Chapter 2
State of the Art in Migration

Fabio Paternò, Carmen Santoro and Rasmus Olsen

2.1 Introduction to Migration Frameworks

In recent years, research on interactive migratory services has started by a number of R&T labs.

For instance, one of the first works related to migration was the one of Bharat and Cardelli (1995), who put forward an initial solution for migrating entire applications (but without proving support for the user interface part), and which revealed to be problematic due to different execution environments and resources available in the involved devices.

Among the earliest works we can also cite the one carried out by the HCI group at Stanford University, which aimed to generate interfaces for information appliances for extensible collections of services, namely ICrafter (Interface Crafter). ICrafter is a service framework for a class of ubiquitous computing environments known as *interactive workspaces* (Ponnekanti et al. 2001). The main objective of ICrafter was to allow users of interactive workspaces to flexibly interact with the services contained in the workspace. This project was nevertheless limited to create support for controlling interactive workspaces by generating user interfaces for applications obtained by dynamic composition of elementary services, and thus did not provide support for migration and consequently continuity of task performance across different devices.

A number of previous European projects have considered some of the research issues addressed in this proposal, such as FP5 CAMELEON (Cameleon 2004) and CONSENSUS (Consensus 2004). In CAMELEON, issues related to the design of multi-device interfaces were considered. This project developed a good conceptual framework for addressing such issues, but it was only able to develop tools to provide support mainly at design time for multi-device, form-based user interfaces However, often there is a need for interfaces that are able to support multimodality and interactive graphics in such a way as to adapt at run-time. CONSENSUS Project

F. Paternò (✉)
CNR-ISTI, HIIS Laboratory, Via G. Moruzzi 1, 56124 Pisa, Italy
e-mail: fabio.paterno@isti.cnr.it

F. Paternò (ed.), *Migratory Interactive Applications for Ubiquitous Environments,*
Human-Computer Interaction Series, DOI 10.1007/978-0-85729-250-6_2,
© Springer-Verlag London Limited 2011

defined a device independent mark-up language (RIML, Renderer Independent Markup Language), which aimed to preserve the intent of the author and then transform the device-independent components into device specific ones through an adaptation system while retaining the author's control over the final result/presentation. Although the CONSENSUS project involved many companies (such as SAP, Nokia and IBM) showing the industrial relevance of these issues, and also paid a lot of effort to locate its work within international standardisation bodies, the main lack can be found in the fact that its approach was limited to design time support and little was done for automatic adaptation at run-time. Moreover, in (Bharat and Cardelli 1995), an agent-based application migration approach has been presented, in which agents carrying pieces of code and the state of the migratory application are sent from one host to another, where a server allows the agent to rebuild the migrating application. However, such an approach is not general suitable for our goals, since it is not able to adapt to several kinds of platforms, most of which can be mobile devices with potentially limited resources of power, storage and processing.

In the Pebbles project, carried out at CMU University in Pittsburgh (USA), a PUC (Personal Universal Controller) environment has been developed, which supports the downloading of logical descriptions of appliances and the automatic generation of the corresponding user interfaces (Nichols et al. 2002). However, the application area of this approach is limited to home appliances that require rather similar interfaces.

Another relevant project is Aura, whose goal was to provide an infrastructure for the mobile user that configures itself automatically (De Sousa and Garlan 2002). Thus, when a user moves to a different platform, Aura attempts to reconfigure the computing infrastructure so that the user can continue working on tasks started elsewhere. In this approach, the different context of use is supported by the selection of a similar application achieving the same goal (for example, text editing can be supported through MS Word or Emacs depending on the resources of the device at hand).

We aim to identify more flexible solutions where the application is still the same, but the interactive part is able to dynamically adapt to the new device and environment exploiting a wide variety of modalities. Some research work on migratory interfaces was described in (Bandelloni and Paternò 2004): however, that work was still far from identifying general solutions, which are for example able to handle migration through a variety of interactive devices supporting different combinations of modalities in such a way to preserve accessibility and usability. The migration onto one or multiple devices detected and classified regarding their possibilities in terms of task support requires complex and distributed processing. To achieve our goals in OPEN, we have analysed various types of software architectures able to support multimodal migration (considering both client-server and peer-to-peer solutions). It is important to find a solution which is able to better support users freely moving by dynamically generating multimodal user interfaces for mobile and stationary devices. This can be achieved through intelligent support able to provide usable results taking into account the users' characteristics, the tasks that the users aim to accomplish, and the resources of the devices available (in terms of

interaction support, network capabilities, etc.). In the user interface generation process, our system will use transformations involving device-independent languages, which enable the system to find general solutions and limit the part of the generation that is device-dependent.

