Coupling Interaction Resources and Technical Support

Nicolas Barralon, Joëlle Coutaz, and Christophe Lachenal

Université Joseph Fourier, Grenoble
{Nicolas.Barralon, Joelle.Coutaz}@imag.fr

Abstract. Coupling is the action of binding two entities so that they can operate together to provide new functions. In this article, we propose a formal definition for coupling and present a graph theoretic notation so that the side-effects of the creation of a coupling can be analyzed in a formal and systematic way. We then describe I-AM (Interaction Abstract Machine), a middleware that supports the dynamic coupling of interaction resources such as screens, keyboards and mice, to form a unified interactive space. Using our notation, we illustrate how couplings are supported in I-AM.

Keywords: Multi-surface interaction, multi-instrument interaction, devices assembly, ambient intelligence, ubiquitous computing, UI development tool.

1 Introduction

Technological advances in computing, sensing, and networking, are rapidly leading to the capacity for individuals to create and mould their own interactive spaces. As envisioned by Weiser [18], interactive spaces will be assembled opportunistically by coupling private devices with interaction resources from public hot spots to access services within the global computing fabric. Interactive spaces will also take the form of autonomous computing islands whose horizon will evolve, split and merge under users' control.

From the software perspective, this ultra-flexibility calls for multi-scale infrastructures that act as a bridge between heterogeneous and dynamic sets of hardware, operating systems, and interaction resources. Projects such as Aura [15], Easyliving [3], Dynamo [10], Augmented Surfaces [11], and i-LAND [16] are early attempts in this direction. The same holds for prototypes based on proximal interaction and synchronous gestures [8, 9,12], or on physical connection [7, 13, 17]. These projects address a particular aspect of the vision where distributed and migratory user interfaces (UI) replace the conventional centralized paradigm. However, all of these systems are based on software concepts that were proposed twenty years ago when user interfaces were confined to a single computer equipped with a single screen, a single keyboard and a single pointing device.

With the exception of e-Gadget [5] and Smoblet [14], no one seems to have called into question whether such software concepts and tools are appropriate for the newly emerging forms of interaction based on coupling interaction resources. For example,

C. Stephanidis (Ed.): Universal Access in HCI, Part II, HCII 2007, LNCS 4555, pp. 13–22, 2007.

one can couple two entities, such as a wallet and home keys, by shaking them together [9]. As a result, an alarm can signal when one is separated from the other. But, what should happen when the keys are coupled with a pair of shoes? Are then the shoes coupled with the wallet?

In this article, we present a theoretic framework for reasoning about coupling [1] followed by the description of I-AM (Interaction Abstract Machine), a middleware that supports the dynamic coupling of interaction resources such as display screens and keyboards to form a unified interactive space.

2 Coupling Entities: Definition

The word "coupling" may be used to denote an act, or the result of this act.

- As an act, *coupling* is the action of binding two entities so that they operate conjointly to provide a set of functions that these entities cannot provide individually.
- As the result of an act, *a coupling* is an assembly of the source entities, that is, a new compound entity that provides a new set of functions that the source entities, alone, cannot provide.

In both cases, given two entities, the nature of the act determines the resulting set of functions. For example, in Hinckley's dynamic display tiling [8], users obtain different functions depending on the way the tablets are bumped together: if one tablet rests flat on a desk surface, and a second tablet is bumped into the base tablet, then the resulting function is the creation of a larger display. Alternatively, if the two tablets are bumped symmetrically, the same content is replicated on both displays to support face-to-face collaboration.

We express the twofold acceptation of coupling (either as an act, or as an entity) in the following formal way. Let:

- **E** be a non-empty finite set of entities and **F**, the set of functions that these entities provide individually,
- *func*, the function that returns the set of functions f ($f \subset \mathbf{F}$) that an entity $e \in \mathbf{E}$ provides: $f = func(e)$,
- **A,** a non-empty set of sequences of actions a,
- **C**, the set of couplings between entities belonging to **E**, using sequences of actions $a \in \mathbf{A}$,
- $e \in \mathbf{E}$, the compound entity that results from binding e_1 and e_2 by the way of the sequence of actions $a \in \mathbf{A}$,

then, the coupling c ($c \in \mathbf{C}$) is defined as the Cartesian product **E** x **E** x **A** in **E**:

$$c: \mathbf{E} \times \mathbf{E} \times \mathbf{A} \rightarrow \mathbf{E}$$

and is denoted as:

$$c = (e_1, e_2, e): \forall f_i \neq f_1 \wedge f_i \neq f_2 : \ (f_1 \cap f_i = \varnothing) \wedge (f_2 \cap f_i = \varnothing) \qquad (1)$$

where $e_1, e_2 \in \mathbf{E}$, $f_1 = func(e_1), f_2 = func(e_2), f = func(e)$

or as:

$$c = (e_1, e_2, f) \qquad e_1 \overset{c}{\text{———}} e_2 \qquad (2)$$

or as: (e_1, c, e_2) or (3)

