OPEN: Open pervasive environments for migratory interactive services

Anders Nickelsen
Dept. of Electronic Systems
Aalborg University, Denmark
an@es.aau.dk

Fabio Paternó
ISTI-CNR, Pisa, Italy
fabio.paterno@isti.cnr.it

Agnese Grasselli
Vodafone Italy
Milan, Italy
agnese.grasselli@vodafone.com

Kay-Uwe Schmidt
SAP, Darmstadt, Germany
kay-uwe.schmidt@sap.com

Miquel Martin
NEC Europe Ltd., Heidelberg, Germany
miquel.martin@neclab.eu

Björn Schindler
Technische Universität Clausthal, Germany
bjoern.schindler@tu-clausthal.de

Francesca Mureddu
Arcadia Design, Cagliari, Italy
francesca.mureddu@arcadiadesign.it

ABSTRACT

One important aspect of ubiquitous environments is to provide users with the possibility to freely move about and continue to interact with the available applications through a variety of interactive devices such as cell phones, PDAs, desktop computers, intelligent watches or digital television sets. Migratory applications are able to follow the user by sensing changes in the user's context and adapting to available devices, ideally without interrupting the user experience. However, applications themselves must contain functions to monitor context information, coordinate a migration, handle application adaptation and interact with the user during the migration process. To make life easier for developers and users of migratory applications, we propose an integrated Migration Service Platform (MSP), where all the common migration functions are centralised. We show how the platform is realised as middleware that contains a server for the central functions and lightweight client-side running on the end-user devices. We show how migratory applications can interact with the platform and thereby do not have to contain migration functions themselves. By using the platform, they can register and be controlled by the platform to enrich the user experience with the application.. We describe the challenges following the centralisation of a migration platform that can support different types of applications, both games and business applications, implemented with either web-technologies or as component-based applications.

Keywords

Migration Service Platform; Middleware; State Adaptation; Service Continuity; Context-Awareness;

1. INTRODUCTION

One important aspect of ubiquitous environments is to provide users with the possibility to freely move about and continue to interact with the available applications through a variety of interactive devices such as cell phones, PDAs, desktop computers, digital television sets or intelligent watches. In such environments one potential source of significant frustration is that people have to start their application session over again from the beginning after changing to a different interactive device. Migratory applications can overcome this limitation. Migratory applications, as defined by OPEN [1], are applications which are able to follow users, sense the user's context (where context is any information that can be used to characterise the situation of an entity [7]), and adapt to the changing context, e.g., set of available devices, while also preserving the continuity of application sessions, thereby ensuring the continuity of the tasks supported by the application. No proper migration occurs if there is a contextual change and an adaptation of the application features to the new device, but there is no continuity in the resulting user activity because, for instance, the user has to restart from the beginning when the new configuration is activated. Likewise, a situation in which there has been a context change, and also the state of the application has been preserved, cannot be properly called migration, if adaptation necessary, but not performed.

Therefore, migration encompasses three major aspects: *Context change*, which regards discovery, access and selection of context information; *adaptation*, which covers the problems of adapting the application to the characteristic of the new context based on the available context information; and *continuity*, on how to guarantee continuity in task performance. As is described in Section 3, there are already several approaches solving the parts of migration separately (e.g. for adaptation or continuity) but our approach is innovative as it provides a holistic solution for the migration problem.

The OPEN project provides an integrated solution to the migration problems able to address all three aspects in a Migration Service Platform (MSP), a middleware for migratory applications. This paper describes how the MSP handles the major challenges of migration by aggregating required functions that are shared between migratory applications into one middleware layer.

The rest of the paper is organised as follows. In Section 2, we describe the two domains of interest for the project, namely games and business applications. We exemplify the domains with relevant scenarios. Then, the requirements to the integrated platform that can support both domains are derived. Work related to migration, or parts of a migration process, that are relevant for OPEN are described in Section 3. In Section 4, we present the architecture of our proposed migration platform, and finally, Section 5 concludes the paper.

2. SCENARIOS AND REQUIREMENTS

Migratory applications that enable a continuous access across different devices can improve the overall user experience and provide new application use-cases. Ideally, the migration platform should be able to take all existing applications and make them migratory. In the OPEN project we have focused on specific classes of applications, in particular, Web applications and distributed applications in the game and business domains. Migratory applications existed in neither of these domains before the OPEN project, and the span of different application technologies enforces the platform to be general enough to support many new applications and technologies. Below, two representative scenario from the domains are presented and the requirements to the platform are derived from the scenarios subsequently.