In general, migratory services address a complex set of issues from the system viewpoint as well (Riva et al. 2006). They share some ideas and leverage work done on process migration (Milojicic et al. 2000), virtual machine monitors (Sapuntzakis et al. 2002), mobile agents (White 1997), active networks (Wetherall 1999), and dynamic reconfiguration in distributed systems (Kramer and Magee 1985). Research works that can be seen as precursors of our context-aware migration model include process migration for load balancing and service component offloading (Messer et al. 2002).

2.2 Support for Application Migration

2.2.1 Middleware Support for Migration

In ubiquitous computing, system support is provided by platform services, including service registration and discovery protocols as the most characteristic ones. UPnP (UPnP 1999) and Jini (Jini) are two relevant examples of such protocols. One straight forward approach to the seamless integration of heterogeneous devices, services and resources is to build specific bridges or gateways between them. However, this approach has practical limitations imposed by the complexity of integrating many potentially different bridges without the existence of a common infrastructure. Therefore, a more generic and suitable approach can be taken if a specific and unifying platform layer is placed on top of the heterogeneous devices, services and resources. In this regard, the OSGi specification (OSGi) provides a suitable and general framework (Lee et al. 2003). Other approaches adopt Jini as an integration platform, whilst ad-hoc solutions have also been proposed, e.g. (Allard et al. 2003). OSGi is oriented towards the implementation of residential gateways. It supports the integration and runtime combination of software components, known as bundles. The OSGi framework provides bundle installation, management and monitoring functionality which are of the utmost importance for both service providers and end users alike, and enables run time reconfiguration of components and their interaction useful for application reconfiguration. Typically a set-top box is used to host an OSGi framework runtime, and this residential gateway is in charge of integrating all the different devices, services and resources of the whole system, discovering components, checking dependencies, launching bundles on demand, and performing other related tasks. Common devices—a TV screen and a remote control—can be used as interfaces to the OSGi runtime, although many other options are also available through the use of different software bundles and communication media. Concerning user interfaces, there is a general trend towards the use of HTTP servers

as an evolution of the classical RPC model (Mattern and Sturm 2002). It is foreseeable that any mobile device will in the short-term be able to run an instance of a Web browser and support technologies that provide push mechanisms, such as Ajax (Garret 2005). Similar capabilities will also be present in future HDTV devices, and more devices will surely follow. Service migration is a fundamental functionality in intelligent environments and is particularly useful when paired with reliability and high availability features. Furthermore, component-based frameworks (e.g., Enterprise JavaBeans (JavaBeans) or Jini service containers) enable service download in ad-hoc networks, a key feature to provide adaptable service platforms for mobile end- user devices of any kind. Mobile devices, which accompany the user throughout their daily routines, should be integrated with the aforementioned residential gateway, so as to provide a coherent, seamless interaction with the system and perform valuable tasks for the user such as discovery of resources, interaction with applications and user monitoring and communication. Common problems many client devices share are related to their limited computational capabilities (due to size or power constrains). An approach to coping with these limitations is the one adopted by the Jini surrogate architecture, which allows a server to execute code on behalf of the device, in such a way that devices are only required to present the user interface to the application, leaving heavyweight tasks for the server.

In addition, the OPEN platform for migratory services contains mechanisms for configurability and adaptability to support the seamless device integration. This means that the service platform functionality can be configured for different device classes and adapted at runtime to available device resources, i.e. based on on-the-fly installation of core services to a device recently added to the environment. Based on this adaptive service platform, service components can also be dynamically installed and migrated to support the application migration in conjunction with user interface migration.

A new US commercial product, called Pluto (Pluto) shows that migration concepts start to raise industrial exploitation interest. However, Pluto seems still at a preliminary stage with respect to the migration potentialities: it is limited to redirect information streams depending on the user's position while we think about migration also as something which is able to keep the state of the user input in the source device(s) and make it available seamlessly to the target device(s), for example in applications such as games, interactive TV, and collaborative work.

2.2.2 Network Mobility Support for Migration

Network mobility support for migration includes the addressing and message routing between the devices participating in the migration as well as towards remote application end-points. Since we aim to make our migration platform transparent to the application server, regarding the support of connectivity to remote application end-points, several standard solutions exist that support such 'transparent connectivity' on different layers of the protocol stack. Most notably, mobile IP (mIP, version

4 (Perkins 2002) or 6 (Johnson et al. 2004)) on the network layer, transport layer mobility e.g. (Riegel and Tuexen 2006), or mobility support via session control signaling, e.g. based on the Session Initiation Protocol (SIP) (Rosenberg et al. 2002).

