Coupling Interaction Resources in Ambient Spaces: There is More than Meets the Eye!

Nicolas Barralon, Joëlle Coutaz

Université Joseph Fourier
385 rue de la Bibliothèque, BP 53, 38041 Grenoble Cedex France
{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 two complementary conceptual tools to reason about coupling interaction resources. The first tool is a graph theoretic and algebraic notation that can be used to identify the consequents of causal couplings so that the side-effects of the creation of a coupling can be analyzed in a formal and systematic way. The second tool formulates the problem of coupling using an 8 state automaton that models the life cycle of a coupling and provides designers with a structure to verify that usability properties have been satisfied for each state. We conclude with the concept of meta-UI, an overarching interactive system that shows that coupling is only one aspect of a larger problem space.

Keywords: Ubiquitous computing, ambient intelligence, ambient interactive spaces, devices assembly, devices coupling, meta-UI.

1 Introduction

Man is a natural builder. Babies love assembling cubes and objects into complex constructs. TV sets are augmented with high-fidelity loud speakers and wall-size screens to enhance the feeling of "being there". Computer displays are increasingly augmented with additional external screens and exotic input devices such as iStuffs [1], etc. But as of today, human constructs are elaborated from (and for) two different worlds separated with clear boundaries: the computer world (with millions of PC's interconnected over the planet) and the physical world (places, artifacts of all sorts, including cars fitted with hundreds of processors, but still insulated from the computer world). Each one of these worlds has its own well-established interaction paradigms and perceived affordances [17], making it easy to couple objects into useful and usable constructs. As we move to ubiquitous computing and ambient interactive spaces, the boundaries disappear, and the story is not as simple.

In his influential paper on ubiquitous computing, Mark Weiser envisioned technologies that "weave themselves into the fabric of everyday life until they are undistinguishable from it" [23]. The PC, as we use it today, will go out of its box, and will be part of the rest of the world. Many scenarios for ambient computing, including those envisioned by Mark Weiser, praise the power that will result from the interaction between "mixed-reality" (or "embodied-virtuality"). However, with this power arise new problems. Among these problems is how to understand coupling.

Recent research in ambient computing demonstrates that coupling opens the way to unbounded forms of interaction and services. For example, one can couple two objects, such as a wallet and home keys, by shaking them together [10]. As a result an alarm can signal when one is separated from the other. This way, the owner is less likely to forget one or the other along the way. But, how do we know that the keys can be coupled with (and decoupled from) the wallet? How do we know that they can be coupled by shaking them altogether? What should happen when the keys are coupled with a pair of shoes, are then the shoes coupled with the wallet?

This article is a scientific essay on coupling entities with special attention to entities that play the role of interaction resources. In the context of this work, an entity may be physical (denoted P), digital (or numerical, N), or mixed (M). A table, a keyboard are P entities; a keyboard driver is an N entity, a finger tracker is an N entity as well. A mixed entity results from coupling N and P entities. An M entity plays the role of an interaction resource when it allows users to exchange information with a computer system.

This article is structured in the following way: in the next section, we provide a formal definition for the notion of coupling illustrated with two systems serving as running examples. We then build upon analogies with chemistry in the following way: in Section 3, we define the valence of an entity and refer to the compatibility between entities. Then in Section 4, we propose to model mixed entities as N-P molecules, and in Sections 5 and 6, we reason on causal relationships between couplings using formal notations. In Section 7, we detail the life cycle of a coupling then show, in Section 8, how it can serve as a framework for usability investigation. In the last section, we show how coupling interaction resources is only one facet of a more general problem: that of providing end-users with a meta-UI to build, control, evaluate, and ultimately program their interactive space.

2 Coupling Entities

2.1 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 [9], 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:

$$c = (e_1, e_2, e) : \forall f_i \neq f_1 \wedge f_i \neq f_2 \;:\; (f_1 \cap f_i = \emptyset) \wedge (f_2 \cap f_i = \emptyset) \qquad (1)$$
$$\text{where } e_1, e_2 \in \mathbf{E},\; f_1 = func(e_1), f_2 = func(e_2), f = func(e)$$

or as:

$$c = (e_1, e_2, f) \qquad (2)$$

or as:

$$(e_1, \mathrm{c}, e_2) \qquad \text{or} \qquad \mathrm{e}_1 \xrightarrow{\;\mathrm{c}\;} \mathrm{e}_2 \qquad (3)$$

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. Fig. 1 illustrates couplings that we will use as running examples in the following discussion.

2.2 Illustration

I-AM (Interaction Abstract Machine) is a middleware that supports the dynamic coupling of entities such as screens, keyboards and mice, to form a unified interactive space [15]. These entities may be distributed across multiple machines running distinct operating systems including MacOS X and Windows XP. In this space, users can distribute and migrate windows as if they were handled by a single computer[1]. The two screen entities of Fig. 1-a are coupled when the sequence of actions a that bring the screens in close contact is performed (detected by infrared-based proximity sensors). This sequence of actions is similar in spirit to Hinckley's synchronous

[1] The illusion of a unified space is provided at no extra cost for the developer who can reuse the conventional GUI programming paradigm. I-AM is similar in spirit to iRoom and i-LAND. Although iRoom supports heterogeneous workstations, windows in iRoom cannot cross screens boundaries. In i-LAND, windows can cross screens boundaries but the underlying workstations run the same operating system.

gestures [9]. An alternative sequence of actions, inspired from SyncTap [18] called “Click’n Couple”, 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). The function *f* now available 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 (See [15] for details).