2.1 Migratory games

Thomas is a college student who loves Formula 1 and spends several hours a day playing video games. He is used to watch all F1 Grand Prix races on his laptop while playing video games on his game console connected to the Plasma TV.
Thomas has left the study room of his college library just a few minutes before the start of the first Grand Prix of the season. He starts playing with the mobile phone while waiting for the bus on his way home. Thomas invites his friend Brad to be ready for the race by sending him a message through a chat service.
As Thomas gets home, his gaming and chatting sessions are migrated to his Plasma TV and he can continue playing the game, controlling it by the phone. The Grand Prix is going to start, so Thomas opens a HD window on the Plasma TV, to watch the Grand Prix. Now, he can watch high-definition F1 Grand Prix on one area of the screen, and at the same time virtually race his own car against the real pilots, while still having a look at the chat window. Brad joins the game and they compete together in the racing game.
Suddenly, Thomas' grandfather enters the sitting room, asking him to hold the ladder while he tries to fix a broken ceiling lamp in the kitchen. Since Thomas cannot go on playing while holding the ladder, he makes a pit stop in the game, and migrates the IPTV to the kitchen LCD and the game to the phone screen. The game dashboard is automatically migrated to the phone screen to keep Thomas informed about his car's state.
In the meanwhile Brad, who is still playing on his mobile, reaches Thomas' house and enters in the living room. Thomas closes his chat session from his mobile phone and Brad's game seamlessly switches from his mobile to the STB. After a few minutes Thomas gets back, just in time to see Hamilton winning the Grand Prix and Brad coming in second.

2.2 Migratory business applications

The business scenario is an emergency scenario where governmental agencies, organisations and companies work together in order to provide public security in emergency situations (e.g. flooding, large fires, and huge accidents). In such cases it is vitally important to have all the information available gathered together, so that adequate response plans can be made and eventual mistakes due to oversight can be reduced. Examples of such type of information are simulating and forecasting of flood consequences, water level, traffic data, etc.

Different experts, e.g. a flood and a traffic expert, can simulate in their own applications flood and traffic data on a map. The migratory emergency applications developed in OPEN makes it possible to gather and integrate all that information on one screen and one map. It enables experts to better analyse and plan response activities in an emergency situation. It gives them more flexibility for the visual representation of their data on a map because it supports the migration of different application components (as the traffic and flood simulations) to one target device. By migrating the necessary components or even whole applications to one target entity, experts have all the needed information overlaid on one map at their disposal. They can even synchronise the source and target devices so that virtual discussions how the emergency situation is eventually going to change are much easier to follow.

2.3 Migration process overview

The above scenarios describe several migrations. The general procedure and necessary steps for performing one migration from a source device to a target device are described in Figure 1. Initially, all devices and applications are registered, including device and network capabilities as well as application's requirements to devices and networks. Registration only occurs once, also for multiple migration. Then, migration can be triggered, either manually by the user or automatically by the application or as a reaction to contextual changes. After the trigger, the original application is paused to enable state extraction. The application state is adapted to suit the target device or network. Then, an application is instantiated on the target device, the adapted state is injected, and the new application can continue the session. Any persistent network connections to remote servers are redirected to the target device seamlessly.

2.4 Requirements to a common migration platform

Various important requirements need to be fulfilled by a platform that provide common functions to facilitate migra-