Mobile IP (MIP) has been designed by the IETF as network layer protocol to hide the mobility of network nodes from the upper layers of the protocol stack. MIP does so by always showing the mobile nodes' home address (HoA) to the higher layers instead of the actual mobile nodes' location (the Care-of-Address, CoA). MIP has been originally designed for terminal mobility, i.e. the user's terminal can change its access network while communicating but the actual device remains the same after a handover; MIP unfortunately does not offer means to operate a session handover (i.e. when an ongoing session is migrated from one device to another) or partial migration (at least a single media stream is migrated to a new device while part of the communication is maintained at the source device). One advantage of using MIP, though, is that only the nodes involved in the migration need to implement MIP, while the nodes that do not migrate a service but communicate with the migratory nodes do not need to implement MIP - this is particularly relevant when migratory nodes communicate with a 'remote' node such as a content server.

At the session layer, SIP provides session initiation, modification and termination. At first, SIP was designed by the IETF (Rosenberg et al. 2002) for multiparty video conference but was quickly used for managing IP-based multimedia sessions. 3GPP recently adopted SIP in release 5 of the UMTS specifications in order to implement a standardized platform for access control and session control, namely the IP-based Multimedia Subsystem (IMS) (TS23.228). On top of negotiating the end-to-end Quality of Service (QoS) and managing the media settings between the session endpoints, SIP can also be used for mobility. The original IETF SIP already defines the procedure for session mobility, i.e. maintaining the session settings after the session is moved to a new device.

In addition to the candidate mobility support solutions mentioned above, more recently other approaches with strong architectural implications have been discussed, see e.g. the Host Identity Protocol (Moscovitz et al. 2007) that operates between network and transport layer. All the existing mobility support solutions require adaptation and extensions in order to utilize them efficiently in the OPEN platform for service migration:

- Only individual flows may need to be re-directed in a QoS-aware manner, as only part of the application may be migrated. This re-direction may need to be transparent to the remote communication peers.
- The presence of fixed mobility anchor points (Home Agent or similar) may not be assured in certain ad-hoc communication scenarios.
- Mobility solutions and QoS-aware architectures often cannot be simply combined without resulting in severe inter-operability issues (see the example of mobile IP mobility support in a scenario of QoS- aware session signaling as in IMS (Renier et al. 2006)). Furthermore, QoS information can trigger and control the OPEN migration scenarios, hence close coupling of the mobility support mechanism and the QoS-aware migration solution is required.

2.2.3 Context Management Support for Migration

The Context Management support for Migration includes the management of context information that is used to trigger and support the migration procedure; this context includes user and device information, network/connectivity information, as well as service level information, namely the presence and capabilities of certain application parts on different end-systems.

Then, the main tasks are to define protocols and algorithms to gather relevant context information, and to make it accessible to context-sensitive functionalities. General frameworks for context-sensitive networking have been developed in the past, e.g. the Context Toolkit (Dey and Abowd 2000) or the Ad-hoc Context Aware Network architecture (ACAN) (Khedr and Karmouch 2002). However, such frameworks have to be adapted to the OPEN migration case which in particular includes answers to the following questions: what context information is relevant for the migration scenarios, how can it be obtained/measured (including active or passive measurement procedures for communication performance), what network and application domains are relevant for the context management procedures? Adaptations and extensions to other special use-cases of context-sensitive networking are investigated in other EU projects, e.g. for the scenario of Personal Networks (MAGNET-b 2004; Ghader et al. 2005). However, support of session and user interface migration has not been researched in these projects. One important part of the context information is the information about the service platform and application-level modules themselves. There is a set of off-the-shelf protocols and solutions that have been developed in the past, e.g. UPnP (UPnP), SLP (1999), Salutation (1999), Jini (Jini), but they all focus on service discovery, and while adaptation to delivering context according to the OPEN requirement is possible this will require major changes as to support the dynamic nature of context information.

2.3 Migratory Services for Games

In general terms there is not much literature and technology available on the subject of migration services for games. As a matter of fact, despite the high demand in industry, trends from major technology providers seem not much in favour of portability. From one side, the console market (e.g.: Xbox, PlayStation, Wii) is a very closed market: console makers usually ask a game to be made exclusively for their specific platform. From the other hand, there is the attempt to expand games on mobile phones. However, the lack of mature technologies on mobile phones makes this job a very manual and costly one. To run a game a vast amount of service platform software is needed, the major part of it being the “game engine”. Since there are not yet “game engines” that are portable from a PC-like platform to a mobile phone-like platform, usually the porting of a game to a mobile phone simply requires to redevelop the game from scratch, possibly developing also the necessary service

platform part yourself. It is worth mentioning that for this reason, there are some specialised companies that port games, with a low level coding activity. A recent summary of migratory technology for games can be found in (Mehdi et al. 2006), which, however, relies to some extent on a public deliverable published by FP6 project OLGA. Another FP6 integrated project, Games@Large (Games@Large) claims to support "ubiquitous game-play". This project targets a different usage scenario than OPEN, as they state in their mission: "to research, develop and implement a new platform aimed at providing users with a richer variety of entertainment experience in familiar environments, such as their entire house, hotel room, cruise ships and Internet Cafe." However, they do not provide any specific support for migration across different devices.

