Computer Persona: a user interaction architecture for mobile environments

Majid Ali Khan, Damla Turgut and Ladislau Bölöni
Department of Electrical and Computer Engineering
University of Central Florida
Orlando, Florida, 32816
Email: khan@bond.cs.ucf.edu, turgut,lboloni@cpe.ucf.edu

Abstract— **In this paper, we propose a user interaction architecture, which allows users to treat a set of computer-like devices as a single computer persona. The computer persona covers a static set of devices (called the permanent range) but it can also dynamically extend to or leave rented or disposable devices (the temporary range).**

We identify three major challenges in the implementation of a computer persona: (a) the control and coordination problem, (b) the state handoff problem, and (c) the data staging problem. We discuss these challenges in detail and propose several solutions for each of them. We also present a heuristic based algorithm for data staging mechanism.

I. Introduction

While the goal of the ubiquitous computing envisaged in papers such as [1], [2] is not yet reached, today's professional interacts with several dozen of computer-like devices on a daily basis, using 3-10 such devices simultaneously. We call this later class of devices *interaction points*, each of them having a set of *open documents* with the associated *interaction state*. We are using the term "document" in a wide sense, meaning that a document might be a Word document, a radio program which the user is currently listening to or an open messaging session. The documents are handled by *applications* registered to handle the specific document types.

In a mobile computing environment, the set of interaction points dynamically change in time. For example, the user might use the PC, the PDA and the office phone as the interaction points in his office. In his car the interaction points are the navigational computer, the entertainment system and the hands-free cellphone adapter. Different interaction points appear on a rental car, in a hotel room or on the airplane.

We start with the assumption that the user is primarily interested in the documents, not the applications. The goal of the computer persona is to maintain (as far as possible) the set of open documents given the new interaction points. For example the user is listening to a radio program through its office computer. It is likely that he will want to continue to listen to the same radio program in his car as well. If he was editing a document in OpenOffice on his workstation, by connecting to his PDA he will find the same document opened in Pocket Word.

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

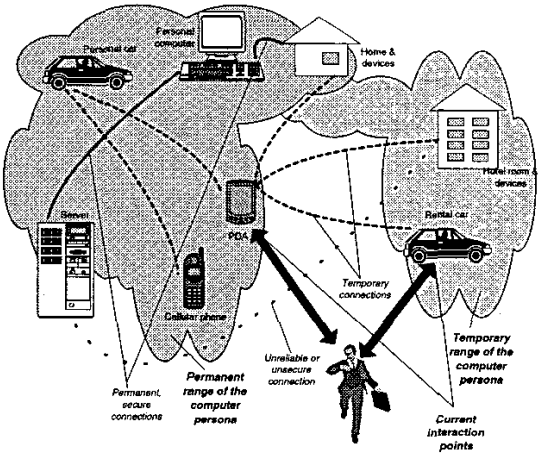

Fig. 1. The networking architecture of a computer persona

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 the 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 so that the persona uninstalls all personal applications and destroys all personal data from the rented device. This cannot be performed under the control of the remote server, since the existence of a connection to the rest of the persona cannot 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

0-7803-8255-2/04/$20.00 ©2004 IEEE.

awareness of the environment.

In conclusion, the mental image of the user will be that he is interacting with a single entity through different access points instead of a set of independent devices requiring individual attention. We identify three major challenges in the implementation of a computer persona, which are also major research directions for our group. These are (a) the control and coordination problem, (b) the state handoff problem and (c) the data staging problem. In the remainder of this paper, we discuss each of them in detail in Section II, Section III and Section IV respectively. The simulation and its results are shown in Section V. We then present related work in Section VI and conclude with Section VII.

II. Control and coordination

The **control and coordination problem** refers to the organization of the persona software, its decision making mechanism and methods of extending to or retracting from temporary domains. We developed an implementation based on autonomous, mobile agents, relying on the Bond extension of the Jade/Leap agent platform [3], [4]. To avoid reinventing the wheel, the agents rely on the existing operating system services and the applications' existing functionality. On each device, the persona agents need to handle the users `connect` and `disconnect` gestures, maintain the list of the open documents on the current device, and control the document upload and download operations. Some devices might serve as temporary storage for documents.

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 *entry point* is usually an 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. When it is 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. The operation of the persona is coordinated by the *persona manager agents*, PMAs. After the discovery is complete, a customized PMA is transferred to the device. The first actions of the PMA are the installation of custom applications for which the user has 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 has to be deleted. The computer persona leaves the device by shutting down and deleting the persona manager agent.