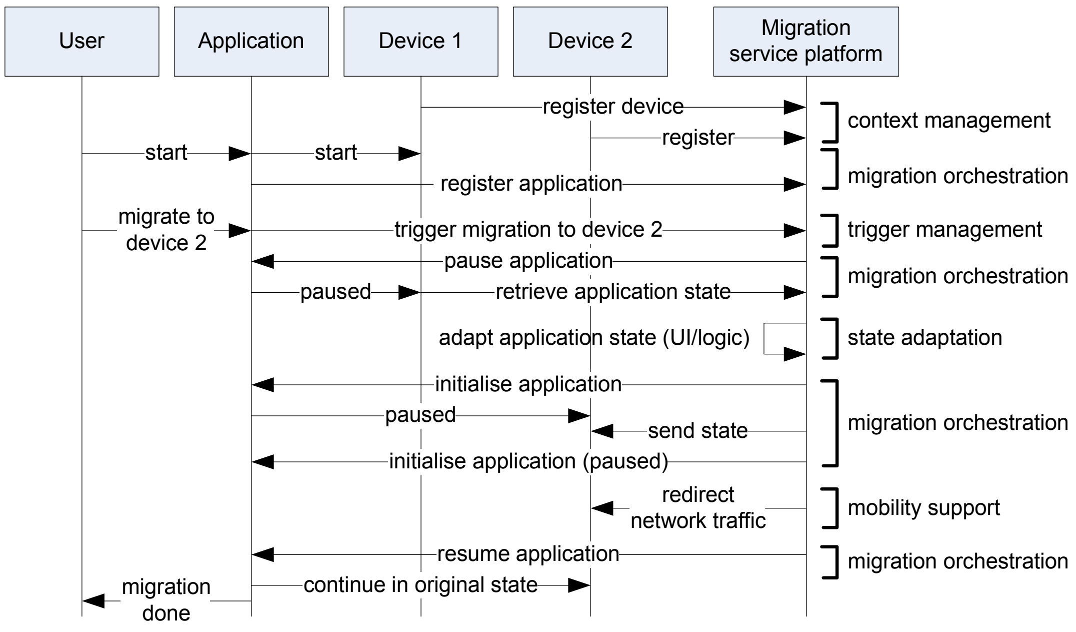

Figure 1: Migration procedure involving the user, the application, the two devices that the application migrate between and the underlying migration service platform.

tion. These requirements are derived from the above migration scenarios and are presented in the following.

Heterogeneous and pervasive: The platform needs to support several types of devices - both end user terminals and intermediate networking devices that are actively involved in the migration process. The platform must be able to fully utilise the varying capabilities of the involved devices and adapt application state according to changing conditions in the surrounding context.

Continuity: The procedures involving the devices must be seamless, in order to avoid interrupting the application users and device owners, also such that the user can continue operation after a migration. Seamless migration requires methods for state preservation on several levels, both application and system level, including the network. Migratory applications need to support extraction and injection of state information. The platform needs to support that the device and/or application may move between network. To avoid modifications of existing application servers, the platform must support transparent terminal and session hand-over between devices during migration.

Application technology support: Contemporary applications are realised using significantly different technologies ranging from traditional web pages over general rich internet applications to specifically targeted embedded or component-based applications. Such application have different requirements to performance as well as reliability, which must be respected and supported by the migration platform.

Intuitive user interaction: For users to accept the interaction paradigm of migratory applications, the interaction with applications as well as the platform must be straightforward, easy and intuitive. The platform must support automatic migration triggers as well as accept explicit and direct trigger request from the application user.

Migration dimensions: Application migration can be defined in several dimensions, which must be supported by a migration platform. Migration can be *full* where the entire application is migrated or *partial* where a subset of application elements are migrated. Moreover the purpose of migration can be *distribution*, where an application (full or partially) is cloned and distributed between available devices. Conversely, the purpose can be aggregation, where application elements are gathered into fewer devices than originally.

Secure: Migration must be secure since it may deal with personal and business information. The interaction between different users and devices owned by different parties present security challenges for the platform. Transport of information and exchange of application state information must be protected. Also, rights management must be ensured, such that unauthorised "stealing" of application sessions by triggering migration is prohibited.

3. STATE OF THE ART

Migration of applications combines aspects of several different research domains. In the following we present related work in the research areas of migratory user interface, reconfiguration of application logic and reconfiguration of networks.

3.1 Migratory User Interfaces

In recent years, a number of approaches have addressed the problem of interacting with applications in environments characterised by a wide variety of interactive platforms. SUPPLE [12] generates adaptive user interfaces taking functional specifications of the interfaces, a device model and a user model as input. Other researchers have investigated the use of overview techniques for supporting adaptation to mobile devices. For example, Lam and Baudish [18] proposed summary thumbnails, which consist in a thumbnail view of the original Web page, but the texts are summarised enabling a good legibility. WebSplitter [13] aims at supporting collaborative Web browsing by creating personalised partial views of the same Web page depending on the user and the device. More generally, all such approaches do not support migration of a user interface from one device to another, but provide only solutions for adaptation among different platforms. The issues related to device adaptation raised interest in model-based approaches for user interface design and generation, 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 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, since in this way it is possible to dynamically exploit the characteristics of the various devices, which is not a negligible aspect especially when mobile devices are considered. Bharat and Cardelli [4] 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. Luyten and Coninx [20] 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. The authors consider migration and distribution of only graphical user interfaces for desktop and mobile systems, while we provide a solution supporting migration for a broader set of interactive platforms (including vocal devices). In general, we can notice that there is a lack of general solutions able to make user interfaces completely or partly able to migrate without requiring any particular tool at development time.

