

# Palpable Computing and the Role of Agent Technology

Giovanni Rimassa, Dominic Greenwood, and Monique Calisti

Whitestein Technologies AG, Pestalozzistrasse 34, 8032 Zürich, Switzerland,  
{gri, dgr, mca}@whitestein.com

**Abstract.** This paper presents a computing approach, called *Palpable Computing*, complementing and extending Ambient Computing notions and techniques. The main contribution of this paper lies in defining and discussing what role Agent Technology can play in Palpable Computing, and which of its ideas and technical approaches are (or are not) well suited to be adopted as a support for the Palpable Computing vision.

## 1 Introduction

*Ambient computing* has been associated with various trends and definitions. Regardless of which perspective is taken, the common and central focus is on a new paradigm for user centric computing and interaction. In the near future, people will have access to distributed networks of intelligent devices populating their daily environments and providing them with information, communication and diverse services at any time and in any location. These networked systems are expected to adapt themselves to user requirements and even anticipate their needs. The ability to appropriately engineer such systems has generated and is continuously triggering challenging questions in several areas of computer science, engineering and networking.

*Palpable computing* aims to provide some concrete answers by extending and complementing the ambient computing vision, concepts and techniques. The key idea, as detailed in the following sections, is to define systems that users can intuitively notice, understand, use and control in the most appropriate way according to the specific context/situation. Software that adapts to changes in the environment, minimizing human intervention and service interruption, is the central foundation of such an approach. Our proposition is that *Agent Technology* offers powerful concepts and consolidated techniques to specify, design and build palpable software systems [1] - as discussed in the second part of this paper.

## 2 The PalCom Project

Defining and investigating Palpable Computing is the overall goal of the PalCom project [4]. PalCom is an integrated project in EU's 6th Framework Programme under the initiative "*The Disappearing Computer*" in the *Future and Emerging*

*Technologies (FET)*, part of the *Information Society Technologies*. The project started in January 2004 and will last four years.

The project consortium is composed by twelve partners, mostly academics, with a very diverse set of skills and interests. A significant percentage consists of Computer Science departments and IT companies, but other partners deal with Architecture, Interaction Design, and Ethnography. The chosen approach to effectively leverage this broad set of know-how and expertise was to act simultaneously in a technology-driven and a user-driven fashion.

The term “palpable” is meant to denote systems that can be noticed and mentally apprehended. Palpable systems should support people in understanding and controlling their operation, letting the users choose the level of information provision and automation they see most fit for a specific situation. While a precise definition of “palpable system” and “palpability” will be built throughout the project duration, a first step was to set up a dialectic relationship between Palpable Computing and Ambient Computing. This was achieved with the definition of six pairs of *palpable qualities*:

1. **Invisibility** complemented by **Visibility**
2. **Scalability** complemented by **Understandability**
3. **Construction** complemented by **De-construction**
4. **Heterogeneity** complemented by **Coherence**
5. **Change** complemented by **Stability**
6. **Sense-making** complemented by **User control**

The first element of each pair in the list above is generally a cornerstone of Ambient Computing, whereas the second element opposes it and represents the new focus added by Palpable Computing. In true dialectic fashion, the advancement is expected to be made by overcoming the conflict and finding an innovative, better, balance.

The PalCom project is expected to provide two major results. On the one hand, the technology-driven activity and the user-centered prototyping and scenario evaluation will yield an *Open Architecture* to drive design and implementation of software system that can exhibit palpability. On the other hand, the overall effort in characterizing what palpability is and which systems and use circumstances are palpable, will result in the production of a *Conceptual Framework* serving as guidance to conceive, understand, and assess palpable systems in the broadest perspective.

## 2.1 The PalCom Open Architecture

The Open Architecture is the technical nexus of the PalCom project. Its goal is to serve as the means of transcribing the palpable qualities discussed in Section 2 into a set of interrelated, computationally realised concepts drawn into a coherent and encompassing structure. This implies that it must capture the essence of how human actors interact with, and within, their everyday environments through distributed populations of palpable and non-palpable resources.