III. State handoff

The **state handoff problem** is the problem of transferring an individual document's state from one interaction point to the other. Different types of documents need special handling, but we can identify three large groups of documents, which need to be discussed. *Local documents* such as text or image files store their state in a local file. *Streaming documents* such as a radio station are identified by the source of the stream and the current location of the user in the stream. *Conversational documents* such as an instant messaging session or a phone conversation depend on a remote partner, and might have real-time constraints.

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 and messages can be assembled into conversations [5] (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.

The computer persona needs to maintain the set of ongoing conversations taking place 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.

The change of interaction points is usually under the control of the user. However, the persona can initiate a change in the interaction point to establish an interaction point (e.g. by ringing the cellphone or posting an alarm on the PDA) or can 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 it 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.

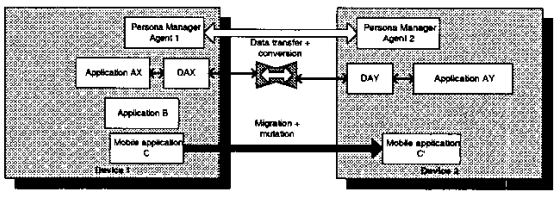

Fig. 2. Migrating the point of contact inside the permanent range

Figure 2 shows the sequence of actions that take 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.

The persona manager agents (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 that 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, since 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 PMA will signal the application that the contact has terminated. The action that application B takes depends on the local policy, which can range from no action to the termination of the 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, Abstract Window Toolkit (AWT) based user interface. Depending upon the other features or possibilities of the platform, other mutations might be needed.

[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.

IV. DATA STAGING

The **data staging problem** is concerned with planning the data transfers such that the user has an up-to-date version of every document at the device when he is accessing them. In *reactive data staging*, the new and the old locations of the data are known, and the goal is to find an optimal transfer schedule. For *predictive data staging*, data is transfered to final or intermediary locations based on a prediction of the future interaction points. This prediction might be based on techniques of *user modelling* or context information obtained from sensors [6]. The history of the persona (in terms of change patterns of interaction points and applications) can be used to create and update the user model. Security, privacy, data safety and handling of intermittent connections should also be considered while planning the data transfers. A mathematical model for a data staging program together with a heuristic algorithm for a simplified version is presented in [7].

The nature of the data staging problem for a personal network is multifaceted. First, it can be seen as an optimization problem, where the goal of optimization can be total time, used bandwidth or other criteria. From another point of view, data staging can be seen as a constraint satisfaction problem, the goal being to maximize the number of documents which are accessible in the new domain.

For the size of problem implied by a personal network, the constraint satisfaction aspect is more important. The number of alternative solutions is usually very small and often only partial solutions can be found. The constraint satisfaction problem is NP-complete, and can be reduced to the circuit satisfiability problem [8].

We developed a hybrid approach to solve the data staging problem in personal networks. The general outline of the algorithm uses a set of heuristics based on the common sense steps humans use when executing the data staging steps manually. The individual steps however, explore the solution space using a rule-based expert system. As an implementation platform we chose the Jess expert system shell [9].

We identified three possible scenarios of user movement for data staging. These include

- Known current and destination contexts with no intermediary context
- Known current, destination and intermediary contexts
- Known current and intermediary contexts with probabilistical set of destination contexts

In this paper we only deal with first two scenarios. The third scenario is left for future work. Before continuing our discussion on data staging any further we need to introduce several notations as follows:

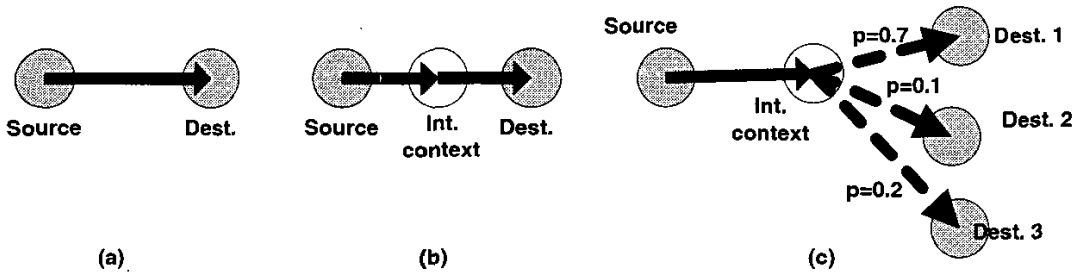

Fig. 3. Three scenarios for user movement in the personal network: (a) directly from source context to destination context, (b) from source context to the destination context through an intermediary context and (c) from source context to an intermediary context to probabilistical set of destination contexts