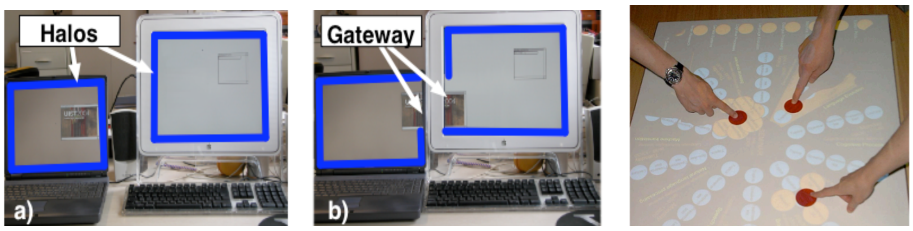


Fig. 1. (a) The PC and the Macintosh screens are decoupled and run two applications. (b) The two screens are coupled to form a single display area by bringing them in close contact. (Halos have been artificially enhanced on the pictures to increase readability.) (c) Partial view of the FAME room. Selectable information (N entities) is rendered as round shape items that match the shape of the physical tokens (form factor compatibility between the N’s and P’s). A flower menu is obtained by placing a token on a round-shape N item. Here, users have opened three “flower menus”.

The FAME table (see Fig. 1-c) is a component of an augmented room that supports the collaborative exploration of information spaces [11]. A table and two walls play the role of output interaction resources. Each one is coupled to its own video-projector to display digital information (N entities). In addition, the table is coupled to a camera that senses colored, 4 cm wide round shape tokens made of plastic. A token (a P entity) is coupled to the tracker of the table (an N entity), when the action *a* “put token down on the table” is performed. The coupling “token-tracker” results in an M entity that plays the role of an input interaction resource. This M entity is coupled with a round shape digital entity displayed on the table when the token (i.e. the P component of M) is brought over the entity. A “flower menu” pops up around the token to inform the user that the function *f* “information selection” is now available. The user can now select digital information by moving the token to the appropriate petal of the flower.

Our definition, which involves two source entities, does not imply that coupling is exclusive. An entity may be coupled to several other entities. The possible configurations that can result depend on the valence and the compatibility of the entities involved. These are discussed next.

3 Valence of an Entity and Compatibility between Entities

The *valence* of an entity is an integer that measures the maximum number of entities that can be bound with it at a given time. For example, in I-AM as well as for Hinckley’s tablets, the valence of a screen is 4: a screen can be coupled to a maximum of 4 screens (one on each side). The valence of a FAME token is 2: at a given time, it can be coupled to at most 1 table and 1 item of digital information.

Compatibility has been used in many ways in HCI to motivate design decisions [4, 12, 14, 24]. Here, the *compatibility* between two entities denotes the capacity for these entities to be coupled provided that they satisfy a set of constraints that apply to both of them. Constraints may apply to:

- Physical form factors. In I-AM, surfaces that can be coupled must be rectangular. In FAME, tokens must be round and red in order to be tracked by the system.
- Software discoverability and interoperability. In I-AM, MacOS and/or Windows platforms are compatible, but Linux is not supported.
- Cognitive compatibility at multiple levels of abstraction from physical actions to intentions and goals. In FAME, selectable information is rendered as round shape items that match the shape of the physical tokens (to enforce their perceived affordance) (see Fig. 1-c). In its current implementation, the FAME tracker is able to track about 12 tokens simultaneously with an 80ms latency on a dual PowerPC 7400 (G4) 1.4 Ghz machine. As a result, if more that 12 tokens are coupled with the table, then the latency of the system is not sufficient to support the feeling of tightly coupled interaction [3]. Adding a 13th token is technically feasible, but not compatible with human expectation at the physical action level.
- Contextual compatibility. The context in which the coupling of entities is created and evolves can influence their compatibility. In this article, we focus on coupling under the control of the user. Dobson and Nixon, in [7], provide approaches for adapting a system according to the context of use. Their approach can be applied to our problem, where compatibility between entities depends on context.

Valence and compatibility between entities determine conditions for the realization of couplings. In the next section, we illustrate the use of these characteristics for the construction of mixed entities.

4 Mixed entities as N-P molecules

P's and N's can be coupled in a number of ways to form new mixed entities. In particular, two basic mixed entities, denoted respectively as **P-N** and *P-N*, can be coupled by the way of their N component, or their P component, or by a mix of them. As shown in Fig. 2, one may obtain the following configurations: **N-P**-*P-N*, **P-N**-*N-P*, **N-P**-*N-P*, and **P-N**-*P-N*. Are all of them possible? The answer depends on the valence of the components and the compatibility between them.

N — **P** — *P* — *N*   **P** — **N** — *P* — *N*

N — **P** — *N* — *P*   **P** — **N** — *N* — *P*

Fig. 2. Basic N-P constructs.

By analogy with chemistry, mixed entities are *N-P molecules* elaborated from any number of N and P atoms whose configuration satisfies their valence and compatibility. Intituively, form factors matter in N-P-P-N configurations, whereas software compatibility prevails in P-N-N-P constructs. For example, in I-AM, only rectangular screens can be coupled. In any case, the resulting assembly must be cognitively compatible with user's physical abilities and expectation. The assembly of N-P molecules may be performed either at design time, or at run time. We believe that the design/run time distinction is important in the context of ambient computing where dynamic reconfiguration under human control is key.