3.2 Application reconfiguration

Dynamic reconfiguration of applications and especially their behavior deals with the adaptation of application logic during run-time based on continuously changing usage environments [10] [16]. The migration of an application is one use case for those kinds of systems as the usage environment usually changes during migration and therefore, the behavior should be seamlessly adapted. One way to realise such adaptability is to build the application out of interacting components. Kramer and Magee [17] proposed two main types of adaptation for those kinds of systems, namely the structural change in terms of component creation/deletion and its connection/disconnection. In addition to those structural changes, geographical changes, interface modification and implementation modification are further kinds of adaptation [2]. The characteristics of certain components are either known during development time or evaluated during run-time like in [5] for example. Several approaches have been developed so far offering solutions for those kinds of adaptations, many of them applied in the area of context-aware applications [19]. Among others, Floch et al. [9] proposed to use architecture models to realise run-time adaptability. Those architecture models define architectural properties like required components, their bindings, and performance requirements. A middleware is then responsible to fulfil these requirements during run-time and to take action if needed [21]. The integration of such adaptation mechanisms into a migration platform is important in order to provide the most appropriate behavior of an application before and after migration. How the module which realises such adaptability can be integrated in such a platform will be shown in later.

3.3 Mobile code, sessions and terminals

An obvious candidate research area for migratory service is mobile code, with mobile agents as the most used realisation of mobile code. Migration means moving (parts of) an application between computing entities, much similar to the moving agents of the mobile code paradigm. Code mobility is a well-studied area of distributed applications research, and in particular [11] is a renowned reference that defines code mobility as *the capability to reconfigure dynamically, at run-time, the binding between the software components of the application and their physical location within a computer network.* The idea behind mobile code is that by bringing the code close to the resources needed for a certain task it is possible to perform the task in a more effective way. For comparison, the goal of migration is to move the code close to where the users needs it, and adapt to the available resources to give the user a better experience performing the task. The resulting mechanisms of reaching either goal may be similar, but migration extends the existing research area with challenges of user interaction, code state adaptation and session continuity. Solutions for terminal mobility have seen a lot of research attention and can be approached on different layers in the network stack. MobileIP [29] [15] which is working at the network layer is the most transparent solution since any protocol in higher layers can be used with it. Mobile Stream Control Transmission Protocol (SCTP) or mSCTP [8] [32] which operates at the transport layer allows IP addresses to be added/deleted in the SCTP association and changing the primary address the peer will use when transmitting data to an endpoint. Session Initiation Protocol (SIP) [30] which operates at the application layer and can overcome terminal mobility [34] by letting the host experiencing the mobility sending a SIP INVITE message with the new IP address. SIP has been used for session mobility in [23] where a HTTP session is moved between Web browsers on different devices, and in [6] where SIP is used to split a combined audio and video stream into separate streams and transfer them between devices. Session mobility using SIP is also discussed in [34] [31] and while all these are working solutions they assumes that both end-points of the communication are SIP enabled, hence it is not suited for OPEN.

4. PLATFORM ARCHITECTURE

The architecture of the migration platform proposed by OPEN to address the migration challenges is presented in Figure 2. The platform is called the OPEN Migration Service Platform (MSP).

The MSP is realised as a middleware between the migratory application and the application execution platforms in terms of devices and networks. The MSP contains functionality to enable migratory applications to migrate between different execution platforms. By realising the functionality as middleware, the migration functionality shared between multiple migratory applications is aggregated into a general platform, such that the application developers can focus on application development and not migration development. Applications interaction with the MSP through a specifically defined interface [22].

The platform employs a client/server infrastructure to centralise information collection and decision making. The deployment of the middleware server and client parts is illustrated in Figure 3. Besides the central server, called the *migration server*, a network entity called a *mobility anchor point*, exists in the infrastructure to make migration of applications client-parts transparent to application server-parts that do not necessarily support migration.

In the following, an overview of the migration functions in the MSP is presented.