E	A user environment consisting of multiple contexts
$C[i]$:	A context i in user environment E
$D[d,i]$:	A device d in context i
$D[i]$:	All the devices in context i
$PA[d]$:	All possible applications on device d
$DT[a]$:	All document types supported by application a
c:	A user document
$OD[c,a,d]$:	An open document c on application a on device d
$L[d,i][e,j]$:	A link (i.e network interconnection) between device d in context i and device e in context j
$Capa[d]$:	Available storage on device d
$PF[d,t]$:	Preference level of device d for document type t
$CD[i,j]$:	Common devices in context i and j
O_c:	atomic operation to close a document
O_o:	atomic operation to open a document
O_t:	atomic operation to transfer a document

$RLD[c,a,d,i,b,e,j]$:
Relocation of a document c from application a on device d in context i, to application b on device e in context j.

For the first scenario, the user moves from a current context, $C[i]$, to a destination context $C[j]$. The user has a set of open documents in current context, $OD[c,a,d]$ where $d \in D[i]$. Each of these open documents has a document type, t and takes storage space s. There are various possible interconnections between devices in $D[i]$ and $D[j]$. Some of the interconnections are within context $C[i]$ and $C[j]$, i.e. $L[d,i][f,i], d, f \in D[i]$ and $L[e,j][h,j], e, h \in D[j]$. While other interconnections reach across contexts from $C[i]$ to $C[j]$, i.e. $L[d,i][e,j]$ where $d \in D[i]$ and $e \in D[j]$. We have to find an optimal set of combination for relocating each open document c on some device $d \in D[i]$ to some device $e \in D[j]$. The algorithmic steps used for developing the heuristics are given in Algorithm 1. First, we get all possible devices in both contexts. Then a set of devices common in both contexts is chosen. Since the common devices will be available in both contexts, we do not need to relocate the applications running on these devices. We then get all possible virtual interconnections between devices that occur due to the presence of common devices and transitive property of interconnections. In the next step, we get all possible combinations of application and devices in destination context that can possibly host open documents from current context using an interconnection. We then minimize the set of combinations based on preference level of a device. We then choose one combination per document (preferably with direct interconnection) and display the results.

For the second scenario, the user moves from a known current context $C[i]$ to a known destination context $C[k]$ via an intermediary context $C[j]$. The intermediary context bears different features since it is used mainly to store documents rather than open it. So the decision to select a device is solely based on available storage capacity and interconnections, unlike previous case where the document type also played a role. The algorithmic steps for the second scenario are given in Algorithm 2.

V. Simulation Experiments and Results

We simulated several test cases for the two and three context data staging using various number of devices, applications and interconnections. Following are some of our test cases:

A. Two Context Simulation

The configuration is that the user is moving from an 'office' context to 'home' context. The user had access to *Desktop*, *PDA* and *CellPhone* at his office and *Laptop*, *PDA* and *CellPhone* at his home.

Case1: There is no interconnection across contexts. At office, *Desktop* and *PDA* are interconnected via *wireless* connection. And at home, *Laptop* and *PDA* are interconnected via *USB*. **Data staging solution**: The data staging algorithm presents a solution in which documents on PDA and CellPhone are not relocated. While the documents on *Desktop* are moved to *Laptop* by using the *PDA* as the intermediary sotrage device.

Case2: There is no interconnection across contexts. At office, *Desktop* and *PDA* are interconnected via *wireless*

Algorithm 1 Algorithm for data staging in two contexts

1. Get all the available devices in both contexts, $D[i]$ and $D[j]$
2. Get a set of common devices, $CD[i,j]$ in both contexts where $CD[i,j] = D[i] \cap D[j]$
3. Retract all open documents on $CD[i,j]$, as there is no need to relocate them
4. Given the set of interconnections, we can infer several virtual interconnections between these devices using following rules:

$\exists d,m\ (m \in CD[i,j] \wedge L[d,i][m,i] \Rightarrow L[d,i][m,j])$
$\exists d,m\ (m \in CD[i,j] \wedge L[d,j][m,j] \Rightarrow L[m,i][d,j])$
$\exists d,e,f\ (L[d,i][f,j] \wedge L[f,j][e,j] \Rightarrow L[d,i][e,j])$
$\exists d,e,f\ (L[d,i][f,i] \wedge L[f,i][e,j] \Rightarrow L[d,i][e,j])$

5. For each $OD[c,a,d]$,$d \in D[i]$ with document type t and storage s:

Get all possible combination of application b and device e, denoted by $RLD[c,a,d,i,b,e,j]$ such that:

$e \in D[j]$ and
$s < Capa[e]$ and
$t \in DT[b]$, where $b \in PA[e]$ and
$\exists L[d,i][e,j]$ for $d \in D[i]$ and $e \in D[j]$ and
$PF[e,t] > 0$

6. For each combination in $RLD[c,a,d,i,b,e,j]$:

Select the combination with device $e \in D[j]$, such that $PF[e,t]$ has the highest value.