Intrinsically-mixed entities are those for which the coupling of numerical and physical entities has been performed before hand by designers so that end-users can exploit them directly without performing any additional binding. For example, a PDA is an intrinsically-mixed entity: it binds together digital and physical components that have been pre-packaged into a working unit.

Alternatively, entities are *constructively-mixed* when the end-user is in charge of performing some coupling before being able to use them. A FAME token must be coupled to the table in order to be used as a pointing device. Thus in FAME, pointing devices are constructively-mixed entities. Similarly, an external keyboard, which is a physical entity, needs to be coupled with a driver to play the role of an input interaction resource. Clearly, constructively-mixed entities can include entities that are intrinsically-mixed. The Nabaztag shown in Fig. 3, is an example of this type of assembly.

The Nabaztag (which means "rabbit" in Armenian) is an intrinsically-mixed entity built from a 9 inches tall plastic bunny shape object with a loud-speaker, moving ears, and colored lights that pulsate on its nose and belly. It includes a Wi-Fi connection to the Internet so that it can be coupled to Internet services such as the weather forecast, inter-personal messaging, and mood expression (the rabbit has a mood!). Using a Web server on a PC, users can couple any number of N entities (i.e. Web services) to the N component of the Nabaztag provided that these services interoperate with the N component. The result is a well-balanced star-like N composition coupled to a single P.

However, one may wonder how a single P can (simultaneously) render the state of a large number of N services and allow users to manipulate this state through a limited number of input means (i.e., the ears of the plastic rabbit and a push button located at the top of its head). One possible venue is for the Nabaztag to borrow interaction resources of the interactive space by the way of causal couplings.

5 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 as *consequents* of the *causal* coupling. In the following discussion, we use the Nabaztag as an informal illustration of the problem followed by two formal notations to reason about causal couplings and their consequents.

5.1 Illustration

Fig. 3 illustrates causal relationships between couplings when the Nabaztag is coupled with a smart home. The N components of this smart home include a presence detector, a surveillance system, and an IP-device discovery facility. It includes a number of intrinsically-mixed entities such as an augmented IP fridge and an IP answering machine. When the owner is away, any intrusion or abnormal situation is notified to the owner via the mobile phone.

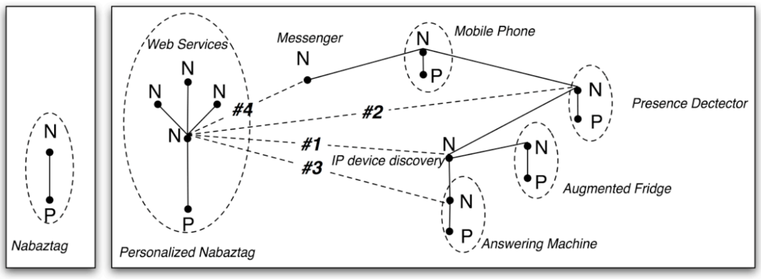


Fig. 3. On the left, the original off-the shelf Nabaztag is an intrinsically-mixed entity. On the right, the personalized Nabaztag used in a smart home becomes a constructively-mixed entity. Couplings #2, #3, and #4 are the consequents of the causal Coupling #1.

The Nabaztag plays the messages sent by buddies using its speaker-phone, but it is unable to remember them. Thus, when there is nobody at home, one would like the Nabaztag to forward incoming messages to the recording facility of the answering machine and/or to the mobile phone. Because, the Nabaztag is an IP device, it can be detected automatically by the IP-device discovery facility resulting in the creation of Coupling#1. In turn, Coupling#1 entails three consequents (Couplings#2, #3 and #4) in order to provide the forward-to service: Coupling#2 to determine whether there is somebody at home, Coupling#3 to use the recording facility of the answering machine, and Coupling#4 to forward messages to the mobile phone.

Coupling the Nabaztag to the smart home raises a number of issues, in particular: what consequent couplings of a causal coupling make sense? The following formal analysis provides a systematic framework for answering this question using a graph theoretic notation and an algebraic notation.

6.2 Formal Analysis with a Graph Theoretic Notation

We represent couplings using the graph notation (3) introduced in 2.1 where nodes denote entities, and where edges express the existence of couplings. Symbols "*" and "=" denote causal and consequent couplings respectively. 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*. The "?" symbol denotes the couplings that are under evaluation (i.e. keeping them as consequents or rejecting them has yet not been decided). To express their transitory state, causal couplings, as well as consequents and undecided couplings are represented as dotted edges. Let:

