Computer Persona - An Approach For Interacting with Large Number of Computer Like Devices

Ladislau Bölöni and Majid Ali Khan

University of Central Florida, Orlando FL 32816, USA,
lboloni@cpe.ucf.edu, ma680109@pegasus.cc.ucf.edu

Abstract. In this paper we propose a software organization, which allows users to treat a set of computer-like devices as single *computer persona*. The computer persona covers a static set of devices (called the *permanent range*) but it can also dynamically extend or leave rented or disposable devices (the *temporary range*). The mental model of the users will be of a single entity whose capabilities are extended with the devices currently in its range and which she is accessing through devices of various capabilities.
Implementing a computer persona requires the techniques of speech act based user interfaces, mobile and mutable agents and data staging / data fusion.

1 Introduction

Our approaches for developing software were designed with the scarcity of hardware in mind. From the mainframes where multiple users were sharing a single computer, we have evolved to the personal computer mantra of one user to one computer. Currently, however, we are living in a world where there are tens of computers or computer-like devices for every user, and this number can easily reach hundreds in a couple of years. One of the main difficulties in utilizing the possibilities of these devices is the natural human tendency to give attention to a limited number of entities. It was proposed that the number of entities a human can maintain in its short-term memory is about 6-7. Even somebody who interacts with many persons per day, there are a limited number of *persons* with which he can have non-routine interactions. On the other hand, we can interact with a large number of *objects*, which can be seen as simply members of their class and used according to their class properties. Our mental image of computers is closer to a person than to an object. In fact, users frequently anthropomorphize a computer or application by modeling it as a *computer persona.*

Unfortunately, the proliferation of computer personas presents the same problems as the interaction with multiple humans. In this paper we propose an approach which allows us to utilize the capabilities of a large number of devices by allowing them to be modeled by the user as a *single* computer persona.

The paper is organized as follows. In section 2 two scenarios illustrate the possibilities of the approach. The architecture of a computer persona is discussed in section 3. Section 4 puts the persona approach in the context of various other approaches to deal with large number of computers. A prototype for a subset of the computer persona is presented in section 5. We conclude in section 6.

2 Scenarios

In this section we present two scenarios to illustrate how the computer persona approach adds value to a set of computer-like devices.

Scenario: Business traveler. The business traveler is an extreme example of a completely mobile workplace. While we feel that this example is over-used, the lives of most of us involve some level of mobility. We will use the business traveler example as a representative of a larger class of similar problems.

A business traveler is carrying a limited number of portable devices: typically a laptop, a PDA and a cellular phone. However, during her travels, she interacts with a large number of computer-like devices (sometimes called the *computational ecosystem*), which are used on a rental basis. Instead of learning to interact with devices on a case-by-case basis, and configure them according to her preferences, the traveler can *extend* her computer persona to these devices.

While flying to a remote location the traveler is using the computer integrated with her seat to review her presentation. Her computer persona is extended to the seat computer during the flight, but it is removed as soon as she leaves the plane. At the remote airport she rents a car, and extends her persona to its board computer. This includes personalization data but also voice interface based versions of her applications. She can listen to her e-mails or review her presentation through the car speaker. She can also use the car audio system for voice communication, her voice mail commands, address book and communication provider account will be available. Arriving at the hotel, the hotel room also contains a number of computer-like devices on which the computer persona can be extended: the air-conditioning, the TV set, the phone.

We will stop here with this scenario. Our conclusion is, that as the number of computer-like devices grows in our environment, a large number of them will be used in rental or disposable mode. The persona model of operating them solves the difficult problem of learning and configuring a large number of devices.

Scenario: Fire emergency. Our next scenario involves an emergency situation. Let's assume that a fire occurred in a large building and a team of firefighters is working to save lives and contain the damage. The fire fighter team naturally has its own computerized support and communication devices. However, a number of additional devices already present in the building can be used to help in fighting the fire and saving lives. The thermostats of the air conditioning system can be used to report the temperature in each room. A child monitor system can detect the presence of a distressed person. The smoke detectors can be used to create a map of affected areas. The GPS device in the pocket of a trapped person can be used to detect her location in the building.