7. If a document c still appears in several combinations:

Select one of the combination at random with preference given to the combination that provides direct connection between $D[d,i]$ and $D[e,j]$

8. For each document c in a combination in $RLD[c,a,d,i,b,e,j]$, display output with the atomic set of operations required to achieve the goal using operators $\{O_c, O_o, O_t\}$

connection. **Data staging solution**: The data staging algorithm presents a solution in which documents on PDA and CellPhone are not relocated. While the documents on *Desktop* are opened on equivalent applications on *PDA*.

Case3: There is no interconnection across contexts. At office, *Desktop* and *CellPhone* are interconnected via *wireless* connection. **Data staging solution**: The data staging algorithm presents a solution in which documents on PDA and CellPhone are not relocated. While the text document on *Desktop* is opened on equivalent *CellPad* application on *CellPhone*. The rest of the documents, like *word document* and *excel document* can not be data staged.

B. Three Context Simulation

The configuration is that the user is moving from an 'office' context to 'home' context via 'car' context. The user has access to *Desktop* and *PDA* at office, *PDA* and *CellPhone* in car and *Laptop*, *PDA* and *CellPhone* at home.

Algorithm 2 Algorithm for data staging using intermediary context

1. Get all the available devices in all contexts, $D[i]$, $D[j]$ and $D[k]$
2. Get a set of common devices, $CD[i,j,k]$ where $CD[i,j,k] = D[i] \cap D[j] \cap D[k]$
3. Get a set of common devices, $CD[i,j]$ where $CD[i,j] = D[i] \cap D[j]$, $CD[j,k]$ where $CD[j,k] = D[j] \cap D[k]$ and $CD[i,k]$ where $CD[i,k] = D[i] \cap D[k]$
4. Retract all open documents on $CD[i,j,k]$, as there is no need to relocate them
5. Given the set of interconnections, we can infer several virtual interconnection between these devices using following set of rules:

$\exists d,m(m \in CD[i,j] \wedge L[d,i][m,i] \Rightarrow L[d,i][m,j])$
$\exists d,m(m \in CD[i,j] \wedge L[d,j][m,j] \Rightarrow L[m,i][d,j])$
$\exists d,m(m \in CD[j,k] \wedge L[d,j][m,j] \Rightarrow L[d,j][m,k])$
$\exists d,m(m \in CD[j,k] \wedge L[d,k][m,k] \Rightarrow L[m,j][d,k])$
$\exists d,m(m \in CD[i,k] \wedge L[d,i][m,i] \Rightarrow L[d,i][m,k])$
$\exists d,m(m \in CD[i,k] \wedge L[d,k][m,k] \Rightarrow L[m,i][d,k])$

$\exists d,e,f(L[d,i][f,j] \wedge L[f,j][e,j] \Rightarrow L[d,i][e,j])$
$\exists d,e,f(L[d,i][f,i] \wedge L[f,i][e,j] \Rightarrow L[d,i][e,j])$
$\exists d,e,f(L[d,j][f,k] \wedge L[f,k][e,k] \Rightarrow L[d,j][e,k])$
$\exists d,e,f(L[d,j][f,j] \wedge L[f,j][e,k] \Rightarrow L[d,j][e,k])$
$\exists d,e,f(L[d,i][f,k] \wedge L[f,k][e,k] \Rightarrow L[d,i][e,k])$
$\exists d,e,f(L[d,i][f,i] \wedge L[f,i][e,k] \Rightarrow L[d,i][e,k])$
$\exists d,e,f(L[d,i][f,j] \wedge L[f,j][e,k] \Rightarrow L[d,i][e,k])$

6. For each $OD[c,a,d]$,$d \in D[i]$ with document type t and document storage s:

Get all possible combination of application b and device e, denoted by $RLD[c,a,d,i,f,j,b,e,k]$ (i.e. relocate document c from application a on device d in context i to application b on device e in context k using intermediary device f in context j) such that:

$d \in D[i]$, $f \in D[j]$, $e \in D[k]$ and
$s < Capa[f]$ and
$s < Capa[e]$ and
$t \in DT[b]$, where $b \in PA[e]$ and
$\exists L[d,i][f,j]$ for $d \in D[i]$ and $f \in D[j]$ and
$\exists L[f,j][e,k]$ for $f \in D[j]$ and $e \in D[k]$ and
$PF[e,t] > 0$