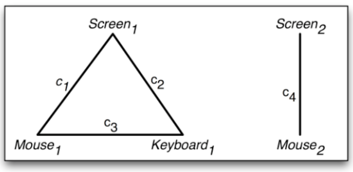
- *EDGE* be the set of edges of the graph under consideration.
- $r_1(c)$ (resp. $r_2(c)$) be the first (resp. the second) interaction resource involved in the coupling *(r1, c, r2)*.
- $F(r_1, r_2)$ be the set of function that result from (r1, c, r2).
- *Compatible*(f_1, f_2, f_3) returns TRUE if the 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)
```

To illustrate the algorithm, let's consider the initial configuration of couplings represented in Fig. 4: on the left image, *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 enlarged display function), *c5* is performed (for example, by a proximity detection service).



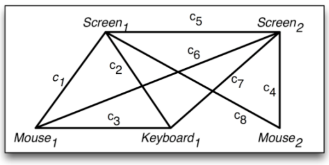


Fig. 4. Left image: the initial configuration that represents a private workstation. Right image: final configuration that corresponds to a public configuration where couplings *c6*, *c7* and *c8* are the consequents of the causal coupling *c5*.

The final configuration that results from the causal coupling *c5* is shown by the rightmost graph of Fig. 4: 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 would be owned by two distinct users who want to collaborate via a unified space, the compatibility functions would be different resulting in a distinct final configuration.

Fig. 5 shows the successive steps that lead to the final configuration of Fig. 4. Fig. 5 (top left) corresponds to the generations 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. 5 (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. 5 (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. 5, bottom center and bottom right). To summarize, the causal coupling c_5 has three consequents: c_6, c_7, and c_8.

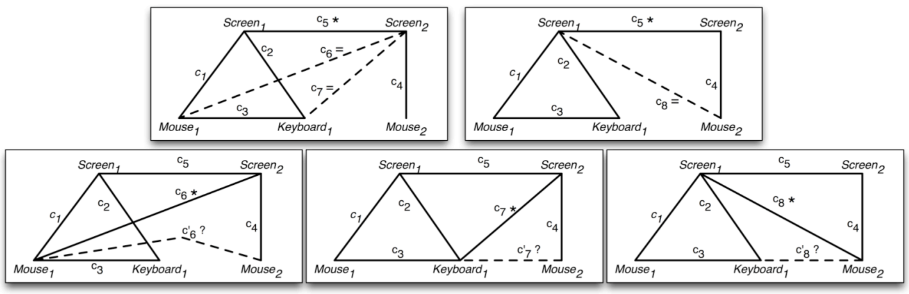


Fig. 5. Evaluation steps resulting from the causal coupling c5.

6.3 Formal Analysis with an Algebraic Notation

Our algebraic notation is based on two operators over couplings:

- The generation operator: *
- The union operator: +
- * is distributive over +
- The priority of * is superior to that of +.

Then, the expression $[c_1 + c_2 + \ldots +c_n]$ denotes the set of couplings $c_1, c_2, \ldots ,c_n$ that exist within a particular system. The creation of a new coupling c_p is a two-step process:

- First, c_p is added to the set by applying the *insertion rule* such that:
  $c_p + [c_1 + c_2+ \ldots + c_n] = [c_p + c_1 + c_2+ \ldots + c_n]$
- Then, the consequents of c_p are computed using *the generation rule* such that:
  $c_p *[c_p + c_1 + c_2 + \ldots +c_n] = c_p * c_p + c_p * c_1 + c_p * c_2 + \ldots + c_p * c_n$

Evaluating $c_p * c_p + c_p * c_1 + c_p * c_2 + \ldots + c_p * c_n$ is to evaluate each one of the terms $c_p * c_i$:

- $c_p * c_p = \varnothing$ (a coupling cannot be the consequent of itself).
- $c_p * c_i = (r_{p1}, r_{p2}, f_p) * (r_{i1}, r_{i2}, f_i)$

c_p and c_i are *transitive* if and only if (iif)
$(r_{p1}=r_{i1} \vee r_{p1}=r_{i2} \vee r_{p2}=r_{i1} \vee r_{p2}=r_{i2}) \wedge Compatible(f_p, f_i, f) \wedge c_p \neq c_i$
otherwise, c_p and c_i are intransitive.

The condition $(r_{p1}=r_{i1} \vee r_{p1}=r_{i2} \vee r_{p2}=r_{i1} \vee r_{p2}=r_{i2})$ expresses the fact that transitive couplings share one interaction resource. This is equivalent in the graph notation to the paths of length 2 that contain $r_1(c_p)$ and $r_2(c_p)$.

If c_p and c_i are transitive then:
$c_p * c_i = (r_{p1}, r_{p2}, f_p) * (r_{i1}, r_{i2}, f_i) = c_{res}$ where $Compatible\ (f_p, f_i, f)= TRUE$
$c_{res} = (r_{p2}, r_{i2}, f)$ if $r_{p1}=r_{i1}$
$c_{res} = (r_{p2}, r_{i1}, f)$ if $r_{p1}=r_{i2}$
$c_{res} = (r_{p1}, r_{i2}, f)$ if $r_{p2}=r_{i1}$
$c_{res} = (r_{p1}, r_{i1}, f)$ if $r_{p2}=r_{i2}$
$Transitive(c_p, c_i) = TRUE$

If c_p and c_i are intransitive then:
$c_p * c_i = (r_{p1}, r_{p2}, f_p) * (r_{i1}, r_{i2}, f_i) = \varnothing$
$Transitive(c_p, c_i) = FALSE$

The algorithm detailed in Fig. 6, makes it explicit the generation of the consequents of a causal coupling. Let's apply the algorithm to the example of Fig. 4:

Initial configuration : $[c_1 + c_2 + c_3 + c_4]$
Causal coupling : c_5

Insertion rule:
$c_5 + [c_1 + c_2 + c_3 + c_4]= [c_5 + c_1 + c_2 + c_3 + c_4]$

Generation rule:
$$c_5 * [c_5 + c_1 + c_2 + c_3 + c_4] = c_5 * c_5 + c_5 * c_1 + c_5 * c_2 + c_5 * c_3 + c_5 * c_4$$
$$= \varnothing \quad + \quad c_6 \quad + \quad c_7 \quad + \quad \varnothing \quad + \quad c_8$$

Insertion rule:
$c_6 + c_7 + c_8 + [c_5 + c_1 + c_2 + c_3 + c_4] = [c_1 + c_2 + c_3 + c_4 + c_5 + c_6 + c_7 + c_8]$

Generation rule:

$(c_6 + c_7 + c_8) * [c_1 + c_2 + c_3 + c_4 + c_5 + c_6 + c_7 + c_8] =$

1. $c_6*c_1+c_6*c_2+c_6*c_3+c_6*c_4+c_6*c_5+c_6*c_6+c_6*c_7+c_6*c_8$
2. $+c_7*c_1+c_7*c_2+c_7*c_3+c_7*c_4+c_7*c_5+c_7*c_6+c_7*c_7+c_7*c_8$
3. $+c_8*c_1+c_8*c_2+c_8*c_3+c_8*c_4+c_8*c_5+c_8*c_6+c_8*c_7+c_8*c_8$

1. $\varnothing + \varnothing + \varnothing + \varnothing + \varnothing + \varnothing + \varnothing + \varnothing$ (couplings are intransitive)
2. $+ \varnothing + \varnothing + \varnothing + \varnothing + \varnothing + \varnothing + \varnothing + \varnothing$ (couplings are intransitive)
3. $+ \varnothing + \varnothing + \varnothing + \varnothing + \varnothing + \varnothing + \varnothing + \varnothing$ (couplings are intransitive)