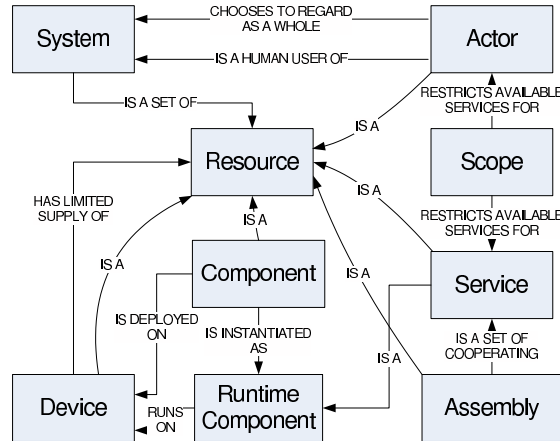
To some degree, the existing architectural specification builds upon established concepts in relevant fields of software engineering expertise. But it also specialises this know-how to weave a consistent computational fabric that, as the sum of its parts, delivers the means to pragmatically enable aspects of palpability in genuine and useful ways.

Realised as a set of documents, the scope of the architecture includes:

- **PalCom Runtime environment (PRE)**. An optional infrastructural element, the PRE allows PalCom Software Components to execute across multiple hardware devices and operating systems. It defines a virtual machine that has a common object format and common binary standard, and is designed to be language independent. In use, it enables strong portability, mobility and dynamic updating of software components.
- **PalCom Communication Model**. Connectivity is a fundamental aspect of the PalCom Architecture as it empowers devices and services to collaborate. PalCom devices are required to be network enabled, but not bound to any particular media or protocol. The basic model supports announcement and discovery of deployed services via publish/subscribe. Peer to peer messaging is also supported.
- **PalCom Component Model**. This model defines the common unit of software functionality, deployment, and composition in PalCom. As a unit of functionality, a PalCom Component represents both infrastructure and domain-specific software building-blocks. Components can be composed to form aggregated behaviours and are instantiated as PalCom Runtime Components.
- **PalCom Resource Model**. Resources are those elements of the Architecture that can be manipulated and reasoned about. This spans from available memory on a device, to a human user in terms of their role and associated actions. The Resource Model defines the relationships between these resources and how they may be manipulated to realise palpable applications.
- **PalCom Contingency Model**. Closely associated with the Resource Model, this addresses a core quality of palpability - resilience. It defines how contingent plans can be devised to ensure continued, seamless operation of active systems whilst providing visibility over the reasons why a problem has occurred.
- **PalCom Assembly Model**. A PalCom Assembly defines the scope of an application in terms of the resource interrelationships required for its delivery. At the highest level assemblies consist of a set of interacting devices.

The relationship between these models is described by an Architecture Overview document that consolidates the major concepts, qualities and usage guidelines to provide an abstract specification to guide system designers in the creation of concrete palpable applications.

The key concepts identified by the PalCom Architecture are illustrated in Figure 1. As can be seen, the notion of *resource* is envisioned as a central concept, encompassing both low-level device-based resources such as memory, and high-level resources such as components, services, devices and actors.



**Fig. 1.** Key concepts of the PalCom Open Architecture

*Actors* interact with a PalCom system through *assemblies*, which realize the notion of a task whose execution is adaptable toward actor-specific needs. To execute their tasks, assemblies orchestrate appropriate domain-specific and infrastructure services available within the PalCom System and configure them according to the tasks quality of service requirements. When orchestrated into a specific assembly, *services* must interact cooperatively with one another to fulfil their respective and collaborative tasks. This requires a *runtime infrastructure* that allows PalCom assemblies and services to execute and communicate with one another in, what is, an inherently distributed computing environment. This runtime infrastructure must also support the concurrent execution of multiple assemblies, which may compete with one another for available resources. The PalCom Open Architecture follows a strict component-based approach. This helps in the realization of services in a strictly modular fashion and will typically employ a shared repository to promote reusability.

## 2.2 The PalCom Conceptual Framework

As shown in Section 2.1, the PalCom Open Architecture, albeit covering many aspects, is basically a high-level specification for the design of software that can run on a variety of devices with strong networking capabilities. However, PalCom features both the ambitious goal to go beyond Ambient Computing and a diverse expertise set in the project consortium. In engineering-driven efforts the most general conceptual document is the system architecture: the Conceptual Framework is both wider in scope and more general in abstraction than the PalCom Open Architecture. It generalizes some concepts and ideas from it, but also adds an entirely new dimension.