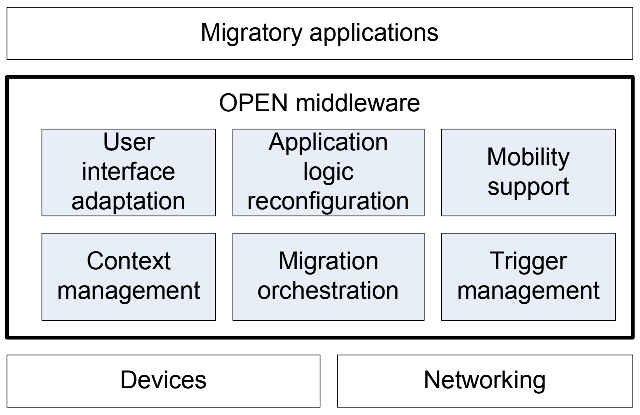

Figure 2: Architecture of the OPEN Migration Service Platform (MSP).

4.1 Deployment of the MSP middleware

A client-server architecture is employed in the OPEN middleware, which is illustrated in Figure 3.

The migration server contains the shared functions and is the point where most decisions are made, since the server is the central point of information. A server may reside on any device, so long as it is reachable by the clients. Typical deployment cases are in the user's home, in an enterprise's or operator's infrastructure or it may even be accessible via the Internet.

The migration client is typically the end user terminal (but may also be a large public display), on which migratory applications and a set of OPEN adaptors are running. The adaptors implement the part of migration functionality that is common across applications, and interact with OPEN server. They are meant to be reused across applications. In doing so, a migratory application needs only to make use of the adaptors to become migration capable. There exist an adaptor for all function that require client-side presence, as depicted in Figure 3. If an application does not make use of adaptors, it needs to implement the client-side migration

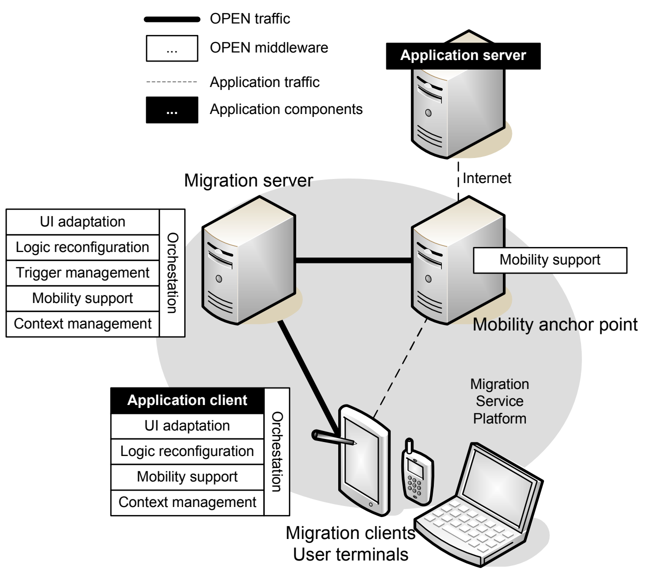

Figure 3: The MSP middleware deployment; a client part resides on the user terminal where also the application client-part is running and a server-part resides in a centralised server within the migration domain. Traffic is tunneled through a mobility anchor point to make the migration seamless for the application server.

functions.

4.2 User interface migration

Users can select either full or partial migration. In case of partial migration they can interactively select the user interface components to migrate. When the migration is triggered to an interaction platform other than the desktop, the migration server transforms its user interface by building the corresponding logical description through a reverse engineering process and using it as a starting point for creating the implementation adapted to the accessing device. In addition to interface adaptation, the environment supports task continuity. To this aim, when a request for migration to another device is triggered, the environment detects the state of the user interface, which depends on the user input (elements selected, data entered) and identifies the last element accessed in the source device. Then, a logical version of the interface for the target device is generated, and the state detected in the source device version is associated with the target device version so that the user inputs (selections performed, data entered) are not lost. Lastly, the user interface implementation for the target device is generated and activated remotely at the point corresponding to the last basic task performed in the initial device. In the process of creating an interface version suitable for a platform different from the desktop, we use a semantic redesign component. This part of the migration environment automatically transforms the logical description of the desktop version into the logical description for the new platform. Therefore, the goal of this transformation is to provide a description of the user interface suitable for the new platform. The focus of the OPEN project is to build the logic, semantic description of existing interactive applications and then dynamically generate

UIs that are adapted to various types of target devices and implementation languages, including with the state updated to the point at which it was left off in the previous device [28] [33].