```
Function []Coupling GenerationRule(
            //causal coupling provoking the generation of consequents
            Coupling causalCoupling,
            //set of effective couplings
            Coupling []effectiveCouplings){

   //insertion of the causal coupling to effectiveCouplings
   Coupling []couplings = {effectiveCouplings, causalCoupling } ;

   //init workingList, the list of couplings onto which
   //the generation rule must be applied
   Couplage []workingList = {causalCoupling } ;

   //traverse the workingList for possible generation
   For (int i=0 ; i< workingList.length ; i++){
      //get item of the list
      Coupling c1 = workingList[i] ;

   //traverse the effectiveCouplings
   For (int j=0 ; j<effectiveCouplings.length ; j++){
   //get item of the list
   Coupling c2 = effectiveCouplings [j] ;

   //test the transitivity between c1 and c2
         If (Transitive(c1, c2)){
            //generate a new coupling
            Coupling gen = composition(c1, c2) ;

            //insert new coupling to the effectiveCouplings
            effectiveCouplings [effectiveCouplings.length]= gen ;

            //the generation rule applies to the new created coupling
            workingList [workingList.length]= gen ;
         }
      }
   }
   Return effectiveCouplings;
}
```

Fig. 6. Generation of the consequents of a causal coupling.

Thus, given the compatibility rules elicited for that particular situation, the final configuration that results from the causal coupling c5 is: $[c_1 + c_2 + c_3 + c_4 + c_5 + c_6 + c_7 + c_8]$. Because compatibility evolves over time, the life cycle of a coupling cannot be a simplistic dual state (coupled/uncoupled) Finite State Automaton (FSA). This aspect is discussed next.

7 Life Cycle of Couplings

As shown in Fig. 7, the life cycle of a coupling includes eight states where a state is defined by the conjunction of the following set of predicates:

- *Coupled (e1, c, e2)* = TRUE if and only if f≠∅ where f is the set of functions that results from (e1, c, e2). If f=∅, then Coupled (e1, c, e2) = FALSE and NotCoupled (e1, c, e2) =TRUE.
- *Locked (e1, c, e2)* = TRUE if the state of e1 does not permit to modify the state of (e1, c, e2). This predicate can be used to express that e1 is not "socially" or "technically" available to enter or exit its coupling c with e2. For example, a user does not want to connect his private PDA to a public screen. The state of (e1, c, e2) is kept unchanged until Locked (e1, c, e2) = FALSE or NotLocked (e1, c, e2) = TRUE.
- *Couplable (e1, c, e2)* is an expression of predicates P, where P≠ Coupled (e1, c, e2) and P≠Locked (e1, c, e2). This expression specifies the conditions (different from Coupled (e1, c, e2) and Locked (e1, c, e2)) that are necessary for (e1, c, e2) to happen. For example, valence and compatibility can be used to express Couplable. Symmetrically, Uncouplable expresses the conditions (different from Coupled (e1, c, e2) and Locked (e1, c, e2)) that are necessary for (e1, c, e2) to end.

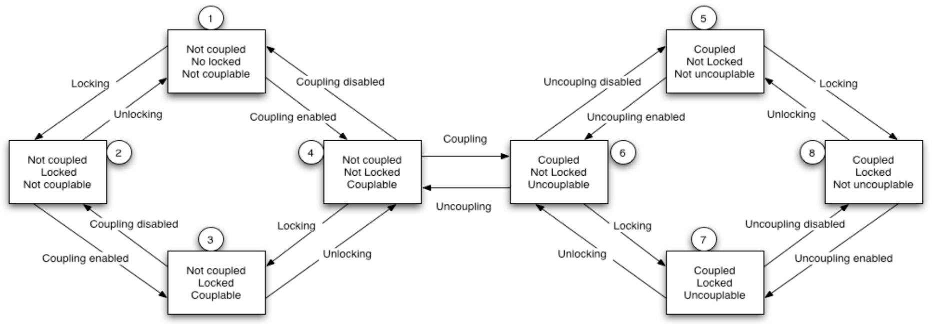


Fig. 7. Coupling (e1, c, e2) as a Finite State Automaton. For the sake of readability, the transitions between states 1 and 3, 2 and 4, 5 and 7, 6 and 8 are not represented.

The automaton shown in Fig. 7 corresponds to the coupling (e1, c, e2)[2]. It is comprised of two sub-automata: one that includes the states 1, 2, 3, 4 where *Coupled (e1, c, e2)* is TRUE, the other one that covers the states 5, 6, 7, 8 where *Coupled (e1, c, e2)* is FALSE. States 4 and 6 serve as gateways between the two sub-automata. State 4 corresponds to the situation where all the conditions for realizing (e1, c, e2) are satisfied. Only a coupling request event is missing to enter state 6.

[2] A similar automaton models (e2, c, e1).

Because of the multitude of states, the study of such automata provides fertile ground for usability investigation.

8 The Life Cycle as an Analytic Framework for Usability

As an illustration, we analyze I-AM and the FAME table with two of the IFIP properties: observability and predictability [8]. Other usability frameworks (such as the Cognitive Walkthrough [22], Nielsen's [16] or Bastien-Scapin's criteria [2]) could be used as well.

8.1 Observability of Couplings in the FAME Table

Observability is the ability for the user to evaluate the internal state of the system from its perceivable representation. When applied to the life cycle of a coupling, this property requires that every state of the automaton be made observable to users.

