End-User Programming and the Intrinsic Complexity of Networked Artefacts

Joëlle Coutaz
Grenoble Informatics Laboratory
University Joseph Fourier
385 rue de la Bibliothèque
38041 Grenoble Cedex 9
+33 4 76 51 48 54
joelle.coutaz@imag.fr

ABSTRACT
In this article, we propose a model for networked artifacts inspired from molecular chemistry. It demonstrates the intrinsic complexity of the domain illustrated with unsolved problems such as mastering the semantics of networked artefacts. Based on this model, we identify similarities with the service-oriented computing paradigm and suggest possible avenues for collaboration between Software Engineering and researchers in EUD including: semantic alignment between end-user programming languages and service descriptions, and human-service interaction as a transversal issue in software design.

Categories and Subject Descriptors
D.2.2 [**Software Engineering**]: Design Tools and Techniques – *User interfaces, evolutionary prototyping.*

General Terms
Design, Human Factors.

Keywords
End-user development, end-user programming, ubiquitous computing, service-oriented computing.

1. INTRODUCTION
Ubiquitous computing promises unprecedented empowerment from the flexible and robust combination of software services with the physical world. For the HCI research community, this means that end-users will be able to shape their own interactive spaces and build imaginative new forms of interaction and functionalities that were not anticipated by the system's designers. For the software engineering community, service combination entails the development of multi-scale computing fabrics that will autonomously adapt to their environment. Here, autonomy denotes self-deployment, self-reconfiguration, self-protection, and self-repair … without human intervention. In other words, in HCI, the human is at the core of the computation loop, where-as, in software engineering, the human is removed from the loop.

Although this duality of viewpoints between HCI and Software Engineering has been recurrent for years, ubiquitous computing makes it critical. Ubiquitous computing will not hold its promise as long as "system-ers" and "human-ers" keep working separately with different scientific objectives. The challenge is not to build autonomous systems, but to protect end-users from software complexity by increasing system autonomy and at the same time provide them with the appropriate services and means for preserving values such as control, freedom, and creativity.

In this position paper, I address one particular complex issue, at the core of ubiquitous computing: the dynamic interconnection of "smart devices". I propose to address this complexity using an analogy with chemistry where a smart artefact is modeled as a composition of physical and digital atoms whose configuration evolves under particular conditions. From this vantage, I will seek to unify the HCI and software engineering goals using end-user development for the HCI side and the service-oriented paradigm for the software engineering side.

2. SMART ARTEFACTS AS CHEMICAL MOLECULES

2.1 The Model
A smart artefact (also called a mixed-reality object or an augmented object) is a chemistry-inspired assembly made from two sorts of elements – digital atoms (D) and physical atoms (P) whose bonds correspond to communication mechanisms or physical attachments. As in chemistry, reaction (or a chain of reactions) results from the confluence of particular events, and produces new smart artefacts.

A smart artefact may break down into simpler artefacts and atoms, or it may be coupled with other artefacts and atoms into a more sophisticated artefact. Alternatively, only the internal geometry of the artefact may evolve. The confluence of events that provokes a reaction denotes a change in context as introduced by the ubiquitous computing community. These events are generated either by humans, by the physical environment, or by the system itself (to guarantee service continuity, for example).

WEUSE IV, May 12, 2008, Leipzig, Germany.

Thus, the simplest smart artefact is composed of a single D atom coupled with a single P atom. Software services are composed of D atoms only, whereas conventional objects of the real world are strictly built from P atoms. An interactive space (e.g., a smart home), may be a unique but large molecule of D and P elements, or it may be a set of molecules each one defining a smaller bubble, but prone to reconfiguration.

As in chemistry, the nature of the events that trigger a reaction has an impact on the resulting product. For example, in Hinckley's dynamic display tiling [4], users obtain different arrangements depending on the way the tablets are bumped together. According to our model, each tablet is a [P–D] molecule where P denotes the rectangular screen and D represents the set of services that are currently mapped on this screen. If one tablet rests flat on a desk surface, and a second tablet is bumped into the base tablet, then the resulting artefact is a larger display that shows the content of the base tablet (see Figure 1-a). Alternatively, if the two tablets are bumped symmetrically, the tablets perform a mutual exchange of information (see Figure 1-b). In case a), a bond is created between the two P's to form a larger screen, and the D that was associated with the moving screen becomes a free atom. In case b, the P's are linked as in case a, but their D's have been exchanged.