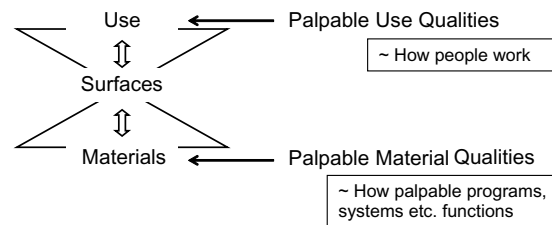
The added dimension relates to *use*. While the Open Architecture guides the designers of palpable systems and applications, the Conceptual Framework is

supposed to allow both users and designers to understand and organize knowledge about palpable systems. The *use* dimension stems from the multidisciplinary approach adopted in PalCom, one of the main staples of which is *participatory design* [7].

The PalCom Conceptual Framework is still a work in progress. So far, a first metaphor has proven to be useful in relating technology-centered and use-centered aspects of PalCom. Three items compose the metaphor:

1. *Materials*. This item represents how palpable systems function.
2. *Use*. This item represents how people work/interact with palpable systems.
3. *Surfaces*, representing the boundary between palpable systems and the people using them.

The elements of the metaphor and their relationships are depicted in Figure 2.



**Fig. 2.** The basic metaphor of PalCom Conceptual Framework

The *material* concept aims at suggesting that a computational system, due to the way it has been built, has some properties of its own (akin to the physical and chemical properties of a “real” material). Properties of the materials in this metaphor are some of the familiar non-functional software qualities: *flexibility, resilience, security, scalability*.

Such properties affect what can and cannot be done with the system, but they are by no means sufficient to predict how the system will be used once made available to real people. The *use* concepts, instead, consider exactly what happens in that case. Most of the qualities that concern the use are then relating to how people perform their tasks. Some examples of PalCom use properties are:

- **Indexicality**. This property is also named *situatedness*. It refers to the property that people’s action and sentences have, of receiving (part of) their meaning through the context in which they are performed.
- **Intersubjectivity**. This property relates to the fact that people usually assume that their own actions and language will be understood by others in a way that will facilitate interaction.
- **Reciprocity of Perspective**. This property signifies that people generally assume that, if they put themselves into someone else’s shoes, they would see the situation in the same way.

As the list above shows, key properties belonging to the *use* dimension are strongly dependent upon a context that is extremely volatile and hard to capture, being linked to social and behavioral traits of users and of their operating environment.

The Conceptual Framework bridges *materials* and *use* dimensions with the concept of *surfaces*. Physical materials lend themselves to use only through their surfaces; likewise, palpable systems will have to expose only a part of themselves to the outside. Surfaces must not be reduced merely to the idea of interfaces in a software engineering sense. What is exactly a surface still depends on the use dimension: two different users could form two equally effective models of the same system. In the PalCom metaphor, the two are using the same material through two different surfaces. Most of the concepts pertaining to the *surfaces* dimension show an external view on some technical entities in a computing system. As an example, the definition of *Connection*, in the context of the *surfaces* part of the PalCom metaphor, is not what a network engineer would give, but rather what a person that is proficient with using wireless devices, though potentially unaware of the technical details of their working, would.

### 3 Agent Technology and Palpability

Previous sections described the PalCom project goals. The main question is now whether, and how, can Agent Technology provide conceptual and practical contributions to this effort. Quite a few hints suggest a positive answer: ideas and requirements emerged in the initial phases of work are pretty much in line with essential features of the Agent Technology approach.

At the Open Architecture level, a loosely coupled and very dynamic component model is advocated. It is also stated that, in order to support the emergence of effective usage from human actors, palpable systems will have to follow a *task-oriented* model (as opposed to a more standard, application-oriented one), focussing on implementing user goals. The communication model fosters asynchronicity, relying on publish-subscribe discovery and communication protocols, while resorting to direct message passing between mutually aware components.

All the above prescriptions nicely match multi-agent systems (autonomous software components, perceiving from and acting on their environment, managing their own resources, communicating asynchronously with one another either through direct messaging or through a shared environment).

At the Conceptual Framework level, the *materials* dimension is a more abstract rendering of the Open Architecture and matches Agent Technology the same way. Slightly more surprising, instead, is that the *use* dimension also has many agent-friendly concepts. Indexicality immediately recalls agent situatedness, while intersubjectivity and reciprocity of perspective can readily point to behavioral implicit communication [2] or mutual agent modeling.

This is not due to chance at all, but follows from the multi-agent systems approach, where researchers have since long drawn inspiration from the social sciences. However, social sciences concepts and theories must be distinguished

from their versions adopted and implemented in software multi-agent systems. When taking *agency* in its broadest meaning, it is true that agent situatedness is exactly indexicality. However, when considering a software agent, the situatedness it can exhibit is just a stripped down version of what a human can do. While situatedness is effective in improving its behavior, it by no means make a software agent comparable to a human.