In notation (1), the focus of attention is the new compound entity obtained by the way of coupling. Notation (2) stresses the importance of the resulting set of functions while maintaining an explicit reference to the source entities. Notations (3) make the bond between the source entities explicit.

As suggested by the wallet, keys and shoes example, coupling two interaction resources may generate new couplings. In the following section, we present a theoretic notation to reason about causal and consequent couplings.

3 Causal Couplings and Their Consequents: A Formal Analysis

As in chemistry, couplings may have *causal relationships*: coupling an entity with a compound entity may entail a chain of reactions. Some bonds may be destroyed, possibly giving rise to multiple entities. Alternatively, additional couplings may be created. A coupling is *causal* when its creation implies, as a side effect, the creation of additional couplings. These additional couplings are called consequent couplings or simply, *consequents*.

3.1 Formal Analysis with a Graph Theoretic Notation

We represent couplings using a graph notation where nodes denote entities, and where edges express the existence of couplings. Symbols "*" and "=" denote causal and consequent couplings respectively. The "?" symbol denotes the couplings that are under evaluation (i.e. keeping them as consequents or rejecting them has not been decided yet). To express their transitory state, causal couplings, as well as consequents and undecided couplings are represented as dotted edges. Let:

- *EDGE* be the set of edges of the graph under consideration.
- $r_1(c)$ (resp. $r_2(c)$) be the first (resp. the second) interaction resource involved in the coupling $(r1, c, r2)$.
- $F(r_1, r_2)$ be the function that results from (r1, c, r2).
- $Compatible(f_1, f_2, f_3)$ returns TRUE if the functions f_1 and f_2 allow the existence of the function f_3. To be TRUE, $Compatible(f_1, f_2, f_3)$ may require the suppression of existing couplings. Although important (and challenging), this possibility is not addressed in this article.

The principle of our algorithm is the following: consider every new edge that results from the transitive closure with paths of length 2 that contain both $r_1(c)$ and $r_2(c)$. If this new edge corresponds to the creation of a coupling whose function is compatible with the functions provided by its neighboring edges, then it is created. In turn, the coupling that this edge denotes becomes a causal coupling and the algorithm is applied again.

More formally:

```
For every causal coupling c
   Build the set of nodes Nc such that :
      n∈Nc ⇔    n∈path ∧ length(path)= 2
                ∧ r1(c)∈path ∧ r2(c) ∈path

   For all n ∈ Nc and n≠r1(c) and n≠r2(c)
      if edge(n,r1(c)) ∈ EDGE
         if compatible(F(c, F(n,r1(c), F(n,r2(c))) then
                EDGE = EDGE ∪ new edge(r2(c), n)
         else
            if compatible(F(c), F(n, r2(c)), F(n, r1(c))) then
                EDGE = EDGE ∪ new edge(r1(c), s)
```

3.2 Illustration

To illustrate the algorithm, let's consider the initial configuration of couplings represented in Fig. 1. On the left, *Screen1* is coupled with *Mouse1* and *Keyboard1*, and *Mouse1* is coupled with *Keyboard1* to provide *Keyboard1* with the input focus function. This configuration corresponds to a private workstation. On the right, a public *Screen2* is coupled with a public pointing device *Mouse2*. Because *Screen1* and *Screen2* are compatible by design (resulting in the enlarge display function), *c5* is performed (for example, by a proximity detection service).