The information from a number of devices can be correlated such that it gives an estimation of the structural stability of the building.

None of these uses require any new capabilities from the devices. The reason why this scenario seems futuristic today is the difficulty to reprogram the devices such that they can be used in the ways presented above. Also, the fire fighters do not have time to learn about the existence of these devices or their user interface. In our parlance, the computer persona of the firefighter team is extended over a number of devices, which are treated as generic programmable devices with different capabilities. The fire fighters are still accessing the same persona, which they have used in their training and other missions, with the difference that the computer persona is now extended over a new set of devices.

3 The architecture of the computer persona

A computer persona is a distributed and dynamic set software entities, which interacts with other computer devices and persons as a single entity. Towards users, it presents an interface with a *continuity of user interaction*, even if the devices through which the interaction happens change.

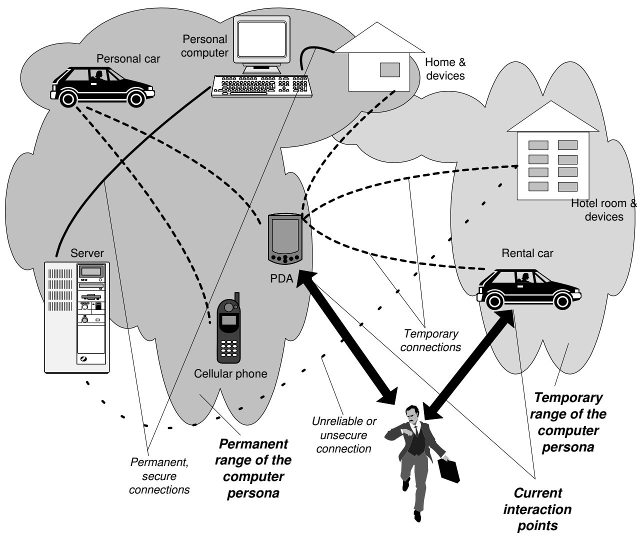

Personal car
Personal computer
Home & devices
Hotel room & devices
Server
PDA
Rental car
Cellular phone
Temporary connections
Temporary range of the computer persona
Unreliable or unsecure connection
Permanent range of the computer persona
Permanent, secure connections
Current interaction points

Fig. 1. The networking architecture of a computer persona

Figure 1 shows the operation of a computer persona. The *permanent range* is the set of computers and computer-like devices, which the user physically owns. We include here the services rented on a long-term basis. Examples of these devices are personal computers, PDAs, cell phones, embedded computers in personal cars, or subscriptions to online services such as Yahoo, or the suite of .NET enabled services. The set of applications run by these devices is relatively constant, thus the computer persona is implemented mainly through techniques of user modeling and data staging.

In contrast, the *temporary range* of the computer persona contains rented or disposable devices. There is a requirement for code mobility, mutation and customization as the persona extends to these devices. In the case of rented devices, appropriate measures need to be taken that the persona uninstalls all personal applications and destroys all personal data from the rented device. This can not be performed under the control of the remote server, because the existence of a connection to the rest of the persona can not be guaranteed at the termination of the rental period. It is perfectly possible that the computer persona becomes communicationally fragmented during its lifecycle. This underlies the necessity of using agent technologies, where individual devices can pursue their agenda in an autonomous manner, while at the same time having an awareness of the environment.

Creating a computer persona requires the coordination of various techniques of the fields of software engineering, mobile and mutable agents, distributed systems, knowledge representation, networking and security. In this paper we will discuss two of the most technically challenging steps in the lifecycle of the computer persona, (a) migrating the point of contact between the devices of the permanent range and (b) extending the computer persona into the temporary range.