Thus, software agents should be left out of the *use* dimension. Actually, their exhibiting lesser versions of most use properties suggests a critical Agent Technology contribution in the *materials* and *surfaces* dimensions.

Much work is still needed to bring about the overall Palpable Computing vision and to precisely define the role of Agent Technology into it. The next subsections consider the two first research directions the authors are presently following.

### 3.1 User-aware Surfaces

Applying Agent Technology at the *surfaces* level naturally recalls *user agents*. Surfaces in PalCom Conceptual Framework are the contact point between human users and a software system, where user agents reside. The more or less common design is to select one specific software agent and attach it to a human user.

Often, system designers follow the presentation approach of giving that software agent an antropomorphic or zoomorphic semblance. The user is then invited to think of the software system as being somehow inhabited by a kind of synthetic creature, helping with the user tasks. Other times, a more standard GUI or VUI is used. Anyway, from the point of view of the system, the human user is wrapped and shielded by her own user agent. The resulting situation is thus:

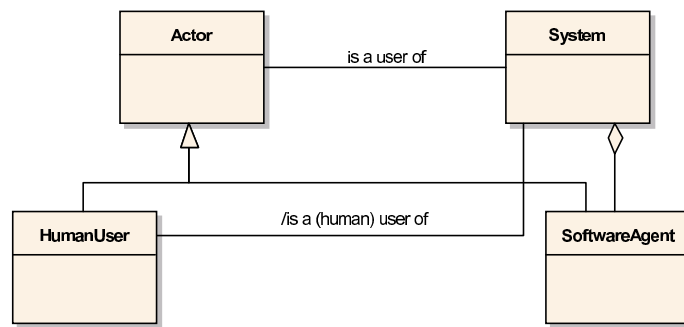
- Human users are invited to see interacting with the software system as *being like* interacting with another human.
- The software system is designed to consider interacting with human users as *being like* interacting with another autonomous software entity (agent).

A very interesting critique of this approach, from a sociological perspective, is made in [5], and partly in [6]. With no completeness ambition, some relevant points made there are:

- User agents foster the idea of a perfect, almost invisible infrastructure at the service of the user, abstracting away the human labor that is still involved in performing most high-tech tasks.
- While trying to keep the previous promise, the software models humans as autonomous, rational entities. It then engineers the human-machine interaction as between equals (i.e. both parties are autonomous and rational).
- However, it is generally the case that agency doesn't lie in the qualities of the human or of the machine. Instead, agency is a relational capacity that is enacted when a (specific) human and a (specific) machine come together and interact within a context that is made by the whole history of both.

Willing to accept this critique, an antropomorphic user presentation risks being counterproductive for an user agent. The level of agency in an interaction depends on the unique human-machine shared context. Inducing the human user to think of the software agent as another human would generate behaviors and expectations that are unsuited to actual system capabilities.

Even more important, though, is the converse: having the software agent modeling its user as *just* another software agent ends up unduly restricting the interaction. The system must be aware of when it is interacting with a human user. Some interaction properties will stay the same whether the user is a human or not, but in order to obtain a palpable system, other interaction traits will drastically change. A possible UML diagram describing this awareness is depicted in Figure 3.



**Fig. 3.** An UML model of Actor-System interaction structure

As Figure 3 shows, a software agent can both be a part of a system and also act as the external user (named Actor in the Conceptual Framework). The association “*is a user of*” connects an actor and the system it uses, and regardless whether the actor is a human or a software agent. However, the derived<sup>1</sup> association “*is a (human) user of*” captures the fact that just knowing that the actor is a human adds valuable information for the system.

The following list presents the features that a software agent needs to have in order to be able to effectively exploit that valuable information. An agent:

- Needs to know whether a perceived event was sent to it by a human or not (thus enabling human-to-machine behavioral implicit communication).
- Needs to model which effects of the actions it can perform can be perceived by the humans it is interacting with (thus enabling a machine-to-human behavioral implicit communication).

<sup>1</sup> In UML, a *derived association* is an association whose pairs can be computed from the association it derives from. In this case, each (HumanUser; System) pair in the *is a (human) user of* association is also an (Actor; System) pair in the *is user of* association.

- Must know or infer the effect that its modification of the environment can have on the humans it is aware of.