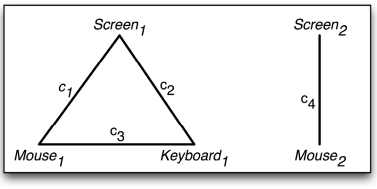

Fig. 1. Initial configuration. On the left, a private workstation; on the right, a public configuration

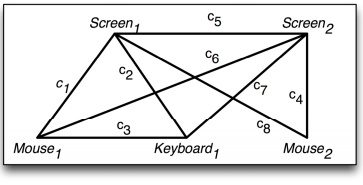

Fig. 2. Final configuration where the couplings *c6*, *c7* and *c8* are the consequents of the causal coupling *c5*

The final configuration that results from the causal coupling *c5* is shown in Fig. 2: The owner of the private workstation can manipulate digital information displayed on *Screen2* and *Screen1* using the private interaction resources *Mouse1* and *Keyboard1*. In addition, information can be designated on both screens with *Mouse2*, but for privacy reason, *Mouse2* cannot be coupled to *Keyboard1*. In a different situation

where the workstations were owned by two distinct users who wanted to collaborate via a unified space, the compatibility functions would be different resulting in a distinct final configuration.

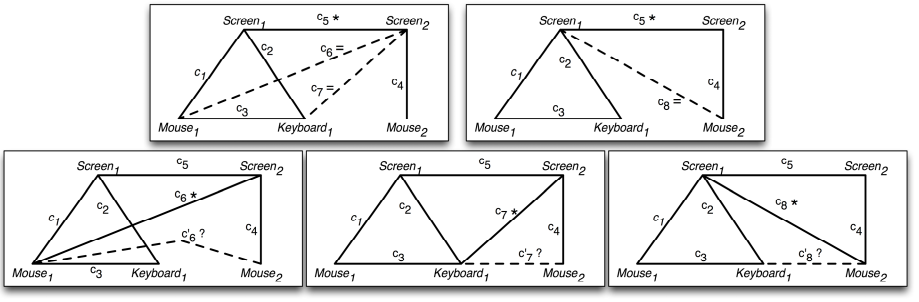

Fig. 3. Evaluation steps resulting from the causal coupling c5

Fig. 3 shows the successive steps that lead to the final configuration of Fig. 2. Fig. 3 (top left) corresponds to the generation of c_6 and c_7 that result from the transitive closure with paths c_5–c_1 and c_5–c_2 respectively. Fig. 3 (top right) shows the generation of c_8 that results from the transitive closure with path c_5–c_4. Because the function that results from c_6 is compatible with that of c_1 and c_5, c_6 is created. The same holds for c_7 and c_8 whose resulting functions are compatible with that of c_5 and c_2, and c_5 and c_4 respectively. c_6, c_7 and c_8 are now causal couplings. Fig. 3 (bottom left) corresponds to the application of the algorithm to c_6 with the evaluation of c'_6 that results from the transitive closure with paths c_6–c_4. The function that results from c'_6 is not compatible with that of c_6 and c_4 (coupling a private mouse with a public mouse to access any display area is considered as inappropriate for this particular situation). The same holds for c'_7 and c'_8 that result from the transitive closure with paths c_4–c_7 and c_8–c_2 respectively. For this particular situation, *Mouse2* cannot serve as input focus for *Keyboard1* (Fig. 3, bottom center and bottom right). In short, the causal coupling c_5 has three consequents: c_6, c_7, and c_8.

I-AM, presented next, offers a technical support for dynamically coupling interaction resources including causal and consequent couplings.

4 I-AM: Overview

I-AM is a technical instantiation of the ontology described in [4] where a computer platform is modeled as an actor characterized by a dynamic set of interaction resources. An interaction resource is a real-world physical object (with its own attributes such as shape, size and location in space) that plays a role such as that of a surface or of an instrument. Typically, a display screen is an interaction resource that plays the role of a surface whereas the keyboard plays the role of an instrument.

In I-AM, the interaction resources may be distributed across multiple machines running distinct operating systems. In this space, users can distribute and migrate user interfaces as if they were handled by a single computer. This illusion of a unified

space is provided at no extra cost for the developer who can re-use the conventional GUI programming paradigm.

4.1 Technical Principles of I-AM

Fig. 4 illustrates the technical principles of I-AM where the interactive space is composed of three machines that run different operating systems. Each machine handles a screen and possibly a mouse and a keyboard. Each screen is assigned the role of a surface and each mouse and keyboard has the role of pointing and input text instruments, respectively. Screens are equipped with link points.