3.1 Migrating the point of contact

We choose to describe the interaction between the computer persona and the user in terms of *speech acts* and *conversations*. Speech act theory was originally developed to describe human communication. An important application of speech act theory is the design of agent communication languages such as FIPA ACL or KQML. The individual speech acts / messages can be assembled into conversations [1], also called interaction protocols (in the FIPA parlance) or message patterns. The speech act model can be extended in a straightforward manner to the interaction between a human user and an agent.

In a computer persona, what the user sees is that speech-acts are *portable* across the changing interaction points. The computer persona needs to maintain the set of ongoing conversations happening through the current set of interaction points. If an interaction point is abandoned, the computer persona needs to map the ongoing conversations to one of the currently existing ones. Certainly, there might be cases when a conversation needs to be interrupted, because either there is no interaction point at all, or the conversation can not be continued at the remaining ones. For example, an instant messaging session can be continued with a voice interface, but a picture browsing session needs to be interrupted until the interaction will happen through a device with a visual display.

The change of interaction points is usually under the control of the user. However, the persona can initiate a change in the interaction point, for instance, to establish an interaction point (e.g. by ringing the cellphone or posting an alarm on the PDA) or to move the interaction to a more desirable channel (for example, it might suggest the transfer to a cheap residential internet connection from an expensive wireless one as soon as this becomes feasible).

Thus, the main challenge when a user is changing its point of contact inside the permanent range of the computer persona is how to maintain and continue existing conversations.

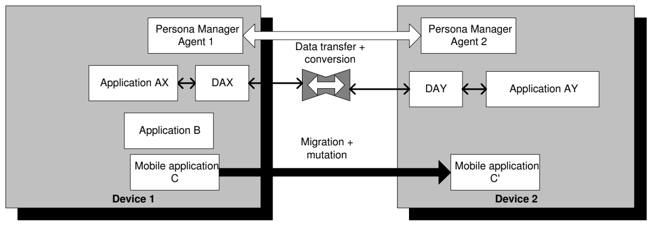


Fig. 2. Migrating the point of contact inside the permanent range

Figure 2 shows the sequence of actions which takes place when the user moves its point of contact from Device 1 to Device 2. We assume that these devices are heterogeneous and running different computing platforms. For instance, Device 1 might be a desktop computer running Linux, while device 2 can be a Personal Digital Assistant (PDA) running Windows CE.

The operation of the persona is coordinated by the *persona manager agents*, PMAs. The PMAs maintain a *model of the user*, including personal data, preferences, and the current interaction point. The PMAs can detect, and in some cases, predict, the change of the current interaction point.

On Device 1, the user was interacting with three applications: AX, B and C. Application AX is a native application of the device, which operates on the data set DAX. Device 2 has an alternative native implementation AY, partially equivalent to application AX on Device 1. When the point of contact is moved from device 1 to device 2, the application AX is notified by the local PMA, it saves its data to the dataset DAX and terminates. The dataset is then transported to device 2 under the control of the PMA and, if necessary, converted to the format of application AY, DAY.

We assume AX and AY to be legacy applications which are not aware that they are participating in a computer persona. Thus it falls on the PMA to assure the seamless operation of the devices. There are several difficulties:

(a) **Partial compatibility of the data formats and lossy conversion.** This leads to a data management / data fusion problem. Customizing data management techniques for the particular requirements of computer persona and developing new techniques is one of our future research objectives.

(b) **Managing the conversations.** The data transfer is just one of the elements of the puzzle. To achieve the single persona illusion, we need to maintain the continuity of the conversation. This is difficult, because legacy applications either do not model their interactions as a conversation, or do not expose it as such. Many applications however maintain their state information as a *session.* The session information can be transported and converted between different applications[1].

Application B is a simpler case. There is no equivalent application on Device 2 (for instance, B is a audio player and Device 2 has no audio capabilities). The MAS will signal the application that the contact has terminated. The action of the application B takes depends on the local policy, which can range from no action, to the termination of application.