2.4 Advances Over the State of the Art

In this section we first discuss some advances in specific areas important for the book and then we summarize the important overall advances that we plan to achieve with the results presented in the following chapters.

2.4.1 Advances in Migratory User Interfaces

The increasing availability of various types of electronic interactive devices has raised interest in model-based approaches, mainly because they provide logical descriptions that can be used as a starting point for generating interfaces that adapt to the various devices at hand. In recent years, such interest has been accompanied by the use of XML-based languages, such as XIML (Puerta and Eisenstein 2002), UIML (Abrams et al. 1999), UsiXML (Limbourg and Vanderdonckt 2004), RIML (Ziegert et al. 2004), TERESA XML (Mori et al. 2004), in order to represent the aforementioned logical descriptions. However, most of such approaches focus on providing device-adaptation support only in the design and authoring phase, whilst we believe that run-time support is equally relevant.

The Personal Universal Controller (Nichols et al. 2002) automatically generates user interfaces for remote control of domestic appliances. The remote controller device is a mobile device, which is able to download specifications of the functions of appliances and then generate the appropriate user interface to access them. XML-based messages are exchanged to request a description of the appliance's features and to signal events to and from the device. However, the process of discovering the device is far from automatic and it might compromise usability since the user needs to manually enter the device's network address in the remote control application before any other action can be performed.

Another approach addressing the problem of interacting with services by means of a variety of interactive platforms is XWeb (Olsen et al. 2000), whose goal is

to provide an interactive (and collaborative) client/server architecture that can be accessed from any interactive platform. Clients interact with the server by means of CHANGE requests on the server's data tree rather than propagation of interactive events; clients are notified whenever server data are changed. Although our approach shares many concepts with this work (e.g.: the idea of platform independency, the adaptation to different devices), XWeb does not address the problem of migrating the user interfaces among different platforms.

Bharat and Cardelli (Bharat and Cardelli 1995) addressed the migration of entire applications (which is problematic with limited-resource devices and different CPU architectures or operating systems) while we focus on the migration of the user interface. Newman and others (Newman et al. 2002) have developed the Speakeasy recombinant computing framework, which is a middleware exploiting mobile code to allow components in different devices to extend one another's behaviour at runtime. However, they have not addressed the issue of adapting the services user interface to the interaction resources available in the device at hand, which we address through device-independent languages (and related transformations).

WebSplitter (Han et al. 2000) aims at supporting collaborative Web browsing by creating personalized partial views of the same Web page depending on the user and the device. Developers have to specify the Web content in XML and define a policy file indicating the tags content that can be presented depending on the user and the device. In addition, they have to define XSL modules in order to transform the XML content into HTML or WML. At run-time, a proxy server generates a presentation depending on user's privilege and the access device. This approach has several drawbacks. Developers have to manage a plethora of low-level details to specify XML content, policy files, and XSL transformations, and support for continuing tasks across various devices is not provided.

In ICrafter (Ponnekanti et al. 2001), services beacon their presence by periodically sending broadcast messages. A control appliance then requests a user interface for accessing a service or an aggregation of services by sending its own description, consisting of the user interface languages supported (i.e. HTML, VoiceXML) to an entity known as the Interface Manager, which then generates the user interface and sends it back to the appliance. ICrafter also shares the multimodal approach to user interaction, but it does not support the transfer of the user interface from one platform to another, maintaining the client-side state of the interface.

SUPPLE (Gajos et al. 2005) generates adaptive user interfaces taking functional specifications of the interfaces, a device model and a user model as input. The remote solver server that acts as the user interface generator is discovered at bootstrap by the client devices, and they can thus request rendering of interfaces to it once it is discovered. However, discovery is limited to the setup stage of the system, and it does not monitor the runtime status of the system, thus loosing some of the benefits that could arise from a continuous monitoring activity. SUPPLE does not support migration of a user interface from one device to another, but only adaption among different platforms.

Aura (Garlan et al. 2002) provides support for migration but the adopted solution has a limited granularity. In Aura, for each possible service, various applications

are available and the choice of a particular application depends on the interaction resources available in the user device. Thus, for a given service, such as word processing, if a desktop PC is available then an application such as Microsoft Word can be activated, whereas in the case of a mobile device a light-weight text editor would be chosen. In Aura the available services (such as the aforementioned text editing service) are registered in a global registry that is the base for matching the service requests. However, with this solution, changing the currently used application whenever a change of device occurs can be rather disorienting for the users since they not only would change the device in use, but also the application. In this situation, the users might experience discontinuity within the interaction. A better solution would be, instead, to allow the users to use on the different devices the same application which opportunely adapts to the device in use to exploit at its best the interaction capabilities currently at hand.

