaged together as an immutable configuration called a core configuration. For example, a laptop, which is composed of a fixed configuration of resources, is a core configuration. The resources that form a core configuration are *core resources*. Other resources such as external displays, sensors, keyboards and mice, can be bound to (and unbound from) a core configuration at will. They are *extension resources*.

A *cluster* is a composition of elementary platforms. The cluster is homogeneous when it is composed of elementary platforms of the same class. For example, DynaWall is an homogenous cluster composed of three electronic white boards [Streitz 99]. The cluster is heterogeneous when different types of platforms are combined together as in Rekimoto's augmented surfaces.

3.2. Formal Definitions

More formally, let

- C be the set of core configurations
- E be the set of extension resources
- $C', C'' \neq \{\}: C' \subset C$ and $C'' = C - C'$
- $E', E'': E' \subset E$ and $E'' = E - E'$
- $c_1, \ldots, c_n \in C$, $e_1, \ldots, e_m \in E$ for $n \in N^*, m \in N$
- Operational be a predicate over a set of resources that returns true when this set forms a working computational artefact whose state can be observed and/or modified by a human user.

A platform is composed of a set of core and extension resources which, connected together, form a working computational artefact whose state can be observed and/or modified by a human user:

$P = \{ c_1, \ldots, c_n \} \cup \{ e_1, \ldots, e_m \}$ and Operational (P)

P is an elementary platform if and only if:

$\neg\exists$ C', E', C'', E'': Operational $(C' \cup E')$ and Operational $(C'' \cup E'')$.

In other words, P is an elementary platform if it not possible to compose two platforms from the set of resources that constitute P.

P is a cluster if it is possible to compose two platforms from the set of resources that constitute P:

$\exists$ C', E', C'', E'': Operational $(C' \cup E')$ and Operational $(C'' \cup E'')$.

Note that:

- A core configuration is not necessarily Operational. For example, the Intel Personal Server, a bluetooth-enabled micro-drive with no interaction device, is a core configuration but not an elementary platform: its state cannot be observed nor modified until it wirelessly connects to extension resources such as a display and/or a keyboard [Want 01].
- A laptop is an elementary platform. When augmented with extension resources such as a second mouse and sensors, it is still an elementary platform. Similarly, Rekimoto's data tiles form an elementary platform that can be dynamically extended by placing physical transparent tiles on a tray composed of an LCD flat screen display [Rekimoto 01].

To our knowledge, design tools for user interfaces address elementary platforms whose configuration is known at the design stage. Clusters are not addressed. On the other hand, at the implementation level, software infrastructures such as BEACH, have been developed to support clusters built from an homogeneous set of core resources (i.e., PC's) connected to a varying number of screens [Tandler 01]. Whereas BEACH provides the programmer with a single logical output display mapped onto multiple physical displays, MID addresses the dynamic connection of multiple input devices to a single core configuration. Similarly, Pebbles allows the dynamic connection of multiple PDA's to a single core configuration.

None of the current infrastructures addresses the dynamic configuration of clusters, including the discovery of both input and output interaction resources. The concept of Abstract Interaction Machine (AIM), as well as the architecture developed for iRoom [Winograd 01] and EasyLiving [Brumitt 01] are attempts to address these issues.

4. SERVICES PROVIDED BY AN AIM

Just like windowing systems and toolkits, AIM covers multiple levels of abstraction.

4.1. The "Windowing Level" of AIM

At the lowest level of abstraction, AIM, just like windowing systems, hides the functioning of the physical platform whether it be elementary or a cluster. In other words, it provides the developer (and/or the higher functional layer) with a logical space of any number of input and output interaction devices whose identity and topology are made accessible on request. Because the number and nature of resources may vary, AIM includes mechanisms for resource discovery.

Interaction resources go wireless and can easily be moved around. Therefore, proximity detection and orientation are two basic spatial services that AIM should include. These services are required for AIM to maintain an operational topology.

The topology of interaction resources includes the spatial relationships between the output devices of the platform, between the input devices, as well as between the input and output devices. Moreover, the topology includes the spatial relationships between the interaction resources and the users currently using (or near by) the system. The knowledge of users' location allows higher software levels to render information in the appropriate form and location. For example, information projected on a round table could automatically be oriented towards the user.

For output, the reference coordinates can be dynamically modified. It can't anymore be the static top left corner of a graphics display!

Any change in the resources (and users) topology is reflected to the upper layer, such as the toolkit level.

4.2 The "Toolkit Level" of AIM

At the toolkit level, the interactors that compose the user interface of a particular application (whatever an application is within the ubicomp world) can be distributed across the resources of the platform. In addition, they may migrate between the interaction resources at run time. The granularity for distribution and migration may vary from application level to pixel level:

- At the application level, the user interface is fully replicated on the platforms of the cluster. If the cluster is heterogeneous, then each platform runs a specific targeted user interface initialized, for example, from a pre-computed user interface. All of these user interfaces, however, simultaneously cover the same functional core.