Let us now move to the application C which is a mobile and mutable agent. In contrast to applications AX and AY, the reaction of the agent is to *migrate* to device 2 and to *adapt* to the requirements of the new platform. Let us assume that C is a Java agent based on the Jade/LEAP agent platform with the Bond extension libraries. On the desktop environment of device 1, with the Java Standard Edition platform, the agent has a full featured Swing user interface. On the Personal Java environment of device 2, the agent needs to run a simpler, AWT based user interface. Depending the other features or possibilities of the platform, other mutations might be needed. An example of such an agent is presented in Section 5.

3.2 Extending a computer persona to the temporary domain

When a computer persona is extended to a device in the temporary range, a new set of problems need to be faced. First, the persona needs to find an *entry point* to the new device. This is usually a account with program execution and file transfer rights, such as an `ssh` account. The first component of the computer persona transferred to the new location is the *Discovery Agent* (DA), which identifies the capabilities of the device, the native applications and potential security threat. The DA is in itself a mutable application. As in the moment when originally migrated to the target device its knowledge of the device is limited, the DA starts as a very small agent and is built out through successive agent surgery operations, as the resources are discovered. After the discovery is complete, a customized persona manager agent is transferred to the device. The first actions of the PMA are the installation of custom applications for which the user has

[1] A *session* is a weaker concept than a *conversation.* Let us assume that both AX and AY are word processors. Maintaining a session means that the user will find on device 2 the application opened with the same documents as he left AX on the device 1. Maintaining the conversation would mean that if he was in the middle of a search-replace operation, he would find the search-replace dialog open with the same values and the same location of the search.

licenses and the customization of the existing applications. At this moment, the new device becomes part of the computer persona and will participate in the lifecycle of the persona exactly as the devices in the permanent range.

A complementary set of tasks needs to be performed by the PMA at the moment when the computer persona leaves the temporary device. All the changed data needs to be transferred to devices in the remaining range of the persona (and preferably, to the persistent range). The custom applications have to be uninstalled, and the personal data deleted. The computer persona leaves the device by shutting down and deleting the persona manager agent.

4 Related work

4.1 Alternative approaches

The computer persona is not the only approach by which the problems posed by the proliferation of computer-like devices can be dealt with.

Generic devices. This approach proposes to make devices generic, without the possibility of personalization. A good example of this is a public telephone, which is used on a rental basis, but can not be personalized. The users see these devices as objects, without the need of individualized attention. This approach is basically trading the benefits of personalization for the simplicity of the user interface. It assumes the existence of a widely accepted, simple interface. While appropriate for some devices, this approach underutilizes the capabilities of the devices.

Invisible computer. The *invisible computer* [2] or *ubiquitous computing* [3] approach proposes devices which can seamlessly blend into an environment (*disappearing user interfaces*). We include here a number of related approaches, like pervasive computing, amorphous computing [4] or context aware computing. which in many cases propose significant departures from the current way computers are used. A large number of pointers towards various approaches together with proposed applications in agent systems are provided in [5]. Most of these approaches are adding value while effectively hiding from the user the fact that they are computer devices. This solves the information overload problem, because we don't need to form mental images in addition to the ones we are already accustomed with.

Server based approaches and thin clients. In these approaches the thin clients are acting as proxy for a server. All the relevant computation happens at the server side, thus the actual load on the client is minimal. The result is similar to the persona approach in the fact that multiple devices are represented by a single mental image. One example would be a world in which all applications are rented as services from the internet.

All these approaches have their place in the computing landscape. The approach we propose, in general, allows for a better utilization of the resources, because it treats all devices as full-featured computers. There is no requirement for continuous reliable connectivity (which is required by the server based approach). A persona organization would also perform a more graceful degradation of the perceived performance than a purely server based approach. This advantage, however, is paid with the higher complexity of migration of code and data (a general graph, instead of the star-shaped data migration pattern of the server based approach). Compared with the "invisible computer" approach, the persona approach is a less radical departure from the traditional user interface model.