Luyten and Coninx (2005) present a system for supporting distribution of the user interface over a federation or group of devices. Migratability, in their words, is an essential property of an interface and marks it as being continuously redistributable. Their work shows the suitability of task model based design of user interfaces and XML-based user interface description languages for supporting user interface distribution and migration in device federations. The authors consider migration and distribution of only graphical user interfaces, while in OPEN we have a new solution supporting graphic, vocal and multimodal user interfaces migration.

In addition, it is worth pointing out that some interesting possibilities might be offered by different kinds of migration. Indeed partial migration (namely, the possibility for the user to select the parts to be migrated) has interesting connections with some other topics like application customisation and even End User Programming and Development (Liebermann et al. 2005). Indeed, if the users can select which parts of the UI to migrate, they are, to some extent enabled to "compose" and build their own customised applications.

Moreover, partial migration can be related to some extent also to the issues connected with Distributed User Interfaces. In this area, a toolkit for developing distributed graphical UIs is described in (Melchior et al. 2009). The toolkit is based on a widget distributed structure, composed of a 'proxy' of the widget (which remains stationary within the process that created the widget), and the 'renderer' of the widget (which is distributed, migratable and which the user can interact with). The toolkit is based on a peer-to-peer architecture in which a multi-purpose proxy is connected to one or more rendering engines able to render (partially or totally) a graphical user interface. In this solution the granularity of distribution is not limited to the user interface group's level but can also extend to the widget level and even down to the different components of the widget itself. In the solution considered within the OPEN Project we have instead opted for a distribution down to the granularity of the single UI element which the user can interact with but no deeper, since we judged such fine granularity not very relevant from the user's point of view since users are interested in moving some pieces of information sufficiently meaningful and not low-level details. In addition, this solution requires that the user interface be implemented using an extension of a specific toolkit (Tcl/Tk). Instead, we are interested in

solutions that mostly refer to standard technologies. For instance, when considering the migration of Web applications, we are interested in migrating any Web application developed with the standard Web languages (XHTML, CSS, and Javascript).

Another related work was the one from Dearman and Pierce (2008), who tried to achieve a better understanding of why and how people use multiple devices in their everyday life. They found out that users employ a variety of techniques for accessing and managing information across the various devices, but at the same time they reported that there is still room for improvement, especially as far as the user experience is concerned. Indeed, participants reported managing information across their devices as the most challenging aspect of using multiple devices. Then, migratory interfaces can be an important support from this viewpoint. In OPEN, we plan to offer better user experience and help the user to use effectively their devices.

A general reference model for user interfaces aiming to support migration, distribution and adaptation to the platform is proposed in (Balme et al. 2004). This book aims to explain how to design and implement a software architecture that is able to support migration of user interfaces associated with applications hosted by different systems and among automatically discovered devices, therefore resulting in a more concrete solution to the issue of multimodal migration of user interfaces.

2.4.2 Advances in the Migration of the Application Logic

2.4.2.1 Adaptive Middleware Technologies

There are numerous middleware solutions from the research community supporting proactive and reactive migration of application logic. These solutions are subsumed under the term adaptive middleware (Sadjadi 2003). They support not only the migration and (re-)wiring of various parts of the application but also the migration of the middleware itself.

According to (Sadjadi 2003) the key paradigms applied to adaptive middleware include reflection, component-based design, aspect-oriented programming and software design patterns. OPEN uses component-based design. In fact it will use the syntactical as well as the behavioral component interface descriptions for the identification and automated wiring of components. Other frameworks (e.g. MoCA 2006) use only text-based properties for looking-up components and therefore do not provide type safety or more sophisticated semantic checks.

Middleware solutions that support the migration of application can be further categorized in those that change the configuration of an application (i.e. the set of components that make up an application, the place of component execution and the established interconnections between the components) and in those that change the behavior of a component. OPEN focuses on the former in a reactive way. In other words it supports changing the configuration of an application (a) without the necessity of knowing all possible types of component services in advance and (b) without the explicit description of the migration behavior. OPEN will also support the component migration including the application state as well. In these cases components can react by using mechanisms for self-adaptation (e.g. reflection).