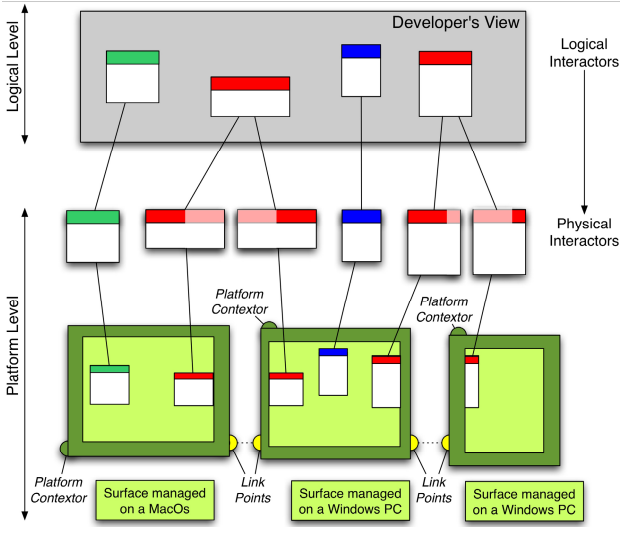

Fig. 4. Technical principles of I-AM

A link point is a reference point located at the edge of a screen. It can take the form of a physical sensor (e.g., infrared sensors, accelerometers as in Hinckley's example of synchronous gestures [8]). It can be a painted dot tracked by a computer vision system, or any conventional identifiable spot on the edge of a screen, or IR sensors or software interaction techniques [2]. Bringing two link points together, from two different screens, results in coupling the two screens. Because link points are geometric points, the screens are automatically related by geometric relations.

The bottom of the figure shows the distribution of the user interface across three screens. Some interactors such as the top left window of the developer's view, are fully rendered within a single screen whereas other interactors, such as the right most window of the developer's view, are split across two screens. In the latter case, the logical interactor of the developer's view is mapped into two effective interactors whose rendering is tightly coupled to entertain the illusion of a single surface: as the user moves one of the effective interactors using any pointing instrument of the interactive space, the other "twin" effective interactor is moved and transformed as if the twins were one single piece.

4.2 The Programmer's View

I-AM preserves the conventional GUI programming paradigm. As shown below, the programmer creates a window interactor, specifies its size and location, fills it with a picture, and asks I-AM to manage the window. The window is then mapped by I-AM on the physical configuration of the screens. With I-AM, the programmer gets UI distribution and migration between multiple surfaces for free, as well as the management of multiple instruments and cursors.

```
// My program is an IAMApp
IAMApp myiamapp = new IAMApp ();

// Create mywindow, and assign
IAMWindow mywindow = new IAMWindow();
mywindow.setCenterLocation (300, 300);
mywindow.setSize (300, 200);
…
// Ask I-AM to manage mywindow
myiamapp.addInteractor (mywindow);
```

4.3 The End-User's View

Fig. 5 and Fig. 6 show early examples of interaction techniques that allow users to mould their interactive space. Fig. 5-a corresponds to the situation where two applications are running on two independent workstations. A closed blue border outlines the screens to denote the absence of coupling. In Fig. 5-b, the screens are now coupled to provide the "single display area" function. An halo outlines the display area and a gateway shows where interactors can transit between the coupled screens.

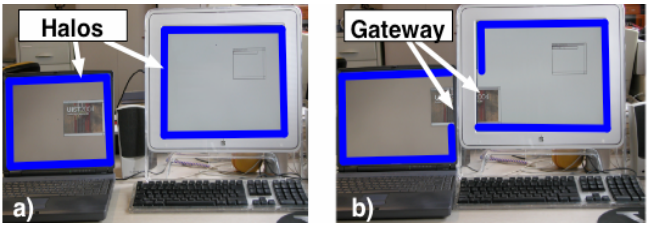

Fig. 5. (a) The PC and the Macintosh are decoupled and run two applications. (b) The two screens are coupled to form a single information space. Halos have been artificially enhanced on the pictures to increase readability.

Within an interactive space, any instrument can be used to modify any interactor. For example, in the configuration of Fig. 5-b, a PC mouse can be used to move a window created on the Macintosh and migrate it to the PC. In addition, the two mice can be used simultaneously. In Fig. 6, the user has selected the text field interactor displayed on the Macintosh screen with the PC mouse. Text can now be entered with the PC keyboard. If the text field is also selected with the Macintosh mouse, text can be entered with the Macintosh keyboard as well.