7. For each $OD[c,a,d]$,$d \in D[i]$ with document type t and document storage s:

Get all possible combination of application b and device e, denoted by $RLD[c,a,d,i,f,j,b,e,k]$ such that:

$d \in D[i]$, $f \in D[j]$ and
$s < Capa[f]$ and
$t \in DT[b]$, where $b \in PA[f]$ and
$\exists L[d,i][f,j]$ for $d \in D[i]$ and $f \in D[j]$ and
$f \in CD[j,k]$ and
$PF[f,t] > 0$

8. For each combination in $RLD[c,a,d,i,f,j,b,e,k]$:

Select the combination with device $e \in D[k]$, such that $PF[e,t]$ has the highest value.

9. If a document c still appears in several combinations:

Select one of the combination at random.

10. For each combination in $RLD[c,a,d,i,f,j,b,e,k]$, display output with the atomic set of operations required to achieve the goal using: using operators $\{O_c, O_o, O_t\}$

Case1: At office, *Desktop* and *PDA* are interconnected via *wireless* connection. And *Laptop* and *PDA* are interconnected via *wireless* from car to office. **Data staging solution**: The data staging algorithm presents a solution in which documents on PDA are not relocated. While the documents on *Desktop* are opened on equivalent applications on *Laptop* using *PDA* as the intermediary device till the user reaches to car.

Case2: At office, *Desktop* and *PDA* are interconnected via *wireless* connection. And *Laptop* and *PDA* are interconnected via *wireless* at home. **Data staging solution**: The data staging algorithm presents a solution in which documents on PDA are not relocated. While the documents on *Desktop* are opened on equivalent applications on *Laptop* using *PDA* as the intermediary device till the user arrives at home.

C. Results

We have run the algorithms on a Pentium 4 2600MHz machine, under Windows XP. Each scenario was run 100 times and the results averaged. We obtained the following data:

Contexts	Devices	Connections	Open docs	Avg time
2	4	3	7	705 ms
2	4	2	7	650 ms
2	4	1	7	620 ms
3	5	3	10	1610 ms
3	5	2	10	1570 ms

We conclude that the algorithms performance is satisfactory for scenarios of realistic size.

VI. Related work

The computer persona is one of the approaches 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 cannot 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, it underutilizes the capabilities of the devices.

Invisible computer. The *invisible computer* [2] or *ubiquitous computing* [1] 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 [10] 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 [11]. Most of these approaches are adding value while effectively hiding from the user the fact that they are computer devices.

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. 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.

VII. 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] M. Weiser, "The computer for the 21st century," *Scientific American*, vol. 265, no. 3, pp. 84–104, September 1991.

[2] D. A. Norman, *The invisible computer*. Englewood Cliffs, NJ: MIT Press, 1999.

[3] "Jade webpage," URL `http://sharon.cselt.it/projects/jade/` .

[4] L. Bölöni and D. C. Marinescu, "An object-oriented framework for building collaborative network agents," in *Intelligent Systems and Interfaces*, ser. International Series in Intelligent Technologies, H. Teodorescu, D. Mlynek, A. Kandel, and H.-J. Zimmerman, Eds. Kluwer Publising House, 2000, ch. 3, pp. 31–64.

[5] I. Smith, P. Cohen, J. Bradshaw, M. Greaves, and H. Holmback, "Designing conversation policies using joint intention theory," in *Proceedings of International Joint Conference on Multi-Agent Systems (ICMAS-98)*, July 1998.

[6] D. Salber, A. K. Dey, and G. D. Abowd, "The context toolkit: Aiding the development of context-enabled applications," in *CHI*, 1999, pp. 434–441.

[7] M. Tan, M. Theys, H. Siegel, N. Beck, and M. Jurczyk, "A mathematical model, heuristic, and simulation study for a basic data staging problem in a heterogeneous networking environment," pp. 115–129, March 1998.

[8] C. H. Papadimitriou, *"ComputationalComplexity"*. Addison-Wesley, 1994.

[9] "Jess webpage," URL `http://herzberg.ca.sandia.gov/jess`.

[10] H. Abelson, D. Allen, D. Coore, C. Hanson, G. Homsy, T. F. Knight, R. Nagpal, E. Rauch, G. J. Sussman, and R. Weiss, "Amorphous computing," *Communications of the ACM*, vol. 43, no. 5, pp. 74–82, 2000. [Online]. Available: citeseer.nj.nec.com/abelson95amorphous.html

[11] D. Servat and A. Drogoul, "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, pp. 441–448.

0-7803-8255-2/04/$20.00 ©2004 IEEE.