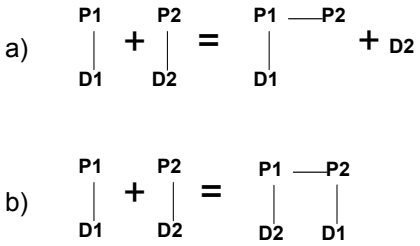

Figure 1. Hinckley's tablets. a) the resulting artefact when a second tablet is bumped into the base tablet. b) the resulting artefact when the tablets are bumped symmetrically.

The Nabaztag shown in Figure 2 is another interesting form of [P – D] molecule.

2.2 The Nabaztag Exemplar

The Nabaztag is an information appliance[1] that results from the binding of a P atom with a D atom: the P_{nab} atom is a 9 inches tall plastic bunny shape including a loud-speaker, moving ears, colored lights that pulsate on its nose and belly, and a Wi-Fi card. Its D_{nab} atom implements some service-oriented protocol over IP. The [P_{nab} – D_{nab}] molecule corresponds to the "virgin" Nabaztag as users get it from the store.

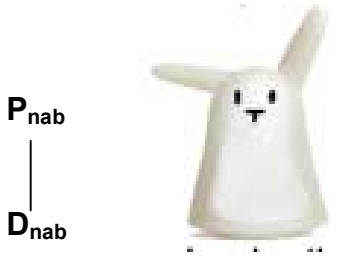

Figure 2. The "virgin" Nabaztag as a [P – D] molecule.

[1] Nabaztag means "rabbit" in Armenian.

To become a working information appliance, the Nabaztag must be registered using a Web server on a personal computer. The registration process is supported by the [P_{PC} – D_{web} – D_{reg}] molecule where P_{PC} denotes the PC, D_{web} the Web server, and D_{reg} the registration software service available from the Web server (see figure 3). By filling in forms on the PC, the user provokes a reaction between the original Nabaztag molecule and the registration molecule resulting in the creation of a bond between atoms D_{nab} and D_{reg} as well as bonds between D_{web} and the D_s services to which this particular instance of Nabaztag is able to subscribe. These services include weather forecast, inter-personal messaging, mood expression (the rabbit has a mood!), etc.

.

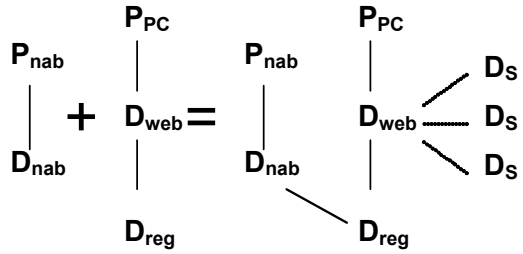

Figure 3. Registrating the Nabaztag. (Dashed lines denote couplings that result from a chain of reactions.)

Through form filling, the user can specify which services are of interest. The resulting molecule is shown in Figure 4-a. Once the user has disconnected from the web server, the Nabaztag, as an information appliance, is a rather well-balanced star-like molecule with a stem coupled to a single P atom (see Figure 4-b). At any time, the user can reconnect to the web server via the PC to modify the Nabaztag molecule

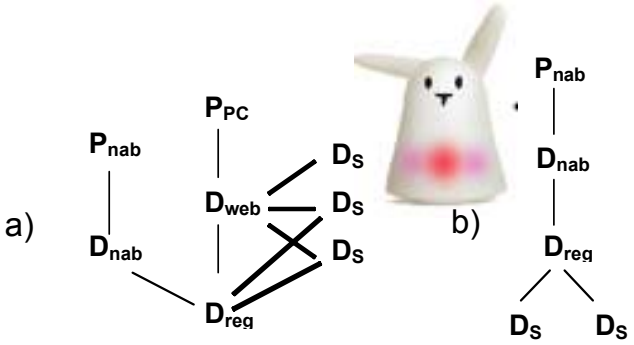

Figure 4. The Nabaztag as a working information appliance. a) The PC is in use. b) The autonomous Nabaztag.

This simple real-life example illustrates two problems: indirectness and semantics uncertainty:

- Users cannot customize their Nabaztag directly. They have to build a number of intermediate molecules that are of no interest to them. In addition, the complexity of these intermediate molecules is a good candidate for potential breakdowns, both technical (e.g., a network disconnection) and cognitive (e.g., the user makes a mistake).
- Users may be uncertain about the semantics of the final molecule. Using our exemplar, how can users be sure that the final boundary of the Nabaztag is the "nice" molecule shown

in Figure 4-b? It is possible that the D_s atoms are coupled with other services whose behavior may have an undesirable impact on the native D_{nab} (for example, in terms of privacy). By the way of the registration molecule, you can marry your Nabaztag to another one. If this function is transitive, your Nabaztag becomes polygamous. In other words, consequent bonds are created between your Nabaztag and other Nabaztags and this may be unknown to you. As another example of chain of reactions, the arrival of the Nabaztag in the home, which may be a huge molecule or a set of coexisting molecules, may entail a chain of reactions that may not be predictable or even observable and controllable.