4.3 Application logic reconfiguration

The Application logic reconfiguration module (ALR) supports applications at the dynamic adaptation of the application logic to their specific needs in constantly changing situations. At this, an application is divided into two parts, namely the reconfigurable application logic, and the rest of the application which could be among others static application logic and the User Interface. The ALR module is responsible for the adaptation of the reconfigurable part of the application logic. Challenges are the reaction on a situation change, management of many components as well as the description of the reconfiguration behavior and performance. To perform the reconfiguration the ALR module needs a description of the application and a description of the components. Therefore, the application and the components have to register at the ALR module. In the description of the application possible wiring rules and component behaviour are described in form of configurations. A component description provides information about the required and provided functionality of the component. In the project, effort has been dedicated to ensure correct and reliable dependencies through the life-time of applications based on dynamic component [26]. To react on a change of the situation the ALR module additionally needs information about the given context. This can be provided by the context management framework. At a change of context information the ALR module has to be informed. In this case the ALR computes the best configuration to the given context and initiates the reconfiguration of the application logic.

4.4 Mobility support

In scenarios where devices or applications change network as a part of the migration process the change must be transparent to the application server (and client) in order for them not to require a reconfiguration of e.g. IP addresses. The network may be changed during run-time for several reasons, including congestion in the current network or platform mobility between networks. A example scenarios is when the original device uses a wireless network and the target device of migration uses a fixed network. This cannot be solved by traditional terminal mobility solutions.

The primary objective of the *mobility support* function is to ensure that network changes do not affect the communication between the client and the application servers, with the risk breaking task continuity. OPEN solves this problem by implementing a SOCKS-based proxy server, called the mobility anchor point (MAP) in Figure 3, between the application client and server. The behavior of the MAP is controlled by the migration server and it handles seamless hand-over of connections during migration, which are transparent to remote application servers [14]. Traditional terminal mobility challenges are addressed by deploying MobileIP in the infrastructure.

4.5 Migration orchestration

The Migration Orchestration, shortly Orchestration, manages the migration process. It consists of the Orchestration server and Orchestration clients. In OPEN, communication between server and clients are performed with the standard XML-Remote Procedure Call (XML-RPC) specification over the HTTP protocol. The Orchestration Server is responsible of:

- The registration/deregistration of the devices connected to the platform.
- The registration/deregistration of the application components.
- The migration process: it receives the migration trigger and directly implements and manages the migration.

The Orchestration implements the migration thought the following phases:

- Application stop on the source device.
- Platform modules orchestration: each adaptation module can contribute to the overall migration process (e.g.: for user interface or logic adaptation).
- State maintenance: the Orchestration saves and transfers the state to the target device;
- Adapted application restart on the target device.

The Orchestration client is the distributed part of the OPEN Migration Service Platform and it provides the required migration features on the devices. It manages device and application components registration on the Orchestration Server. It also provides the functionality to start/stop the application and to save/recover/synchronise the actual state of the application itself. The interaction between the migratory application and the local Orchestration client is managed using a client-server protocol local to the device on which the application is installed. In addition to the XML-RPC based communication, a lean orchestration protocol has been developed for resource-constrained network link, such as for instance based on Near-Field Communication, where the time-window for orchestration signalling and state transfer is short. The main design changes concerned moving migration decisions to the target device to avoid long network delays and to reduce the size of the state object [24].

4.6 Trigger management

The purpose of the *trigger management* (TM) module is to decide the configuration of the migratory system and its applications in order to fulfil the requirements of the user and the applications.

TM decides which configuration to choose based on a set of configurations made available by Migration Orchestration. Migration Orchestration collects information from the registered device about which application components are registered, and decides which configurations are runnable in certain context settings. A configuration is a set of application components on a set of devices, which is determined runnable by the UI adaptation and ALR components. Deciding which configuration to use is done either manually, directly by the user, or automatically based on observations of contextual information from the environment such as device and network capabilities or application requirements (e.g. a game switching from single-player to multi-player). The automatic migration triggers are generated based on a assessment of the user experience quality that the available configurations will give. The configuration that maximises the quality in the current context is chosen. A migration is triggered if the chosen configuration differs from the current.

Two solutions have been evaluated for optimal automatic choice of configuration with dynamic and only partially observable system state. Both methods evaluate based on configuration utility, which is represented as a function of system state. One fast but inaccurate method is based on simple threshold comparisons where the other builds on a Markov Decision Process (MDP) model of the TM, which is more complex to calculate but more accurate during runtime [25].