As a counter-example, let's consider the coupling of the FAME tokens with the N entities displayed on the table. Let t_1 and t_2 be two tokens, and i_1, a selectable N item projected on the table. At the beginning, the user is holding the tokens in his hands, and i_1 is rendered as a round shape graphics. Thus, (t_1, c, i_1) is in State 4. By dropping t_1 on i_1, one couples t_1 with i_1 making the select function available: the automaton for (t_1, c, i_1) enters State 6. To make this state observable, i_1 opens itself as a flower where each petal is couplable to t_1. On the other hand, this action locks i_1 for tokens different from t_1: (t_1, c, i_1) then enters State 7. As a result, dropping t_2 on any petal of i_1 will have no effect (since (t_1, c, i_1) is locked) but dropping t_2 on another selectable item i_2 would work correctly.

Two semi-formal user studies with 30 subjects unfamiliar with the FAME table showed that some people selected the petals using additional tokens instead of traversing the flower menu with the coupled token. If we had this analytical model at the time of the development of FAME, we would have been able to spot this problem and take corrective actions such as making the *Locked* state observable or allowing coupling a flower menu with multiple tokens.

8.2 Observability of Couplings in I-AM

In Fig. 1-a, two applications are running on two independent workstations. The closed blue halo that outlines each screen denotes the possibility for currently uncoupled screens to be coupled (State 4 is made observable). The absence of halos would mean that the screens are not couplable. On the other hand, the distinction between States 1 or 2 (locked/unlocked) is not observable which may cause a problem in a collaborative situation. As shown in Fig. 1-b, once the screens are coupled, the new shape of the halo indicates the gateway through which windows can migrate between the two screens (State 6 is made observable).

8.3 Predictability of Couplings in I-AM

Predictability is the ability for the user to determine the effect of future actions based on past interaction history. Applied to coupling, users should be able to anticipate the set of functions *f* that will result from the set of actions *a*.

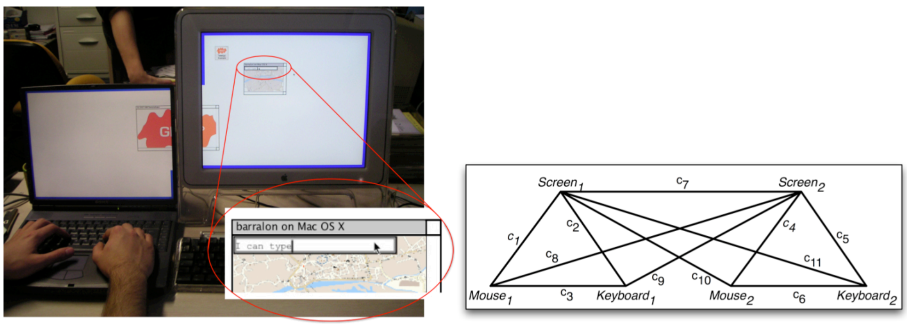


Fig. 8. Entering characters in a text field located on a Macintosh screen using a PC keyboard: to do so, the user has selected the text field with the PC touchpad (left). The corresponding configuration (right) that results from the causal coupling c_7 between the two screens

I-AM preserves the conventions of the GUI paradigm. Windows can sit between two coupled screens although these screens may be connected to different workstations and may differ in resolution and orientation. Mice and keyboards are coupled to provide the input focus function. But, can users predict the final configuration shown in Fig. 8 (right) that results from coupling the two screens? This configuration 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.

Suppose that the user has selected the input text field displayed on the Macintosh screen using the mouse-keyboard of the Mac. The user can then enter text with the Macintosh keyboard. So far, the system behavior is compliant with GUI conventions: in this regard, predictability is satisfied. On the other hand, can the user predict the situation depicted in Fig. 8 (left): the input text field is coupled to the mouse-keyboard of the Macintosh as well as to the mouse-keyboard of the PC (as a result of a PC mouse click in the text field). In this situation, characters can be entered simultaneously from any keyboard. What will happen when the screens are decoupled? This is where things get complex with regard to predictability even in simple situations like the one described below.

Let S1 be a screen coupled by construction (i.e. GUI conventions legacy) to a PC workstation and a mouse M. Let S2 be a screen connected to a second computer with no input device. S1 is now coupled to S2 by bringing S1 and S2 close to each other. According to the I-AM model, M can get coupled to S2 as well: it can be used to modify the information space mapped on S2. Thus the cursor of M can be mapped on S2. Can the user predict what will happen if S1 is uncoupled from S2 while the cursor of M is mapped on S2? Will M be uncoupled from S1 and stay coupled with S2? Or, alternatively, will it follow its home surface? If so, where will the cursor re-appear on

S1? This type of problem was spotted by the developers of PointRight [13] who stated that “a free-space device (such as a wireless mouse) needs an explicit starting screen”. Translated into our framework, this means that when a wireless mouse is dynamically coupled to the interactive space, its associated cursor must be mapped onto a predefined home screen in order to support predictability.

As this example shows, by transitivity, multiple entities are bound together to form an interactive space whose functionalities depend on the set of functions that each coupling delivers. Do these functions, all together, form a “consistent story” for the user? Since the management of the interactive space corresponds to the interplay of multiple automata, how many of them can the system (and the user) reasonably handle at a time? How can this be controlled by end-users? We propose the concept of meta-UI as a coherent framework to address these issues.