Suppose for example, that the Nabaztag joins the smart home represented by the molecule shown in figure 5. This smart home is equipped with a presence detector, a surveillance system, and an IP-device discovery facility. It includes a number of smart objects 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.

Suppose that the user has subscribed the Nabaztag to the buddies messaging service. The Nabaztag is then able to play messages sent by buddies using its speaker-phone. Unfortunately, it has no replay facility. Thus, when there is nobody home, messages are lost. In order for the Nabaztag to forward incoming messages to the recording facilities of the home, additional bonds must be created between the appropriate D services. These can be created autonomously by the smart artefacts, or they can result from the interpretation of an end-user program, or from a smooth combination of both. Whatever the process, an initial coupling is necessary to entail a chain of reactions until reaching a stable appropriate situation.

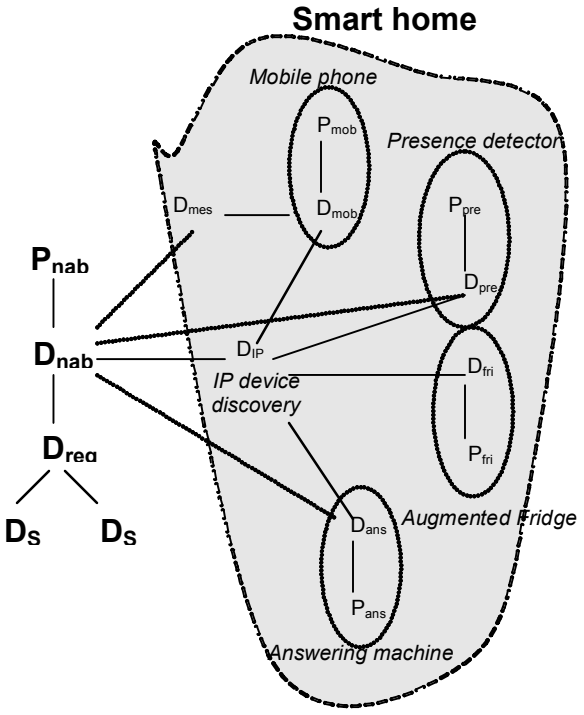

Figure 5. A smart kitchen as a unique molecule.

In our case, because the Nabaztag is an IP device, a bond can be automatically established between D_{IP}, the IP-device discovery facility and D_{nab}. D_{nab} is now able to discover D_{pre}, the presence detector, as well as D_{rec} to use the recording facility of the answering machine or even D_{mes} to forward messages to the mobile phone. These bonds can in turn provoke new chains of reactions. How far can we go? The complexity is combinatorial, even for very simple situations. For example, the three elements D, P, D' may yield into [D–P] + D', or [D'–P] + D, or [P–D–D'], or [P–D'–D] or into a complete graph where all the nodes D, P, D' are interconnected. Are all of them possible, or even desirable?

Drawing again on the analogy with molecular chemistry, the following notions may help reasoning about these issues: valence and affinity.

2.3 Valence and Affinity

In chemistry the notions of valence and affinity have been introduced to rationalize the formulae of different chemical compounds.

In our context, the *valence* of an element is an integer that measures the maximum number of bonds that this element can support. The *affinity* between two elements denotes their capacity of forming new artefacts provided that they both satisfy a set of constraints. In our case, constraints may apply to physical form factors, software discoverability and interoperability, cognitive compatibility, semantic relevance, and more generally human values including ethical values (why not?).

Let's come back to Hinckley's tablets. Valence : because of the form factor, each P can be linked to four P's at most (one screen on each side). Affinity: the D's can be interconnected only if they are interoperable. In case of the Nabaztag, the "marry" function is transitive for my Nabaztag only if I, the owner, believe in polygamy!

Valence and affinity are two ways to limit the combinatorial effect of networked smart devices as well as controlling chains of reactions in a semantically relevant fashion. We need now to address the implications for both software engineering and HCI using the particular case of service oriented computing and end-user development.

3. SERVICE ORIENTED COMPUTING AND END-USER DEVELOPMENT

3.1 Service-Oriented Computing

Service-oriented computing (SOC) is currently an active area of research in Software Engineering. It is motivated by the need to address heterogeneity, dynamic deployment and reconfiguration, and this at multiple scales. SOC is based on three kinds of D's – service providers, service consumers and service brokers, along with the specification of the services that are provided and needed. The bonds between the D's correspond to three kinds of interactions: service publication between providers and brokers to declare to the world the services they offer, service discovery between consumers and brokers to find the desired services, and service invocation between consumers and providers to effectively use the services.