We have designed (but not evaluated) several interaction techniques for coupling screens. If the screens are equipped with infrared or if they are tracked by a computer vision system, coupling can be performed by bringing the screens in close contact. This sequence of actions *a*, is similar in spirit to Hinckley's synchronous gestures where devices are bumped against each other [8]. An alternative sequence of actions, inspired from SyncTap [12] called "Click'n Couple" [1], consists in bringing the cursors of the mice face to face, and then click the mouse buttons simultaneously (i.e. within the same temporal window).

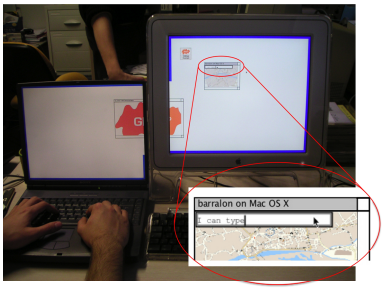

Fig. 6. Entering characters in a text field located on a Macintosh screen using a PC keyboard. Meanwhile a window is transiting through the gateway.

Having presented the technical principles of I-AM as well as the views it offers to the programmer and to the end-user, we now analyze I-AM more formally using our theoretic framework.

4.4 I-AM and the Theoretic Framework

We will use Figures 5, 6 and 7 to illustrate how couplings are handled in I-AM. In Fig. 5-a, two computers, a PC and a Macintosh, run independently. As for conventional window managers, the mouse and the keyboard of a computer are automatically coupled by I-AM to provide the "text input focus" function. The situation of Fig. 5-a is represented as two independent graphs: c1-c2-c3 and c4-c5-c6 (see Fig. 7).

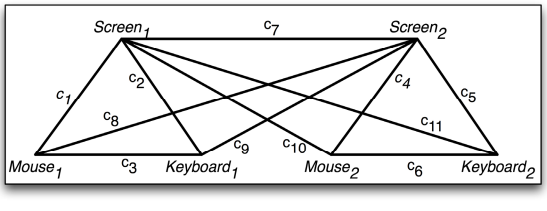

Fig. 7. The configuration that results from the causal coupling c_7 between the two screens

When the user performs any of the appropriate sequence of actions mentioned previously, *c7* is created between *Screen1* and *Screen2* (cf. Fig. 7), and a new

function *f* is now available: *f* is an affine transform that supports different screen resolution and orientation, as well as bezels thickness so that windows and figures can overlap multiple screens without any deformation. Has *c7* consequent couplings?

By applying the algorithm presented in 3.1, the consequents *c8*, *c9*, *c10*, and *c11* are created because coupling *Mouse1* with *Screen2* produces a pointing function that is compatible with *f*. Similarly, coupling *Keyboard1* with *Screen2* produces a text entry function that is compatible with *f*. The same holds for *Mouse2* and *Keyboard2* with regard to *Screen1*. The configuration shown in Fig. 7 expresses the capacity for any interactor displayed on the unified surface to be coupled to the mouse-keyboard of the Macintosh as well as to the mouse-keyboard of the PC. Then, as shown in Fig. 6, characters can be entered simultaneously from any keyboard.

5 Conclusion

In summary, we have developed I-AM, a middleware that supports the dynamic coupling of interaction resources, possibly distributed across an heterogeneous set of platforms. I-AM advances the state of the art by addressing the following problems: 1) Platforms heterogeneity (running a mix of MacOs X, Windows NT and Windows XP); 2) Interaction resources heterogeneity (e.g., screens with different sizes, resolutions); 3) Platforms and interaction resources discovery; 4) Multi-surface interaction grounded on the dynamic coupling of hinged display surfaces whose spatial relations are automatically modeled and maintained; 5) Multi-keyboard, multi-pointer capabilities (so that a user can use the mouse of a PC to manipulate a window displayed on a MacOS screen and drag the window across screens boundaries as if there were a single screen).

Coupling interaction resources is not a new phenomenon. But in the GUI computer world, most couplings are pre-packaged and immutable. As we move to ubiquitous computing and ambient interactive spaces, the story is not as simple. In particular, couplings may engender new couplings. We propose a formalism using a graph theoretic notation to reason about the consequents of causal couplings in a systematic way. We use the notion of compatibility between the function returned by a consequent and the functions provided by its two neighbors. Other criteria could be used as well such as that of observability, predictability, traceability, and controllability. These issues are discussed in detail in [6] with the concept of meta-UI. Meta-UI's will allow users to build and control their own interactive spaces by coupling entities of many forms.

Acknowledgments. This work has been partly supported by Project EMODE (ITEA-if4046) and the NoE SIMILAR- FP6-507609.