- At the workspace level, the user interface components that can migrate between platforms are workspaces. A workspace groups together a collection of interactors that support the execution of a set of logically connected tasks. In graphical user interfaces, a workspace is mapped onto the notion of window. The painter metaphor presented in Rekimoto's pick and drop [Rekimoto 97], is an example of distribution at the workspace level: tools are presented on a PDA whereas the drawing area is mapped onto a white board. Going one-step further with AIM, the tools palette (possibly the drawing area) can migrate at run time between the PDA and the electronic board.

- At the domain concept level, the user interface components that can be distributed between platforms are interactors. Here, interactors render domain concepts. In Rekimoto's augmented surfaces, domain concepts can be distributed between laptops and horizontal and vertical surfaces.

- At the pixel level, any user interface component can be partitioned across multiple platforms. For example, in I-land, a window may lie over two contiguous white boards simultaneously. When the cluster is heterogeneous, designers need to consider multiple sources of disruption. For example, how to represent a window whose content lies across a white board and a PDA? From a user's perspective, is this desirable?

6. CONCLUSION

We are currently implementing an instance of AIM for an augmented room that includes multiple interaction surfaces of varying size: an augmented table where the mouse is replaced by physical tokens tracked by a computer vision system, an augmented wall whose pointing devices are laser beams, and PDA's. We are aware that our concepts are still preliminary, but could serve as input to a discussion in a workshop.

7. REFERENCES

[Bederson 00] Bederson, B., Meyer, J. Good, L. Jazz: An Extensible Zoomable User Interface Graphics Toolkit in Java. In Proceedings of UIST 2000. May 2000. p171-180.

[Benford 00] Benford, S., Bederson, B., Akesson, K., Bayon, V., Druin, A., Hansson, P., Hourcade, J., Ingram, R., Neale, H., O'Malley, C., Simsarian, K., Stanton, D., Sundblad, Y., Taxen, G. Designing Storytelling Technologies to Encourage Collaboration Between Young Children. In Proceedings of CHI'2000, The Hague, Netherlands, April 1-6, ACM, New York, p556-563.

[Bier 92] Bier E., Freeman, S., Pier, K., The Multi-Device Multi-User Multi-Editor. In Proc. of the ACM conf. On Human Factors in Computer Human Interaction (CHI92), (1992), pp. 645-646.

[Brumitt 01] Brumitt, B., Shafer, S. Better Living Through Geometry. Personal and Ubiquitous Computing 2001. Vol 5.1 Springer.

[Hourcade 99] Hourcade, J., Bederson, B. Architecture and Implementation of a Java Package for Multiple Input Devices (MID). 1999. Disponible à l'adresse : http://www.cs.umd.edu/hcil/mid/

[Myers 98] Myers, B., Stiel, H., Gargiulo, R. Collaboration Using Multiple PDAs Connected to a PC. In Proceedings CSCW'98: ACM Conference on Computer-Supported Cooperative Work, 1998, Seattle, WA. pp. 285-294.

[Rauterberg 98] Rauterberg, M. et al. BUILT-IT: A Planning Tool for Consruction and Design. In Proc. Of the ACM Conf. In Human Factors in Computing Systems (CHI98) Conference Companion, (1998), pp. 177-178

[Rekimoto 99] Rekimoto, J., Masanori, S. Augmented Surfaces : A Spatially Continous Workspace for Hybrid Computing Environments. Proceedings of CHI'99, 1999.

[Rekimoto 01] Rekimoto, Yun. Ullmer, Brygg. Oba, Haro. DataTiles: A Modular Platform for Mixed Physical and Graphical Interactions. In Proceedings of CHI2001, Seattle 2001.

[Rekimoto 97] Rekimoto, J. Pick-and-Drop: A Direct Manipulation Technique for Multiple Computer Environments. In Proceedings of UIST'97, pp. 31-39, 1997.

[Streitz 99] Streitz, et al. I-LAND: An interactive Landscape for Creativity and Innovation. In Proceedings of CHI'99.

[Tandler 01] Tandler, P. Software Infrastructure for Ubiquitous computing Environments : Supporting synchronous Collaboration with Heterogenous devices. In Proceedings of UbiComp 2001.

[Want 01] Want, et al. The Personnal Server : The Center of Your Ubiquitous World. Intel Research White Paper May 2001.

[Winograd 01] Winograd, T. Architecture for Context, Human Computer Interaction, Lawrence Erlbaum Publ., 16(2-4), (2001), pp.401-419.