4.2 Mobile and mutable agent systems

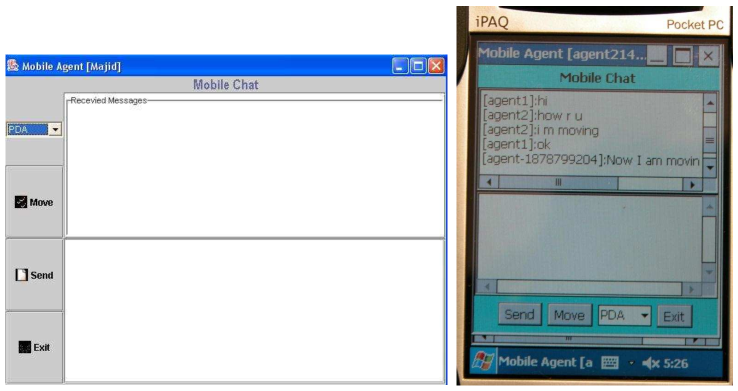
Mobile agents are autonomous programs that move through a network, and maintain their identity through this move. There are two major approaches for agent mobility. The strong mobility approach assumes that agents can move at any point during their execution and they are usually relying on specially designed programming languages (eg. Telescript) or on special versions of the Java JVM. One example of the last one is the NOMADS agents system [6] which is relying on the Aroma VM. Agents implementing weak mobility can migrate only at special checkpoints, and usually rely on freezing and thawing operations to prepare their internal state for migration (Aglets, AgentTcl, Jade, Bond).

As we have shown in the Section 3, mobile agents are a natural solution for maintaining conversations while changing interaction points.

Mutability is the ability to change an application during its runtime in order to either adapt to changing environments or to respond to changing expectations.

Recently several agent systems emerged with support for mutability. Varela and Agha [7] proposed the SALSA language based on the actor programming paradigm. The SALSA language is compiled to Java and targets dynamically reconfigurable Internet and mobile computing applications. The SmartApps approach [8] proposes an approach of "measure, compare, and adapt if beneficial" for scientific applications, with the restructuring occurring at various levels from the selection of the algorithmic approach to compiler parameter tuning. The Bond agent system was one of the first Java based agent systems with support for strong mutability. In [9] we have proposed a mutation technique called *agent surgery*. In [10] we have shown how certain classes of surgical operations on agents can be shown to maintain the original agenda of the agent. The basic operations of agent surgery like splitting and merging agents, and reconciliation of the knowledge bases are exactly the operations which (together with code mobility) are likely to appear in the operation of a computer persona.

In heterogeneous systems, mutation and mobility are strongly intertwined concepts. The difference in the resources available on desktop computer and a cell-phone (for example) is so big that agents cannot be migrated from one to the other without being significantly reconfigured. The solution is to migrate only part of the agent to the new location (using split and merge operations), and replace components of the agents with components that satisfy the constraints of the new location. The operation of extending a computer persona to new computer like devices involves both code mobility and code mutation.



```
[agent1]:hi
[agent2]:how r u
[agent2]:i m moving
[agent1]:ok
[agent-1878799204]:Now I am movin
```

Fig. 3. Screenshot of the transportable chat application on the desktop computer (left) and after migration to the PDA (right)

5 Persona prototype: chat application

To study the practical problems of implementing a computer persona, we started by implementing an interactive chat / instant messaging application with support for mutability and mutation, which can maintain the continuity of the conversation while the interaction point changes. This application corresponds to application C from Figure 2. This application can be used as a template for applications of interactive control, such as a *computation steering console* for the computational grid.