In (Oreizy et al. 1998) reactive system adaptation is also supported. The C2 (Taylor et al. 1996) architectural style comes into play when one sees architectural connectors as well as components as first class entities. The authors have developed an infrastructure that allows adding and removing components and connectors at runtime by using a special scripting language. Moreover they support modifying associations between connectors and components. Finally, a special validator component checks whether the modifications are valid. The motivation is different than OPEN in this case. The authors aim to support runtime system maintenance and not automated migration of parts of the application. In (Trapp 2005) an approach to proactive system and component adaptation is described. A special-purpose executable modeling language enables the explicit description of the adaptation behavior of embedded systems.

2.4.2.2 Component Dependency Management

Component dependency management refers to the management of the dependencies between service providing and service consuming components. The satisfaction of such dependencies is often subsumed under the term ‘wiring’. Wires can be compared to the C2 connectors discussed in the previous section.

Dependency Injection also creates wires automatically. The main idea of Dependency Injection is that service requestors are passive. They do not look up services; they only define needed services, which are delivered by the infrastructure. This enforces separation of concerns and simplifies testing.

One important feature is the migration support for application logic. In the book we show how to update the component wiring in the event of modification of the service landscape (i.e. service appearance, service unregistration, update). IoC containers (Fowler 2006) on the other hand typically configure an application during startup, without explicit support for reconfiguration at a later point in time. The latest version of Spring (2006) is an exception at this point.

2.4.3 Summary of Main Advances Over the State of Art

To summarise, the book extends the state-of-the-art in presenting comprehensive support for service migration as an inherent feature of services in heterogeneous device environments.

So, the main advances identified should enhance this situation in the following ways:

- *Ubiquitous access:* Migration will enable access to any service from any available interface (TV, computer screen, mobile, PDA, etc.). In this way the user does not have to learn diverse procedures to access different services. Consistent interfaces will be available in all devices, with similar interaction procedures.
- *Seamless access:* Since services are accessed through migratory interfaces, they can be dynamically transferred from one device to another maintaining the state

of the user session. In this way, the user has seamless and ubiquitous access to the same information, wherever they are.

- *Lower prices:* Since all the services can be used through off-the-shelf devices and the software should be able to dynamically install and configure itself, by adapting to the available devices, installation and maintenance expenses should thereby be considerably reduced.
- *Enhanced usability:* As the context information is used to obtain migration triggers (e.g. as caused by degrading QoS) as well as to delimit the suitable candidate set of destination devices, these semi-automatic decisions will detach the end-user more strongly from the underlying technological aspects of the migration, hence increase usability.
- *Personalization*, the possibility to obtain services and the associated user interfaces that have been customized by the system according to user profiles, preferences and knowledge.
- *User-decided customisation*s, apart from the possibility for the system to customize the migratable services depending on some user's knowledge and preferences, the OPEN platform, thanks to the possibilities offered by partial migration, will also support to some extent End User programming, by enabling users to build/compose their own applications,
- *Efficient use of network and device resources*, since part of the system support for migration will be to provide adequate QoS-aware mobility solutions, that allow migration of parts of an application while other participating nodes are unaware of this migration, the OPEN applications can also involve devices with low capabilities, as they do not all need to provide the full OPEN service platform solution.

References

Abrams, M., Phanouriou, C., Batongbacal, A., Williams, S., Shuster, J.: UIML: An appliance-independent XML user interface language. In: Proceedings of the 8th WWW Conference, Toronto, 11–14 May 1999

Allard, J., Chinta, V., Gundala, S., Richard III, G.G.: Jini Meets UPnP: An Architecture for Jini/UPnP Interoperability, pp. 268–275. SAINT (2003)

Balme, L., Demeure, A., Barralon, N., Coutaz, J., Calvary, G.: CAMELEON-RT: a software architecture reference model for distributed, migratable, and plastic user interfaces. In: Markopoulos P. et al. (eds.) Proceedings of EUSAI '04, Lecture Notes in Computer Science, vol. 3295, pp. 291–302. Springer Berlin (2004)

Bandelloni, R., Paternò, F.: Migratory user interfaces able to adapt to various interaction platforms. Int. J. Hum. Comput. Stud. **60**, 621–639 (2004)

Bharat, K.A., Cardelli, L.: Migratory applications. In: Proceedings of User Interface Software and Technology (UIST '95), pp. 133–142. Pittsburgh, 15–17 Nov 1995

Cameleon: Cameleon FP5 European project web site. http://giove.isti.cnr.it/projects/cameleon.html (2004)

Consensus: FP5 European project web site. http://www.consensus-online.org (2004)

Dearman, D., Pierce, J.: It's on my other computer!: Computing with multiple devices. In: Proceedings of the 26th Annual SIGCHI Conference on Human Factors in Computing Systems ACM CHI'08, Florence 5–10 April 2008, pp. 767–776.