4.7 Context management

The situation of the user, user's activity, the network state or other information that describes the situation of potential candidate devices for target migration, are good indicators on when and where to migrate and is the basis for the decisions taken in the Trigger Management and for the adaptation of the application state. The Context Management system in the platform ensures easy access for other modules, applications and services to distributed, dynamic information of various types within the network, and offers search, discovery, access and distribution functionality of context information. In OPEN, an existing platform, [3], was extended to run under OSGi in order to accommodate requirements on information reconfigurability. The shift to OSGi means that context measuring and computing sub components (retrievers and processing units, [3]) can be plugged in and out as needed at run time hereby allowing a simple adaptation to the various target platforms found in the OPEN scenarios. Furthermore, the impact of user mobility on reliability of access to dynamic location information was studied in [27] where we show how a Context Management system can deliver location information, which is a key information element in the OPEN scenarios. The Context Management system maintains a certain reliability by adjusting the accuracy of location estimates from a positioning system, based on information about the user's mobility.

5. CONCLUSION

Migration of applications is a new and challenging area spanning several, different domains of research such as user interface migration, reconfiguration of application logic, session and network mobility, context information management, and future network of services as well as a novel application paradigm called *migratory applications*. In this paper we propose a platform to facilitate migratory applications called the OPEN Migration Service Platform (MSP). The MSP enables the development and execution of migratory applications, which can have three primary capabilities: user interface migration, application logic reconfiguration and network reconfiguration. The platform contains functions to deal with the various requirements that must be fulfilled to support migratory applications. These functions are; Application Adaptation (user interface / application logic), Mobility Support, Context and Trigger Management and finally Migration Orchestration. The requirements and the following functions have been derived using two motivating scenarios, namely a migratory game and a migratory business application for handling collaboration during emergency situations. The functions in the platform are realised as middleware software in a network infrastructure on the devices running the applications, a migration server to coordinate and orchestrate the migration process and adapt the migratory application to suit the device capabilities, and a mobility anchor point to migrate network connections when migrating over different networks. The middleware can be used by migratory applications to carry out everything during the migration process, or the applications can implement subsets of the migration functions themselves (for instance adaptation functions). To use the middleware functions, migratory applications use a defined interface to the platform.

6. ACKNOWLEDGMENTS

This work was supported by the EU ICT FP7 project 'Open Pervasive Environments for iNteractive migratory services – OPEN', see www.ict-open.eu.

7. REFERENCES

[1] www.ict-open.eu.

[2] M. Aksit and Z. Choukair. Dynamic, adaptive and reconfigurable systems overview and prospective vision. In *Distributed Computing Systems Workshops, 2003. Proceedings. 23rd International Conference on*, pages 84–89, 2003.

[3] M. Bauer, R. Olsen, M. Jacobsen, L. Sanchez, M. Imine, and N. Prasad. Context management framework for MAGNET Beyond. In *Workshop on Capturing Context and Context Aware Systems and Platforms, Proceedings of IST Mobile and Wireless Summit.* Citeseer, 2006.

[4] K. Bharat and L. Cardelli. Migratory applications. *Mobile Object Systems Towards the Programmable Internet*, pages 131–148, 1997.

[5] J. Camara, C. Canal, and G. Salaun. Behavioural self-adaptation of services in ubiquitous computing environments. In *Software Engineering for Adaptive and Self-Managing Systems, 2009. SEAMS '09. ICSE Workshop on*, pages 28 –37, 18-19 2009.

[6] M.-X. Chen and F.-J. Wang. Session mobility of sip over multiple devices. In *TridentCom '08: Proceedings of the 4th International Conference on Testbeds and research infrastructures for the development of networks & communities*, pages 1–9. ICST, 2008.

[7] A. Dey. Understanding and using context. *Personal and ubiquitous computing*, 5(1):4–7, 2001.

[8] R. S. et al. Stream Control Transmission Protocol (SCTP) Dynamic Address Reconfiguration. RFC 5061 (Proposed Standard), Sept. 2007.

[9] J. Floch, S. Hallsteinsen, E. Stav, F. Eliassen, K. Lund, E. Gjorven, S. ICT, and N. Trondheim. Using architecture models for runtime adaptability. *IEEE software*, 23(2):62–70, 2006.

[10] S. Friedberg. Transparent reconfiguration requires a third-party connect. *TR220, Computer Science*

Department, University of Rochester, New York, 1987.

[11] A. Fuggetta, G. Picco, and G. Vigna. Understanding code mobility. *IEEE Transactions on software engineering*, 24(5):342–361, 1998.

[12] K. Gajos, D. Christianson, R. Hoffmann, T. Shaked, K. Henning, J. Long, and D. Weld. Fast and robust interface generation for ubiquitous applications. *UbiComp 2005: Ubiquitous Computing*, pages 37–55, 2005.