The authors believe that, if software agents acting at the system boundary have these features, Agent Technology can provide a significant push towards realizing the vision of Palpable Computing, improving user experience with the system while leaving the interaction space open for emergence of new uses. In the terms of the PalCom Conceptual Framework metaphor, such software agents can be the key to engineer *user-aware surfaces*.

### 3.2 Autonomic Materials

The PalCom Conceptual Framework chose to use the *materials* element of its metaphor to suggest that human users can perceive software systems as having some basic properties, which can be leveraged to flexibly create usages. In order to discuss a possible role of Agent Technology at the *materials* level, a first step is to initially consider physical materials and see how some consideration can be transferred to computational systems.

Some properties of physical materials, though natural, can be described by the same *feedback loop* idea that is commonly used in engineering (e.g., an elastic material is subject to a feedback force proportional to its displacement from the equilibrium position). It is also well known that the *sense-compute-act* loop of a software agent configures it as a kind of feedback controller with respect to its environment.

Recently, the IBM-fueled vision of *Autonomic Computing* [3] has added yet another perspective on feedback loops and software, claiming that software systems should be *self-regulating*, *self-configuring* and *self-protecting*. The overall emphasis here is on *automation* through *all-in-software feedback control of computational systems*.

Within this perspective we see that Palpable Computing arises from the *situated and contextual use* of some *computational materials* within the fabric of an application. A material is perceived as a resource that projects a tangible, yet often transient, presence into the feedback field persisting between the user surface and the computational infrastructure enacting a palpable application.

As such, a resource, whether it be e.g. memory, a physical device or human user<sup>2</sup>, automatically reflects its functional and embodied properties to both its human and computational consumers. By positioning an agent with deliberative capabilities within this field we can tune the use of resources to the changing context of an extant application while maintaining a coherent understanding of the effect on human engagement. This balance between the management of potentially complex and dynamically fluxing task structures, and the reflective condition of participating human users is one that requires the agent be embodied with decisiveness and thus the ability to take initiative in determining solutions within a changing resource landscape.

---

<sup>2</sup> A human user can be treated as a resource in terms of their role and the operational characteristics associated with that role.

Another aspect of the agent role is to effect contingent reasoning in resolving problem conditions arising from within the computational environment. If a device fails, for example, the agent will use its comprehension of that device's influence both on the environment and its human users to assess the impact and take contingent actions to enact a repair or other resolution. Contingency is essentially the capability of adapting to both predictable and unpredictable events, the former resolvable with contingent planning, but the latter requiring a degree of reasoning to determine courses of action. This positions the agent as a means to regulate stability through causal adaptation.

## 4 Conclusions

Today, there is a significant amount of work that is attempting to understand how we, as humans, will continue to explore and experience interaction with computing machinery throughout our everyday lives. Palpable Computing aims at extending and complementing the Ambient Computing vision by focusing on the definition of *palpable* systems as systems that can be noticed and mentally apprehended by users.

This paper has proposed two main research directions the authors are currently pursuing to understand and propose how Agent Technology concepts and mechanisms can contribute to ground and consolidate the definition of palpability and the implementation of palpable systems.

**Acknowledgements.** The authors would like to thank Thomas Lozza for his valuable contribution and all PalCom partners for their collaboration. Part of the research work described in this paper has been funded thanks to the Swiss State Secretariat for Education and Research (SER) 03.0495-2 Grant.

## References

1. Calisti, M., Greenwood, D., (2003) "On the Road to Ambient Intelligence", European Workshop on Multi-Agent Systems, Oxford, U.K.
2. Castelfranchi, C. "SILENT AGENTS: From Observation to Tacit Communication". In Proc. of First Int'l Workshop on Modeling Other Agents from Observations (MOO 2004), pages 25-32.
3. IBM Research. "Autonomic Computing Manifesto". Available at <http://www.research.ibm.com/autonomic/manifesto/>
4. The PalCom project Home Page. <http://www.ist-palcom.org/>
5. Suchman, L. "Figuring Service in Discourses of ICT: The case of software agents". In Wynn, E. et al (Eds.) Global and Organizational Discourses about Information Technology . Dordrecht, The Netherlands: Kluwer, pp. 15-32.
6. Suchman, L. "Figuring Personhood in the Sciences of the Artificial". Available at <http://www.lancs.ac.uk/fss/sociology/papers/suchman-figuring-personhood.pdf>
7. Winograd, T. "Bringing Design to Software". Addison-Wesley, 1996