Dey, A.K., Abowd, G.D.: The context toolkit: Aiding the development of context-aware applications. In: Proceedings of Workshops on Software Engineering for Wearable and Pervasive Computing, Limerick, 6 June 2000

De Sousa, J., Garlan, D.: Aura: An architectural framework for user mobility in ubiquitous computing environments. In: Proceedings of the 3rd Working IEEE-IFIP Conference on Software Architecture, Montreal (2002)

Fowler, M.: Martin Fowler's web page on dependency injection. http://martinfowler.com/articles/injection.html (2006). Accessed June 2006

Gajos, K., Christianson, D., Hoffmann, R., Shaked, T., Henning, K., Long, J.J., Weld, D.S.: Fast and robust interface generation for ubiquitous applications. In: Proceedings of UBICOMP'05: Ubiquitous computing, Lecture Notes in Computer Science, vol. 3660, pp. 37–55. Springer, Berlin Sept (2005)

Games@Large FP6 European Project. http://www.gamesatlarge.eu

Garlan, D., Siewiorek, D., Smailagic, A., Steenkiste, P.: Project Aura: toward distraction-free pervasive computing. IEEE Pervasive Comput. **21**(2), 22–31 (April–June 2002)

Garret, J.J.: Ajax: a new approach to web applications. Adaptive path, 18 Feb 2005. http://www.adaptivepath.com/publications/essays/archives/000385.php (2005)

Ghader, M., Olsen, R.L., Genet, M.G., Tafazolli, R.: Service management platform for personal networks. In: Proceedings of 1st Summit 2005, Dresden (2005)

Han, R., Perret, V., Naghshineh, M.: WebSplitter: Orchestrating multiple devices for collaborative web browsing. In: Proceedings of ACM Conference on Computer Supported Cooperative Work (CSCW), pp. 221–230. Philadelphia, 2–6 Dec 2000

JavaBeans technology. java.sun.com/products/ejb

Jini network technology. http://wwws.sun.com/software/jini/

Johnson, D., Perkins, C., Arkko, J.: Mobility support in ipv6, RFC 3775. Internet Engineering Task Force (IETF), June 2004

Khedr, M., Karmouch, A.: Enhancing service discovery with context information. IEEE Canadian Conference of Electrical and Computer Engineering, Brazil (2002)

Kramer, J., Magee, J.: Dynamic configuration for distributed systems. IEEE Trans. Softw. Eng. **11**(4), 424–436 (1985)

Lee, C., Nordstedt, D., Helal, S.: Enabling smart spaces with OSGi. IEEE Pervasive Comput. **2**(3), 89–94 (2003)

Lieberman, H., Paternò, F., Wulf, V.: End-user development. Springer, Netherlands (2005)

Limbourg, Q., Vanderdonckt, J.: UsiXML: A user interface description language supporting multiple levels of independence. In: Matera, M., Comai, S. (eds.) Engineering advanced web applications, pp. 325–338. Rinton Press, Paramus (2004)

Luyten, K., Coninx, K.: Distributed user interface elements to support smart interaction spaces. In: Proceedings of IEEE Symposium on Multimedia, Irvine, 12–14 Dec 2005

MAGNET-b: My personal adaptive global net. http://www.ist-magnet.org/ (2004)

Mattern, F., Sturm, P.: From distributed systems to ubiquitous computing—the state of the art, trends, and prospects of future networked systems. In: Proceedings of the Symposium on Trends in der Informationstechnologie am Beginn des 21. Jahrhunderts, pp. 109–134, May 2002

Mehdi, Q., Kumar, P., Salim, A., Bechkoum, K.: Content adaptation and shared state distribution for multiplayer mobile games. In: Proceedings of 9th International Conference on Computer Games: AI, Animation, Mobile, Educational & Serious Games, CGames '06, Dublin, 22–24 Nov 2006

Melchior, J., Grolaux, D., Vanderdonckt, J., Van Roy, P.: A toolkit for peer-to-peer distributed user interfaces: Concepts, implementation, and applications, EICS'09, pp. 69–78, Pittsburgh, 15–17 July 2009

Messer, A., Greenberg, I., Bernadat, P., Milojicic, D.S., Chen, D., Giuli, T.J., Gu, X.: Towards a distributed platform for resource-constrained devices. In: Proceedings of the 22nd International Conference on Distributed Computing Systems (ICDCS'02), pp. 43–51, Vienna, July 2002

Milojicic, D., Douglis, F., Paindaveine, Y., Wheeler, R., Zhou, S.: Process migration. ACM Comput. Surv. **32**(3), 241–299, Sept 2000

MoCA: Homepage of the MoCA framework. http://www.lac.inf.puc-rio.br/moca/ (2006). Accessed Aug 2006

Mori, G., Paternò, F., Santoro, C.: Design and development of multi-device user interfaces through multiple logical descriptions. IEEE Trans. Softw. Eng. **30**(8), pp. 507–520. IEEE Press, Aug 2004