[13] R. Han, V. Perret, and M. Naghshineh. WebSplitter: a unified XML framework for multi-device collaborative Web browsing. In *Proceedings of the 2000 ACM conference on Computer supported cooperative work*, page 230. ACM, 2000.

[14] K. Højgaard-Hansen, H. C. Nguyen, and H.-P. Schwefel. Transparent connectivity in service migration scenarios. In *In preparation*, 2010.

[15] D. Johnson, C. Perkins, and J. Arkko. IP Mobility Support in IPv6. RFC 3775 (Proposed Standard), June 2004.

[16] J. Kramer and J. Magee. Dynamic configuration for distributed systems. *IEEE Transactions on Software Engineering*, pages 424–436, 1985.

[17] J. Kramer and J. Magee. The evolving philosophers problem: Dynamic change management. *IEEE Transactions on software engineering*, 16(11):1293–1306, 1990.

[18] H. Lam and P. Baudisch. Summary thumbnails: readable overviews for small screen web browsers. In *Proceedings of the SIGCHI conference on Human factors in computing systems*, pages 681–690. ACM, 2005.

[19] S. Loke. *Context-aware pervasive systems: architectures for a new breed of applications*. Auerbach Pub, 2006.

[20] K. Luyten and K. Coninx. Distributed user interface elements to support smart interaction spaces. In *Seventh IEEE International Symposium on Multimedia*, page 8, 2005.

[21] M. Maia, L. Rocha, and R. Andrade. Requirements and challenges for building service-oriented pervasive middleware. In *Proceedings of the 2009 international conference on Pervasive services*, pages 93–102. ACM, 2009.

[22] Martin, M. et al. D4.2: Migration Service Platform Design. Technical report, ICT-OPEN EU FP7 project, 2009.

[23] W. Munkongpitakkun, S. Kamolphiwong, and S. Sae-Wong. Enhanced web session mobility based on sip. In *Mobility '07: Mobile technology, applications, and systems, Singapore*, pages 346–350, 2007.

[24] A. Nickelsen, M. Martin, and H.-P. Schwefel. Service migration protocol for nfc links. In *To appear in proceedings of EUNICE 2010*, 2010.

[25] A. Nickelsen, R. L. Olsen, and H.-P. Schwefel. Model-based decision framework for autonomous application migration. In *Submitted to IEEE Globecom*, 2010.

[26] D. Niebuhr, A. Rausch, C. Klein, J. Reichmann, and R. Schmid. Achieving Dependable Component Bindings in Dynamic Adaptive Systems-A Runtime Testing Approach. In *2009 Third IEEE International Conference on Self-Adaptive and Self-Organizing Systems*, pages 186–197. IEEE, 2009.

[27] R. L. Olsen, J. Figueiras, J. Rasmussen, and H.-P. Schwefel. How precise should localization be? - A quantitative analysis of the impact of delay and mobility on reliability of location information. In *Submitted to IEEE Globecom*, 2010.

[28] F. Paterno, C. Santoro, and A. Scorcia. Ambient Intelligence for Supporting Task Continuity across Multiple Devices and Implementation Languages. *The Computer Journal*, 2009.

[29] C. Perkins. IP Mobility Support for IPv4. RFC 3344 (Proposed Standard), Aug. 2002.

[30] J. Rosenberg, H. Schulzrinne, G. Camarillo, A. Johnston, J. Peterson, R. Sparks, M. Handley, and E. Schooler. SIP: Session Initiation Protocol. RFC 3261 (Proposed Standard), June 2002.

[31] H. Schulzrinne and E. Wedlund. Application-layer mobility using sip. *SIGMOBILE Mob. Comput. Commun. Rev.*, 4(3):47–57, 2000.

[32] M. L. Seok Joo Koh, Moon Jeong Chang. msctp for soft handover in transport layer. *IEEE Communication letters*, 8:189 – 191, 2004.

[33] R. St
"uhmer, D. Anicic, S. Sen, J. Ma, K. Schmidt, and N. Stojanovic. Lifting events in rdf from interactions with annotated web pages. *The Semantic Web-ISWC 2009*, pages 893–908.

[34] E. Wedlund and H. Schulzrinne. Mobility support using sip. In *WOWMOM '99: Proceedings of the 2nd ACM international workshop on Wireless mobile multimedia*, pages 76–82, New York, NY, USA, 1999. ACM.