References

1. Barralon, N.: Couplage de resources d'interaction en informatique ambiante. Thèse de l'Université Joseph Fourier (2006)
2. Barralon, N., Nguyen, V., Rey, G.: Techniques de couplage de bureaux: Ambient-Desktop comme illustration. In: Proceedings UbiMob'05, Grenoble, 31 mai-3 juin, pp. 193–200 (2005)

3. Brumitt, B., Shafer, S.: Better Living Through Geometry. In: Personal and Ubiquitous Computing, vol. 5(1), Springer, Heidelberg (2001)
4. Coutaz, J., Lachenal, C., Dupuy-Chessa, S.: Ontology for Multi-surface Interaction. In: Rauterberg, M., et al. (ed.) Proc. Interact 2003. IFIP, pp. 447–454. IOS Press, Amsterdam (2003)
5. Christopoulou, E., Kameas, A.: Using Ontologies to Address Key Issues in Ubiquitous Computing Systems. In: Markopoulos, P., Eggen, B., Aarts, E., Crowley, J.L. (eds) EUSAI 2004. LNCS, vol. 3295, pp. 13–24. Springer, Heidelberg (2004)
6. Coutaz, J.: Meta-User Interface for Ambient Spaces, In TAMODIA'06. In: Coninx, K., Luyten, K., Schneider, K.A. (eds.) TAMODIA 2006. LNCS, vol. 4385, pp. 1–15. Springer, Heidelberg (2007)
7. Gorbet, G.M, Orth, M., Ishii, H.: Triangles: Tangible Interface for Manipulation and Exploration of Digital Information Topography. In: Proc. of CHI'98, Los Angeles, pp. 49–56 (1998)
8. Hinckley, K.: Synchronous gestures for multiple persons and computers. In: Proc. UIST03, pp. 149–158. ACM, New York (2003)
9. Holmquist, L.E., Mattern, F., Schiele, B., Alahuhta, P., Beigl, M., Gellersen, H.W., Smart-Its Friends, A.: Technique for Users to Easily Establish Connections between Smart Artefacts. In: Proc. of Ubicomp, Atlanta, Georgia, 2001, pp. 116–221 (2001)
10. Izadi, S., Brignull, H., Rodden, T., Rogers, Y., Underwood, M., Dynamo, A.: A public interactive surface supporting the cooperative sharing and exchange of media. In: Proc. UIST 2003, pp. 159–168. ACM, New York (2003)
11. Rekimoto, J., Masanori, S., Augmented Surfaces, A.: Augmented Surfaces: A Spatially Continuous Workspace for Hybrid Computing Environments. In: Proc. CHI'99, pp. 378–385. ACM, New York (1999)
12. Rekimoto, J., Ayatsuka, Y., Kohno, M.: SyncTap: an Interaction Technique for Mobile Networking. In: Chittaro, L. (ed.) Mobile HCI 2003. LNCS, vol. 2795, pp. 104–115. Springer, Heidelberg (2003)
13. Resnick, M., Martin, F., Berg, R., Borovoy, R., Colella, V., Kramer, K., Silverman, B.: Digital Manipulatives: New Toys to Think With. In: Proc. of CHI'98, Los Angeles, pp. 281–287 (1998)
14. Siegemund, F., Krauer, T.: Integrating Handhelds into Environments of Cooperating Smart Everyday Objects. In: Markopoulos, P., Eggen, B., Aarts, E., Crowley, J.L. (eds.) EUSAI 2004. LNCS, vol. 3295, pp. 160–171. Springer, Heidelberg (2004)
15. Sousa, J., Garlan, D.: Aura: an Architectural Framework for User Mobility in Ubiquitous Computing Environments. In: Proc. IEEE-IFIP Conf. on Software Architecture, Montreal (2002)
16. Streitz, N., et al.: i-LAND: An interactive Landscape for Creativity and Innovation. In: Proc. CHI'99, ACM/SIGCHI, pp.120-127 (1999)
17. Tandler, P., Prante, T., Müller-Tomfelde, C., Streitz, N., Steinmetz, R.: ConnecTables: Dynamic Coupling of Displays for the Flexible Creation of Shared Workspaces. In: Proc. of UIST'01, Orlando, Florida, pp. 11–20 (2001)
18. Weiser, M.: The computer for the 21st century. Scientific American, pp. 94–104 (September 1991)