9 The Concept of Meta-UI

A *meta-UI* is an interactive system whose set of functions is necessary and sufficient for end-users to control and evaluate the state of an 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 space. In the context of this article, a meta-UI *is not* an abstract model, nor a language description, whose transformation/interpretation would produce a concrete effective UI. It is an over-arching interactive system whose role is to ambient computing what desktops and shells are to conventional workstations.

The notion of meta-UI is described in detail in [5]. As summarized in Fig. 11, a meta-UI is characterized by its *functional coverage* in terms of *services* (including coupling), and *object types* (including mixed entities). In turn, the services and objects are invoked and referenced by the way of *interaction techniques* (or UI) that provide users with some *level of control*: who has the initiative (users or the system?), and once a service is launched what kind of control do users have (observability only, traceability only, or dynamic and incremental control?).

An interaction technique is a language (possibly *extensible*) characterized by the *representation* (vocabulary) used to denote objects and services as well as by the way users can construct sentences and assemble these sentences into programs. Given the role of a meta-UI, the elements of the interaction technique of the meta-UI must cohabit with the UI’s of the domain-dependent services that it governs: these elements may be embedded within the UI of the domain-dependent services, or they may be external to the UI of these services. Using the Nabaztag and smart home example, we illustrate the concept of meta-UI for coupling.

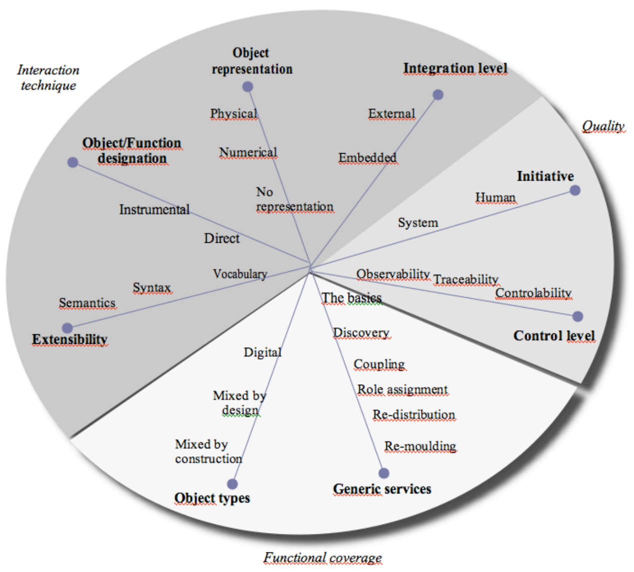


Fig. 11. The dimension space of Meta-UI's.

Forwarding messages to the answering machine or to the distant SMS, may be pre-programmed within the N component of the Nabaztag or the Nabaztag may not hold this program at all. In the first case, the program may be triggered when Coupling#1 is performed. As mentioned above, this coupling (and its consequents) may be performed on the system initiative, and pursued autonomously with no human control. Alternatively, the user may be kept in the loop: from implicit, the process becomes explicit. The level of control that end-users have on couplings is fundamental. At minimum, *observability* should be supported, i.e. users should be able to evaluate the internal state of the coupling from its current perceivable representation. The next step is *traceability* by which users can observe the evolution of the coupling over time, but they cannot modify this evolution. With *controllability*, users can observe, trace, and intervene on the evolution of couplings. They can even program couplings.

For example, if the Nabaztag does not host the "forward-to" program, the smart home may include an *end-user development environment* (EUDE) that would allow users to build programs, i.e. new N entities, to modify the behavior of the smart home. Powerful meta-UI's must include an EUDE. Based on visual programming, tools like Jigsaw support the construction of simple sentences such as "if someone rings the bell, take a picture and send it to my PDA"[19]. Using a rule-based paradigm, a CAPpella [6] and iCAP [20] go one step further by allowing end-users to elaborate programs to control the behavior of ambient spaces. End User Programming has been around for many years [21]. It is now becoming a key challenging issue to be addressed in the near future.

10 Conclusion

Coupling is not a new phenomenon. In the GUI computer world, most couplings are pre-packaged and immutable. Typically, mice are coupled with display screens, while mice and keyboards are coupled for the input focus function. As a consequence, coupling is taken for granted by HCI designers and developers. However, in ambient computing, there is more than meets the eyes: a coupling is not an insulated dual state phenomenon.

First, coupling two interaction resources requires that they meet a number of conditions including their mutual compatibility, valence, and availability. Are these conditions observable, predictable, traceable, and controllable? We propose an 8 state automaton that models the life cycle of a coupling and that provides designers with a framework to verify whether usability properties are satisfied for each state of a particular coupling.