The implementation is based on the JADE/LEAP Agent Development Environment [11] which was complemented with a development version of the Bond 3 agent framework [12]. The application was tested on three platforms: two desktop environments (Microsoft Windows XP and Linux/KDE), where we used the Java 2 Standard Environment version 1.4.1, and a HP iPAQ PDA running Pocket PC 2000, where the Personal Java 1.2 by Insignia was used. The prototype application can be downloaded from the Bond webpage [12].

The main difficulties of the implementation involved maintaining the conversation state and bridging of the platform heterogeneities. The user interface differences between the Win32 API and the KDE/Qt architecture were bridged by the platform independent Java Swing toolkit. On the Ipaq however, the Personal Java environment does not support Swing user interfaces. Furthermore, the serialization data format was found to be incompatible between the two Java implementations. As discussed in section 4.2, this problem can be solved by using mutable agents. When migrating from the desktop platforms to the PDA, the implementation of the Swing user interface is removed through a mutation operation and a different user interface plane added which implements an AWT based user interface. The identity of the agent and its full conversational state is maintained though this operation, although the PDA interface has a more limited functionality. When migrating from the PDA platform to the desktop environment, an inverse process takes place. Figure 3 shows screenshots of the same agent on a Windows machine (left), and after the mutation and migration to the PDA (right).

6 Conclusions and future work

In this paper we have presented the computer persona, an approach for handling the interaction between the human user and a large number of computer-like devices. The persona approach opens interesting new possibilities for organization of the human-computer interaction, but it also poses a number of theoretical and practical research problems. The maintenance of data across the persona entity leads to problems of data staging and data fusion. The mutability aspect poses problems of application integrity. The security aspects of the computer persona should be considered for any real-world deployment.

References

1. Smith, I., Cohen, P., Bradshaw, J., Greaves, M., Holmback, H.: Designing conversation policies using joint intention theory. In: Proceedings of International Joint Conference on Multi-Agent Systems (ICMAS-98). (1998)
2. Norman, D.A.: The invisible computer. (MIT Press)
3. Weiser, M.: The computer for the 21st century. Scientific American **265** (1991) 84–104
4. Abelson, H., Allen, D., Coore, D., Hanson, C., Homsy, G., Knight, T.F., Nagpal, R., Rauch, E., Sussman, G.J., Weiss, R.: Amorphous computing. Communications of the ACM **43** (2000) 74–82
5. Servat, D., Drogoul, A.: Combining amorphous computing and reactive agent-based systems: a paradigm for pervasive intelligence? In: Proceedings of the first international joint conference on Autonomous agents and multiagent systems, ACM Press (2002) 441–448
6. Suri, N., Bradshaw, J.M., Breedy, M.R., Groth, P.T., Hill, G.A., Jeffers, R., Mitrovich, T.S.: An overview of the nomads mobile agent system. In: Proceedings of ECOOP'2000, Nice, France. (2000)
7. Varela, C., Agha, G.: Programming dynamically reconfigurable open systems with SALSA. ACM SIGPLAN Notices **36** (2001) 20–34
8. Rauchwerger, L., Amato, N.M., Torrellas, J.: SmartApps: An application centric approach to high performance computing. In: Proc. of the 13th Annual Workshop on Languages and Compilers for Parallel Computing (LCPC), August 2000, Yorktown Heights, NY. (2001) 82–??
9. Bölöni, L., Marinescu, D.C.: Agent surgery: The case for mutable agents. In: Proceedings of the Third Workshop on Bio-Inspired Solutions to Parallel Processing Problems (BioSP3), Cancun, Mexico. (2000)
10. Bölöni, L., Marinescu, D.C.: A component agent model - from theory to implementation. In: Proceedings of the AT2AI Workshop, Vienna, Austria, April 2000, to appear. (2000)
11. Jade webpage. (URL `http://sharon.cselt.it/projects/jade/`)
12. Bond webpage. (URL `http://bond.cs.ucf.edu`)