Clearly, the SOC paradigm matches the chemistry-inspired composition that we propose here. On the other hand, the service

specifications, which formally express the notions of valence and affinity between atomic elements, convey system issues, not end-users needs. Typically, the contracts between software components are primarily syntactic. The semantic level is still overlooked. In particular, service descriptions do not include the description of the human task that they are able to support, nor do they express the quality of service they can achieve in terms of human values.

As a result, service composition is determined by system-oriented people, not by end-users. In addition, according to my knowledge, there is no way for a particular consumer D1 to directly control the services that its service provider D2 uses effectively. As demonstrated by the Nabaztag, end-users need to specify some form of range for chains of reactions. One possible approach is to explore a recursive model as proposed by Fractal for software components: a configuration of components can be encapsulated as a component []. By doing so, one can define clear boundaries along with the specification of a description that would include human-centered concerns.

3.2 End-User Development

The Nabaztag exemplar, although simplistic, shows the necessity for end-user development (EUD). As discussed at the Dagsthul seminar on EUD [2], we still do not know which notation(s) to offer to users, nor to support them with debugging facilities and reuse. Partial solutions have been developed for specific domains, problems, and target users. Clearly, we need to seek for generalization. “Lingua Franca” is an interesting attempt in this direction [3]. It is a scripting XML-based pivot language intended to support multiple end-user notations such as a visual language and the tangible Media Cubes.

Having defined the “right” notation for a particular class of end-users addresses the syntactic issue, but leaves opened the semantic dimension. This is where, I think, we need to align the semantics of end-user languages with that of service descriptions and compositions so that: 1) users can develop a good understanding of the current state of the networked artifacts, 2) users can control this state including delegate tasks to the networked artifacts, and if necessary, deploy and repair faulty artifacts. A recursive model of services would permit to define the appropriate level of encapsulation to hide complexity. Such a compound service would be very robust and trustworthy, and would not need human intervention (just like a car whose complexity is particularly well encapsulated because engines are now very reliable).

But this is not enough: programs correspond to predefined scenarios. Currently, electricity providers sale devices that let you program a number of situations: energy saving during the week-end and at night for such and such rooms, and a nice level of comfort during week days based on your life style [1]. In ubiquitous computing where human opportunism and unpredictability are paramount, these programs may need to be revised on the fly either by the end-user or the system. Thus, the service(s) that are in charge of interpreting end-user programs must be carefully designed since they potentially affect all of the elements of the molecule. Therefore, just like security, end-user interaction is a transversal issue. This aspect needs to be addressed explicitly in collaboration with the software engineering community.

4. CONCLUSION

In this article, we have defined a model for networked artifacts inspired from molecular chemistry. It demonstrates the intrinsic complexity of the domain illustrated with unsolved problems. These include human control of a chain of reactions to context changes and understanding the semantics of a particular configuration of networked artifacts. Based on this model, we have identified similarities with the SOC paradigm developed in Software Engineering as well as limitations in one of the central concepts of services-oriented approaches: the absence of human concerns in service descriptions. We also suggest three possible avenues for collaboration between Software Engineering and researchers in EUD: a recursive model of services as a way to keep complexity away from the end-user, semantic alignment between end-user programming languages and service descriptions, and human-service interaction as a transversal issue (not as an epi-phenomenon).

5. ACKNOWLEDGMENTS

This work has been supported by the network of excellence SIMILAR (IST, FP6) and the ITEA EMODE project.

6. REFERENCES

[1] Bourcier, J., Escoffier, C. Lalanda, 2007. Implementing Home-Control Applications on Service Platform. In Proceeding of the IEEE Consumer Communications and Networking Conference, CCNC 2007 (Las Vegas, NV, USA, 11-13 Jan. 2007).

[2] Burnett, M. H., Engels, D., Myers, B. A., Rothermel, G. 2007. User Software Engineering, Dagstuhl seminar Proceedings 07081. DOI = http://drops.dagstuhl.de/portals/index.php?semnr=07081

[3] Hague, R., Robinson, P. Blackwell, A. 2003. Towards Ubiquitous End-User Programming. In Proceedings of the 5th annual conference on Ubiquitous Computing (UbiComp), Seattle, October 2003.

[4] Hinckley, K. 2003. Synchronous gestures for multiple persons and computers. In Proceedings of the 16th Annual ACM Symposium on User interface Software and Technology (Vancouver, Canada, November 02 - 05, 2003). UIST '03. ACM Press, New York, NY, 149-158.