Second, the creation of a new coupling may have side effects on existing couplings. In this article, we have not investigated the destruction of couplings. On the other hand, we propose two formalisms, using a graph theoretic and an algebraic notation, to reason about the consequents of causal couplings in a systematic way. Here, we use the compatibility between the functions returned by a consequent and the functions provided by its two neighbors. Other rules could be used. In any case, are the consequents of a causal coupling observable, predictable, traceable, and controllable?

To provide a preliminary answer, we propose the concept of meta-UI as a unifying overarching interactive system that leads into end-user development. Ideally, the yet-to-be-invented meta-UI will allow users to construct and program powerful N-P molecules of any shape that will make sense for them. Progressively, patterns like the star-like construct of the Nabaztag will emerge. We are only at the beginning. And coupling is only one aspect of this large problem space.

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

References

1. Ballagas, R., Meredith, R., Stone, M., Borchers, J.: iStuff: APhysical User Interface Toolkit for Ubiquitous Computing Environments, in Proc. Of CHI 2003, Ft.Lauderdale, Florida, 2003, pp. 537-544.
2. Bastien, J.M.C., Scapin, D.L.: Critères Ergonomiques pour l'Évaluation d'Interfaces Utilisateurs, Technical report 1993. INRIA. 1993.
3. Bérard, F.: Vision par Ordinateur pour l'Interaction Homme-Machine Fortement Couplée, Thesis, Université Joseph Fourier, Novembre 1999. 200 pp.
4. Card, S.K., Mackinlay, J.D., Robertson, G.: The design space of input devices, in Proceedings of the SIGCHI, Seattle, Washington, United States, 1990, pp.117-124.
5. Coutaz, J.: Meta-User Interface for Ambient Spaces, In TAMODIA'06, Hasselt, Belgium, october 2006, 8 pp.

6. Dey, A. K., Hamid, R., Beckmann, C., Li, I., and Hsu, D.: a CAPpella: programming by demonstration of context-aware applications. In Proceedings of the SIGCHI, 2004, CHI '04. ACM Press, New York, NY, pp. 33-40.
7. Dobson, S., Nixon, P., More principled design of pervasive computing systems. In Human computer interaction and interactive systems. Vol. 3425 of LNCS. Springer Verlag. 2004.
8. Grahm, Ch., Cockton, G.: Design Principles for Interactive Software. Chapman & Hall, London, 1996.
9. Hinckley, K., Synchronous gestures for multiple persons and computers, in Proc. of UIST'03, Vancouver, Canada, 2003, pp. 149-158.
10. 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 2001, Atlanta, Georgia, 2001, pp. 116-221.
11. IST-2000-28323 FAME European project. http://isl.ira.uka.de/fame/
12. Jacob, R. J. K., Sibert, L. E., McFarlane, D. C., and Mullen Jr., M. P.: Integrality and separability of input devices, ACM Transactions on Computer-Human Interaction, 1(1):3-26,1994.
13. Johanson, B. , Hutchins, G., Winograd, T., Stone, M. PointRight: experience with flexible input redirection in interactive workspaces. Proceedings of the 15th annual ACM symposium on User interface software and technology UIST 2002 France, pp.227-234.
14. Kurtenbach, G., Baudel, T.: Hypermarks: Issuing Commands by Drawing Marks in Hypercard, vol. Adjunct Proceedings, ACM, Proc. ACM SIGCHI, p. 64, 1992
15. Lachenal, C. Modèle et Infrastructure Logicielle pour l'Interaction multi-instrument, multi-surface. Thesis of University Joseph Fourier, december 2004.
16. Nielsen, J. Usability engineering at a discount, in Salvendy, G., and Smith, M.J. (Eds.), Designing and Using Human-Computer Interfaces and Knowledge Based Systems, Elsevier Science Publishers, Amsterdam, 1989,394-401.
17. Norman, D.: "User Centered System Design", Lawrence Erlbaum, 1986.
18. Rekimoto, J., Ayatsuka, Y., Kohno, M.: SyncTap: An Interaction Technique for Mobile Networking, in Proc. of MOBILE HCI'03, Udine, Italy, 2003, pp.104-115.
19. Rodden, T., Crabtree, A., Hemmings, T., Koleva, B., Humble, J., Akesson, K.P., and Hansson, P. Configuring the Ubiquitous Home. In Proc. of the 2004 ACM Symposium on Designing Interactive Systems (DIS 04) (Cambridge,Massachusetts), ACM Press, 2004.
20. Sohn, T. and Dey, A.: iCAP: an informal tool for interactive prototyping of context-aware applications. In CHI '03 Extended Abstracts on Human Factors in Computing Systems (Ft. Lauderdale, Florida, 2003). CHI '03. ACM Press, New York, NY, 974-975.
21. Sutcliffe, A., Mehandjiev, N.: End-User Development. Communication of the ACM, special Issue on End-User Development, ACM publ., sept. 2004.
22. Wharton, C., Rieman, J., Lewis, C., and Polson, P.: The Cognitive Walkthrough Method: A Practitioner's Guide. In Usability Inspection Methods, J. Nielsen and R.L. Mack (Eds.), New York: John Wiley & Sons, 1994, pp.105-141.
23. Weiser, M.: The computer for the 21st century. Scientific American, September 1991, pp. 94–104.
24. Zhai, S.: User performance in relation to 3D input device design. SIGGRAPH Comput. Graph. 32, 4 (Nov. 1998), 50-54.