Moscovitz, R., et al.: Host identity protocol, draft-ietf-hip-base-07 (work in progress). Internet Engineering Task Force (IETF), February 2007

Newman, M.W., Izadi, S., Edwards, W.K., Sedivy, J.Z., Smith, T.F.: User interfaces when and where they are needed: An infrastructure for recombinant computing. In: Proceedings of the UIST'02, Paris, 27–30 Oct 2002

Nichols, J., Myers, B.A., Higgins, M., Hughes, J., Harris, T.K., Rosenfeld, R., Pignol, M.: Generating remote control interfaces for complex appliances. In: Proceedings of ACM UIST'02, pp. 161–170, Paris, 27–30 Oct 2002

Olsen, D.R., Jefferies, S., Nielsen, S.T., Moyes, W., Fredrickson, P.: Cross-modal interaction using XWeb. In: Proceedings of UIST 2000: ACM SIGGRAPH Symposium on User Interface Software and Technology, pp. 191–200, San Diego, 2000

Oreizy, P., Taylor, R.N., Medvidovic, N.: Architecture-based runtime software evolution. In: Proceedings of the 20th International Conference on Software Engineering, pp. 177–186, Kyoto, 19–25 April 1998

OSGi Alliance. http://www.osgi.org/

Perkins, C.: IP mobility support for IPv4, RFC 3344. Internet Engineering Task Force (IETF), August 2002

Pluto.: http://plutohome.com/index.php?section=media_entertainment

Ponnekanti, S.R., Lee, B., Fox, A., Hanrahan, P., Winograd, T.: ICrafter: A service framework for ubiquitous computing environments. In: Proceedings of UBICOMP 2001, Lecture Note in Computer Science, vol. 2201, pp. 56–75 (Atlanta, 2001) ISBN:3–540-42614–0. Springer, London (2001)

Puerta, A., Eisenstein, J.: XIML: A common representation for interaction data. In: Proceedings of IUI 2002 (San Francisco, 13–16 Jan 2002), ACM, New York (2002)

Renier, T., et al.: MIPv6 operations in IMS-based access networks. In: Proceedings of WPMC'06, San Diego, Sept 2006

Riegel, M., Tuexen, M.: Mobile SCTP, draft-riegel-tuexen-mobile-sctp-07.txt (work in progress). Internet Engineering Task Force (IETF), Oct 2006

Riva, O., Nzouonta, J., Borcea, C.: Reliable migratory services in ad hoc networks. IEEE Trans. Mob. Comput. **6**(12), 1313–1328, Dec 2007 (2006)

Rosenberg, J., et al.: SIP: Session Initiation Protocol, RFC 3261. Internet Engineering Task Force (IETF), June 2002

Sadjadi, S.M.: A survey of adaptive middleware software engineering and network systems laboratory. Michigan State University, USA Technical Report MSU-CSE 3–35, 2003

Salutation: Architecture specification (Part-1), the salutation consortium. Available: http://www.salutation.org/ (1999)

Sapuntzakis, C.P., Chandra, R., Pfaff, B., Chow, J., Lam, M.S., Rosenblum, M.: Optimizing the migration of virtual computers. SIGOPS Oper. Syst. Rev. **36**(SI), 377–390 (2002)

SLP: Service Location Protocol svrloc—RFC2608, V2 ed., IETF, June 1999

Spring: Homepage of the spring framework. http://www.springframework.org/ (2006)

Taylor, R.N., et al.: A component- and message-based architectural style for GUI software. IEEE Trans. Softw. Eng. **22**(6), 390–406 June 1996

Trapp, M.: Modeling the adaptation behavior of adaptive embedded systems. München: Verlag Dr. Hut, 2005 Zugl.: Kaiserslautern, Techn. Univ. Diss. (2005)

TS23.228: 3rd generation partnership project, IP Multimedia Subsystem (IMS)—Stage 2, TS 23.228, v5.15.0, 3GPP, June 2006

UPnP: Universal plug'n'play. http://www.upnp.org/ (1999)

Wetherall, D.: Active network vision reality: lessons from a capsule-based system. In: Proceedings of the 17th ACM Symposium on Operating Systems Principles (SOSP 1999), pp. 64–79, Charleston, Dec 1999

White, J.: Mobile agents. In: Bradshaw. (ed.) Software agents. MIT Press, Cambridge (1997)

Ziegert, T., Lauff, M., Heuser, L.: Device independent web applications—the author once—display everywhere approach. In: Koch, N., Fraternali, P., Wirsing, M. (eds.) ICWE 2004, Lecture Notes in Computer Science, vol. 3140, pp. 244–255. Springer, Berlin (2004)